**DEPARTAMENTO DE INGENIERÍA TELEMÁTICA**

**UNIVERSIDAD CARLOS II DE MADRID**

**ESCUELA POLITECNICA SUPERIOR**


# PROYECTO FINAL DE CARRERA

# SISTEMA PARA GESTIÓN DE INSTALACIONES BASADO EN TRYTON Y ANDROID

**Autor:**
Francisco Guillermo Peiró Martínez

**Tutores:**
Jose Jesús García Rueda
Ignacio Salvat Ojembarrena

**LEGANÉS, MADRID**
**Diciembre 2013**

**Título:** Sistema para gestión de instalaciones basado en Tryton y Android

**Autor:** Francisco Guillermo Peiró Martínez

**Tutores:** Jose Jesús García Rueda

Ignacio Salvat Ojembarrena

**EL TRIBUNAL**

Presidente (Iria Estévez Ayres):

Secretario (David Díez Hernández):

Vocal (Rosa Romero Gómez):

Realizado el acto de defensa y lectura del Proyecto de Fin de Carrera el día 10/12/2013 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la calificación de: _____

Fdo. El Presidente Fdo. El secretario Fdo. El vocal

# AGRADECIMIENTOS

En primer lugar, me gustaría agradecer a Jose Jesús García Rueda, por la paciencia, confianza y comprensión que ha hecho que finalmente pueda terminar esta etapa de mi vida y por supuesto por todos sus años de dedicación en la Universidad Carlos III de Madrid, cualquier persona que haya presenciado tan sólo una hora de sus clases estoy convencido de que no tendrá ni un sólo mal recuerdo suyo. Nunca olvidaré su clase magistral “Caja negra, caja blanca”.

A Iñigo; amigo, compañero y jefe que me ha proporcionado todos los recursos que he necesitado para poder realizar este proyecto. El tiempo de trabajo en la empresa ha sido un fuerte impulso en mi carrera profesional, y he aprendido en ella como se deben hacer las cosas (Joserra, Enrique, Belén, Mariel, Dani e Isabel, gran equipo de personas) y por qué no decirlo como no se deben hacer otras (SA). Iñigo, parece que en estos tiempos el “joven emprendedor” está muy valorado y aunque ya te queda poco pelo de tanto pensar tú tienes todas estas cualidades; porque no he conocido persona más activa y con la devoción profesional más grande que tú; eres un maestro para mí.

No se me ocurre la manera de agradecer a mis padres lo que han hecho siempre por mí. Gracias a su sacrificio, esfuerzo y generosidad, que no se me olviden también sus tirones de orejas, hoy soy lo que soy. Durante toda su vida se han preocupado de que no me faltara de nada y aunque muchas veces no parezca agradecéroslo que sepáis que sois un ejemplo para mí en todo y que aunque pasen los años siempre aprenderé de vosotros.

A mis hermanos, Jose y Ali, los mejores del mundo. Hacemos un trio perfecto, no conozco hermanos que se parezcan a nosotros. Ali la responsable y metódica; y Jose el

impulsivo e inquieto. Ambos con sus estudios finalizados, no iba a dejar el triplete sin completar, ellos sabían que lo iba a conseguir.

A todos mis grandes amigos de la Universidad, cuanto hombro en el que llorar las penas, porque que mal lo hemos pasado todos y quien diga que no sabe que miente. Finalmente estos de la Carlos III eran más duros de lo que parecían en aquellas presentaciones de las asignaturas. Menos mal que después de los fracasos venían bromas, anécdotas, locuras, cotilleos y cosas sin sentido que, sin todas ellas esta etapa no se hubiese podido llevar de ninguna manera. Que grupo fantástico hemos sido, que diferentes y que iguales entre nosotros. Gracias a Manu, Raúl, Carlos, Santi, Javo, Vaillo, Miki, Rober, Fer, Natalia, Rosa, Bea, Recu, Elena, Homobono, Arancha.... ¡sois lo más!, solo puedo quitarme el sombrero ante vosotros.

Por supuesto a todos mis amigos de esta vida y de la siguiente Javi, María, Anita, Ana y Rosana ya tenéis al ingeniero loco que tanto deseasteis, lo de ingeniero me ha costado un poquito más de lo esperado.

Para el final de los agradecimientos, a Esther. ¡Un doctorado!… un doctorado la tendrían que dar a usted en ingeniería, filosofía, psicología, pedagogía, paciencialogía, todo lo que termine en “gía” y lo que no también. Decepción tras decepción, tú le has echado a este camino las dosis de genio, fuerza, respeto, confianza y cariño necesario para que esto terminara algún día. Todavía no se ha inventado la manera de que yo te pueda agradecer todo lo que has hecho por mí este tiempo, aunque lo inventaré algún día, te lo prometo. Sin duda eres lo mejor que me ha pasado en la vida, la persona que mejor me conoce y la que mejor sabe hacerme levantar en todo momento. No hace falta que te diga que has sido mi gran apoyo, los cimientos para que por fín pongamos la bandera en la cima, y gritemos… ¡Lo hemos conseguido!

# RESÚMEN

Este proyecto de fin de carrera describe parte de un proyecto desarrollado en el ámbito de la empresa privada en el sector de las TIC, concretamente en Infortrex (Outsidebuilder S.L.).

El objetivo del proyecto es el análisis, diseño e implementación de un sistema completo para la gestión de unas tareas concretas. Consiste en la creación de un módulo ERP, basado en Tryton, integrado con una aplicación Android para dispositivos móviles, con el fin de gestionar una serie de instalaciones de dispositivos Bluetooth en una gran cantidad de locales comerciales. El objetivo es el de dotar de una cierta automatización a la gestión de un gran volumen de instalaciones de una red de proveedores para una PYME con recursos limitados.

Las necesidades del cliente hacen que se deba plantear una eficaz solución y que suponga el mínimo coste para la empresa debido a la limitación en recursos, pero no sin ello escatimar en funcionalidad, debido a la posible ampliación a nuevos módulos de gestión como por ejemplo procesos de facturación o control de stock entre otros. En este documento se realiza una completa descripción de la solución implementada así como el procedimiento de gestión de proyecto llevada a cabo.

# ÍNDICE


**1. INTRODUCCIÓN 10**

**1.1 CONTEXTO 11**
**1.2 MOTIVACIÓN 13**
**1.3 OBJETIVOS 14**
**1.4 CONTENIDO DE LA MEMORIA 16**

**2. TRYTON: MÓDULO ERP PARA GESTIÓN DE INSTALACIONES 18**

**2.1 ESTUDIO COMPARATIVO: OPENERP VS TRYTON 22**
**2.2 ENTORNO TRYTON 26**
*2.2.1 INTRODUCCIÓN A TRYTON 26*
*2.2.2 PLATAFORMA DE TRES CAPAS 27*
*2.2.3 MODELOS Y VISTAS 28*
**2.3 DISEÑO DE MÓDULO ERP 40**
*2.3.1 REQUISITOS 41*
*2.3.2 DESCRIPCIÓN DE TAREAS 49*
*2.3.3 DISEÑO DE LOS MODELOS 53*
*2.3.4 DISEÑO DE LAS VISTAS 74*
**2.4 IMPLEMENTACIÓN DEL MÓDULO 83**
*2.4.1 CREACIÓN DEL MÓDULO 83*
*2.4.2 IMPLEMENTACIÓN DE LOS MODELOS 85*
*2.4.3 IMPLEMENTACIÓN DE LAS VISTAS 88*
*2.4.4 GENERACIÓN DE PEDIDOS 95*

**3. APLICACIÓN ANDROID PARA INSTALADORES 100**

**3.1 ENTORNO ANDROID 107**
*3.1.1 ESTRUCTURA DE UN PROYECTO ANDROID 109*
*3.1.2 COMPONENTES DE UNA APLICACIÓN ANDROID 116*
**3.2 DISEÑO DE LA APLICACIÓN 125**

*3.2.1 Requisitos* *126*
*3.2.2 Descripción de tareas* *130*
*3.2.3 Diseño de layouts* *134*
*3.2.4 Diseño de actividades* *150*
**3.3 Desarrollo de la aplicación** **171**
*3.3.1 Creación de la aplicación* *171*
*3.3.2 Implementación de layouts* *171*
*3.3.3 Implementación de actividades* *191*
*3.3.4 Back-end: Servicios Web REST* *223*

**4. SISTEMA COMPLETO 236**

**4.1 Arquitectura** **236**
**4.2 Esquema completo** **238**
**4.3 Descripción del sistema completo** **241**

**5. CONCLUSIONES 247**

**6. BIBLIOGRAFÍA Y ENLACES DE INTERÉS 250**

**7. PRESUPUESTO Y TIEMPO DEDICADO 253**

**7.1. Escenario de desarrollo** **253**
*7.1.1. Coste de personal* *253*
*7.1.2. Coste de material* *255*
*7.1.3. Costes indirectos* *255*
*7.1.4. Total* *256*
**7.2. Escenario de producción** **256**
*7.2.1. Coste de personal* *256*
*7.2.2. Coste de material* *257*
*7.2.3. Costes indirectos* *258*
*7.2.4. Total* *258*
**7.3. Tiempo dedicado** **259**

**8. ANEXOS 263**

**I. MANUAL DEL MÓDULO TRYTON 263**
**II. MANUAL DE LA APLICACIÓN ANDROID 281**
**III. ENTORNO DE DEMOSTRACIÓN 294**

# 1. Introducción

Este proyecto final de carrera describe las fases de diseño e implementación de un proyecto desarrollado en el ámbito de la empresa privada en el sector de las TIC[1], concretamente en una pequeña empresa nacional denominada Infortrex [1].

El proyecto consiste en la creación de una plataforma multidisciplinar capaz de otorgar a la empresa de una automatización en las tareas que intervienen en el registro del trabajo de una red de proveedores y su posterior comunicación a un cliente cumpliendo con sus necesidades.

En la situación actual, en la que la globalización es más que una realidad y el estancamiento económico es más que un obstáculo para mantener la competitividad empresarial, sólo las empresas que se preparan, disponen de alguna posibilidad de futuro. Hemos de asumir desde un principio que la fuerza competitiva de la PYME[2] española no vendrá dada por la reducción del coste de mano de obra, de materiales, o del resto de costes asociados a producción, sea cual sea el producto o servicio. La mejora ha de ser cualitativa por medio de la innovación, optimizando organización y procesos para facilitar flexibilidad y rapidez en la evolución de los cambios [2].

La búsqueda de competitividad, la reducción de costes, la optimización de procesos y la unificación de criterios son inquietudes existentes en la PYME actual. Las empresas necesitan de herramientas para la gestión de su negocio con el fin de integrar y automatizar todos los procesos. Eso se puede conseguir con la incorporación de las

[1] Tecnologías de la Información y la Comunicación

[2] Pequeña y Mediana Empresa

nuevas tecnologías, que facilitan la centralización de los datos para poder compartirlas y comunicar dentro de su ámbito de negocio. Una de las herramientas que más efecto tiene en este sentido es el ERP[3] [3], consiste en un sistema de gestión que automatiza e integra casi la totalidad de las prácticas de negocio, tanto en el ámbito de operación como de producción dentro de una empresa.

Funcionalmente hablando, los ERPs son sistemas formados por módulos que se adaptan a las necesidades empresariales y que integran los datos formando un todo. Modularidad, adaptabilidad e integridad son los rasgos diferenciales del ERP, que le hacen un sistema particular respecto de cualquier otro software.

## 1.1 Contexto

Infortrex nació en 2008 con el objetivo de ayudar a las empresas a ser más competitivas desde el punto de vista tecnológico. Dispone de amplía experiencia y su conocimiento de las últimas tecnologías permite ofrecer un servicio de alta calidad a los clientes, buscando siempre la solución que mejor se ajuste a sus necesidades.

Desde Infortrex se cree que las relaciones a largo plazo con sus clientes son la base para el éxito de cualquier proyecto. Se trabaja cada día con gran entusiasmo y profesionalidad, con el objetivo de alcanzar la excelencia y, de este modo, ofrecer al cliente un gran retorno a la confianza depositada en la empresa.

[3] Enterprise Resource Planning

Ahora bien, un cliente solicita la colaboración de la empresa donde fue desarrollado este proyecto final de carrera para hacer real un proyecto innovador basado en e-commerce en movilidad [4]. El cliente ha contratado los servicios de la empresa para ser el responsable de las tareas de provisión de unos dispositivos electrónicos; su instalación física en multitud de locales comerciales, el mantenimiento, la reparación y sustitución de los defectuosos o con un comportamiento inesperado de los mismos, dichos dispositivos se irán solicitando eventualmente según las necesidades del cliente. Estos dispositivos son diseñados previamente por Infortrex y a lo largo de la memoria no se entra en detalle de su funcionalidad ya que se escapa de la finalidad de este proyecto. Por todo esto se debe diseñar una solución que permita poder controlar todas y cada una de las instalaciones.

La empresa ya dispone de una importante red de proveedores activa, con cobertura a nivel nacional, que hace posible la instalación física de los dispositivos electrónicos antes mencionados. Por el momento el reducido volumen de estas instalaciones hace posible que la tramitación con nuestros proveedores, la comunicación de los trabajos realizados con el cliente así como la gestión por parte de la empresa del sistema completo se esté realizando de forma poco eficiente y con un gasto importante de recursos de la empresa. En la actualidad, la empresa no dispone en producción de una herramienta empresarial capaz de controlar stock de dispositivos, de distribuir los pedidos, revisar las instalaciones, monitorizar los tiempos de actuación ni de crear los albaranes de los pedidos de instalación para enviar a los proveedores entre otras tareas.

El proyecto nace desde la necesidad de cumplir de modo notable con el compromiso adquirido con el cliente, pasando por ofrecer un escenario de calidad y estable de trabajo a los proveedores y en gran medida a mejorar los procesos de gestión empresarial de la propia empresa.

Las necesidades de la empresa y del cliente hacen que se deba plantear una eficaz solución y que suponga el mínimo coste debido a una importante limitación en recursos, pero no sin ello escatimar en funcionalidad, debido a la posible ampliación a nuevos

módulos de gestión como por ejemplo procesos de facturación, control de stock entre otros o evaluación de proveedores.

## 1.2 Motivación

Desde su creación la empresa ha tomado como estrategia, en la medida de lo posible, la evaluación y el empleo de código abierto [5] en sus proyectos por varias razones, una de ellas inevitablemente es el coste que supone la adquisición de licencias de productos propietarios y otra es la rapidez en la puesta en marcha de este tipo de soluciones. Una de estas apuestas firmes ha sido en lo referente al ERP, debido a la necesidad más que urgente por tomar una medida respecto a la tarea encargada por nuestro cliente de gestionar todas las instalaciones de los dispositivos electrónicos. Por ello y por el interés de la empresa de disponer en plantilla de personal con conocimientos en el desarrollo de módulos ERP basada en una plataforma de código abierto ha hecho posible que este proyecto se lleve a cabo.

Por otro lado la empresa se ha visto obligada a pensar que una herramienta de gestión empresarial empieza a ser necesaria para emplearla en procesos internos, y se espera que la experiencia que adquiera con este proyecto final de carrera sirva como impulso para emplear varios módulos adicionales y unificarlos diseñando un software imprescindible para la gestión de varios de los departamentos.

Para la tarea de registro de las instalaciones por parte de los técnicos instaladores, la empresa optó por desarrollar una solución capaz de informar en el menor tiempo posible la finalización de la misma, lo que influye de manera sustancial en el tiempo de respuesta hacia el cliente. Adicionalmente a este detalle, los dispositivos electrónicos que instalan los proveedores disponen en su interior por necesidades del cliente de un módulo

Bluetooth [6], por lo tanto el instalador deberá utilizar, mientras desarrolla su tarea, de un teléfono móvil para poder comprobar que el dispositivo instalado funciona correctamente y la empresa opta por aprovechar este aspecto para el registro del trabajo. Por lo tanto la solución que la empresa considera como idónea es la de implementar una aplicación para smartphones basada en el sistema operativo Android [7] ya que es el sistema operativo con un mayor índice de inserción en el mercado y por su facilidad a la hora de desarrollar aplicaciones.

De este modo el proyecto se concibe como el diseño y el desarrollo de un módulo ERP basado en código abierto que solucione las comunicaciones que intervienen en la petición de instalaciones y su posterior confirmación con el cliente, la gestión de los pedidos, la distribución de tareas, integrado con una aplicación Android encargada de registrar los trabajos de los proveedores para garantizar el menor tiempo de respuesta posible.

## 1.3 Objetivos

Los objetivos establecidos en el momento de iniciar de este proyecto fueron los siguientes, clasificados por tipologías:

**Objetivos principales**

- Diseñar e implementar un módulo ERP básico basado en Open Source, capaz de soportar las funcionalidades requeridas por el cliente, con la posibilidad de ampliar a otras nuevas en un futuro.

- Diseñar e implementar una aplicación Android, capaz de registrar las instalaciones por nuestra red de proveedores de un modo sencillo e intuitivo soportando el registrar instalaciones incluso en áreas fuera de cobertura.

- Ofrecer una capa capaz de integrar ambas herramientas y que conformen un único sistema completo.

**Objetivos derivados**

- Ofrecer a la empresa una base a la posible futura herramienta empresarial, otorgando sencillez, modularidad y robustez con un coste reducido.

- Adquirir conocimientos en la creación de módulos ERP bajo una herramienta de Open Source y adaptación de los ya existentes.

- Adquirir nociones en la creación de aplicaciones Android para poder ofrecer soluciones en futuros proyectos.

**Objetivos de formación**

- Complementar la formación teórica del universitario con la experiencia en la empresa.

- Contactar con la realidad empresarial trabajando a diario en un ambiente idóneo para ello.

- Potenciar las habilidades directivas dirigidas a fomentar la comunicación, las relaciones interpersonales, la capacidad de adaptación y el trabajo en equipo.

## 1.4 Contenido de la memoria

La memoria se estructura en capítulos con el siguiente contenido:

En el capítulo 2 “Tryton: Módulo ERP para la gestión de instalaciones” se analiza en una primera instancia dos alternativas de sistemas ERP integrados de código abierto, OpenERP [8] y Tryton [9]. Se desglosa las ventajas e inconvenientes de emplear cada uno de ellos, para posteriormente explicar la decisión tomada junto a las características de la arquitectura, unas nociones de cómo desarrollar módulos en Tryton, y más en detalle el proceso seguido para obtener el módulo requerido por la empresa desde el diseño hasta la implementación de los modelos y las vistas.

En el capítulo 3 “Aplicación Android para instaladores” se describe un resumen de las principales características del entorno de desarrollo de Android así como los componentes básicos, para más adelante detallar el proceso de diseño y la implementación de la aplicación tanto de los Activities [10] como de los Layouts [11]. Una especial mención a la lógica de back-end [12] de la aplicación que también entra en este capítulo y que empleará Servicios REST[4] [13] para resolverlo.

[4] Representational State Transfer

En el capítulo 4 “Sistema Completo” se realiza una descripción de la interacción entre cada una de las entidades que intervienen en el sistema completo, de modo que se ofrezca una imagen clara de cómo trabaja toda la plataforma, tras conocer en detalle todas las partes desarrolladas en los Capítulos 2 y 3.

Durante el capítulo 5 “Conclusiones” se comenta las conclusiones obtenidas durante la realización del proyecto, problemas encontrados y la consecución de los objetivos planteados al inicio del proyecto.

En el capítulo 6 “Bibliografía y enlaces de interés” se listarán todas las referencias de utilidad empleadas para resolver dudas, obtener información y consultar algún aspecto importante necesario para entender el entorno del proyecto.

En el Capítulo 7, “Presupuesto y Tiempo dedicado” se cubre la parte económica del proyecto académico, es decir, el coste del ingeniero que realiza el proyecto. Se incluye además, los costes indirectos y de materiales utilizados para llevar a cabo el proyecto. De igual modo se incluye el coste que supondría a la empresa la entrada a producción de la plataforma completa y un detalle de la planificación empleada para la ejecución del proyecto.

De un modo aparte tratamos tres anexos en los que dos de ellos ofrecen un manual de funcionamiento del módulo Tryton y de la aplicación Android y el tercero consiste en una descripción de la prueba de concepto empleada para la presentación del proyecto.

# 2. Tryton: Módulo ERP para gestión de instalaciones

Para que una empresa pueda ser eficiente en sus procesos productivos, necesita ser consciente de todos sus recursos empresariales y gestionarlos de manera eficaz. Sin embargo, no todas las empresas son capaces de alcanzar esa eficiencia ya sea por la complejidad corporativa o de una forma más habitual por la inversión necesaria para llevar este control [14].

En los casos en los que una empresa requiere tener controlados todos los recursos, es necesario recurrir a algún sistema automatizado de gestión y administración empresarial. Actualmente estos sistemas se llaman ERP, y se especializan en manejar todo el conjunto de datos que son relevantes para la continuidad de la empresa [15].

Las soluciones ERP están orientadas a facilitar la integración de los sistemas de las empresas, asegurar la comunicación y mejorar su productividad industrial o la eficiencia de la gestión financiera, gestionar ventas, la atención al cliente, el comercio electrónico o e-commerce, el inventario y las operaciones, la producción, la planificación, e incluso los reportes…. Un ERP por definición se estructura en varios módulos, los cuales se pueden y deben enlazar entre ellos. Es habitual en una primera instancia que en estas soluciones, por defecto, se encuentren compuestas por algunos módulos funcionales habituales en las empresas, a continuación se listan a modo de resumen los más destacados:

- Finanzas: encargado del almacenamiento de cada transacción y su impacto administrativo, de modo que facilita en gran medida la labor de auditores.

- Gestión de producción: módulo encargado del movimiento de artículos, gestión de materiales o la planificación de órdenes de fabricación.

- Inventario y Logística: con el que se controla habitualmente el stock, los almacenes y los posibles flujos de entrada y salida de productos.

- Recursos Humanos: responsable de la gestión de personal, aplicación de la normativa legal, productividad, incentivos y por supuesto las nóminas.

Es evidente que ni todas las empresas necesitan de todos estos módulos ni que con ellos se resuelve el problema completo de la gestión de sus procesos empresariales, por ello se complementan con módulos adicionales e incluso se limita la funcionalidad de los anteriores. De esta forma, una única herramienta puede prestar servicio a empresas de diferentes ámbitos ya que con el simple hecho de habilitar o deshabilitar los módulos y modificar las relaciones entre ellos ofrecen una funcionalidad a medida para cada entorno, este es el esqueleto de un ERP.

La instalación y la configuración de una solución de este tipo en una empresa, requiere de un conocimiento muy elevado sobre sus procesos internos, ya que el detalle en el que se realiza la implantación es a un bajo nivel. Para aclarar un poco este último asunto cabe destacar una de las características básicas de estas herramientas, la adaptabilidad a roles. No todos los empleados de una empresa deberán disponer de los mismos permisos ni de la misma visibilidad de los diferentes módulos, por ejemplo un técnico de una empresa no necesita ni es aconsejable que disponga de acceso al módulo de recursos humanos muy probablemente y de igual modo un administrativo encargado de gestionar nóminas es lógico que solamente disponga de visibilidad de algunas partes de los módulos de recursos humanos y de contabilidad. Por lo tanto en función de las

necesidades de la empresa una herramienta ERP puede resultar tremendamente compleja, debido a los módulos personalizados a los diferentes roles y usuarios.

El coste temporal y económico empleado en la implantación de la solución se ve compensando rápidamente, ya que se dispondrá de una forma de observar íntegramente a la empresa, se podrá:

- Detectar de una manera objetiva los puntos débiles en la gestión.
- Optimizar los procesos existentes
- Centralizar datos actualizados y coherentes con los procesos internos instantáneamente.
- Acceder a toda la información de la empresa basándose en roles y de forma modular
- Compartir información entre grupos de trabajo

Es importante destacar que la implantación de la herramienta ERP no es ni mucho menos trivial, de hecho es habitual que frente al coste de una solución a medida muchas empresas opten por adaptar sus procesos a las configuraciones predeterminadas de la propia solución, por lo que los cambios que son necesarios afrontar en la forma de trabajar que los empleados tienen interiorizados debe ser analizados debido a que esto puede resultar en que la implantación sea un fracaso. Una conclusión de estas características debe evitarse, con políticas transparentes y actividades de concienciación

ya que la concepción de la información y su tratamiento es un proceso crítico para cualquier empresa [16].

Los ERP de código abierto han encontrado su lugar en el mercado del software empresarial debido a la madurez y la experiencia de los desarrolladores y empresarios, a la enorme flexibilidad que ofrecen y a la posibilidad de implementación a precios mucho más asequibles que las soluciones propietarias. Las tres ventajas principales del Open Source son: el precio, como ya se ha indicado anteriormente; el acceso al código y el acceso a la comunidad que se crea en torno al producto de forma que es posible compartir conocimientos e incluso que el código sea revisado por muchas personas para mejorar su funcionalidad [17].

Existen una multitud de soluciones ERP Open Source, a continuación se revisan las opciones estudiadas:

- OpenERP es una solución que integra funciones de ventas, CRM[5], gestión de proyectos, gestión de almacenes, gestión financiera y recursos humanos entre otras. Están disponibles más de 700 módulos adicionales oficiales. Un detalle importante es que permite el uso remoto mediante un cliente con una interfaz web.

- Tryton, aunque surgió como un fork[6] de OpenERP, a día de hoy hay pocas similitudes a nivel interno, ya que se ha depurado y limpiado el código de OpenERP para ser más escalable y fiable.

[5] Customer Relationship Management
[6] Creación de un proyecto en una dirección distinta a la principal tomando el código fuente del proyecto ya existente

## 2.1 Estudio comparativo: OpenERP vs Tryton

La elección tomada por la empresa en un primer momento es la de trabajar con OpenERP, a continuación se analiza diferencias y similitudes entre está opción y la solución tomada finalmente, Tryton.

La razón por la que ambas plataformas emplean, como lenguaje de programación, Python es la de que se favorece la productividad del desarrollador, es aceptado por la mayoría de los sistemas operativos, es fácilmente legible y puede aprenderse en poco tiempo.

OpenERP de la empresa Tiny sprl. anteriormente fue conocido como Tiny ERP, disponía de una base tecnológica buena, debido a que proveía de una estructura completa para un ERP: una capa mapeo objeto-relacional (ORM[7]), servicios de red e incluso la aplicación cliente y mucho más importante aún, la plataforma ofrece gran simplicidad a la hora de desarrollar y su modularidad.

Como ya se ha mencionada con anterioridad Tryton surge como un fork del proyecto de código abierto comercial OpenERP, que comezó a ser desarrollado a finales de 2008. La razón detrás de Tryton no es la de ser un competidor directo de OpenERP, sino más bien de proveer una nueva vía de afrontar el problema de la gestión de recursos empresariales. Busca ofrecer una solución sólida y consistente frente a una funcionalidad avanzada.

[7] Object-Relational mapping

Tryton ha evolucionado en estos años del siguiente modo:

- Se han eliminado decenas de miles de líneas de código y se han añadido más de 300.000 líneas de código.

- Los módulos ya existentes en Tryton se han reescrito al completo, debido a que la mayoría de los módulos fundamentales ya escritos fueron creados cuando la mayoría de las características avanzadas aún no existían. De este modo se obtiene una mejor armonización entre los módulos base, una modularidad optimizada y una plataforma más poderosa para futuros desarrollos a medida.

A continuación, Fig. 1.1 se lista los módulos básicos y oficiales disponibles para ambas alternativas:

| Área | Tryton | OpenERP |
|---|---|---|
| **Contabilidad y administración de finanzas** | Sí | Sí |
| **Sistema de gestión de documentos** | Sí (límite de tamaño de ficheros: determinado por el sistema de ficheros) | Sí (límite de tamaño de ficheros: determinado por el sistema de archivos o por PostreSQL(1GB)) |
| **Contabilidad analítica** | Sí | Sí |
| **Gestión de nóminas** | No | Sí |
| **Portales** | Sí | Sí |

| | | |
|---|---|---|
| **Gestión de ventas** | Sí | Sí |
| **Gestión de almacenes** | Sí | Sí |
| **Gestión de proyectos** | Sí | Sí |
| **Gestión de compras** | Sí | Sí |
| **Gestión de manufactura** | Sí | Sí |
| **Recursos humanos** | Sí | Si |
| **CRM** | Básico | Sí |
| **Punto de ventas** | Sí | Sí |
| **VoIP** | Soporte para SIP[8] y widgets "Llamar a". | No hay módulos oficiales. |

*Figura 1.1. Comparativa módulos oficiales Tryton/ OpenERP*

Adicionalmente a este listado es importante el siguiente dato, OpenERP dispone de más de 250 módulos oficiales adicionales, mientras que Tryton ofrece algo más de 70 módulos. La diferencia en la cantidad de módulos viene acompañada por la cantidad de colaboradores activos (822 de OpenERP frente a 42 de Tryton).

A parte de estas diferencias conviene destacar algunas otras, que a continuación se enumeran:

[8] Session Initiation Protocol

- OpenERP únicamente soporta PostgreSQL [18] como base de datos, mientras que Tryton permite el uso tanto de PostgreSQL, MySQL [19] como de SQLite [20].

- Ambas versiones ofrecen al usuario un cliente de escritorio, pero tan solo OpenERP actualmente dispone una versión Web y un cliente móvil, este es un punto diferenciador importante ya que es un aspecto que interesa de un modo destacado a la empresa.

- Ambos sistemas ofrecen calendario para los módulos, pero la diferencia entre ambas es importante. OpenERP dispone de un widget dentro de todos los clientes capaz de mostrar vista de calendario mientras que Tryton es capaz únicamente de mostrarlo empleando CalDAV, WebDAV, por lo tanto no viene incorporado en el interior de las propias vistas del cliente.

- OpenERP es capaz de mostrar vista de Diagramas Gantt en sus clientes web mientras que Tryton no lo es.

- El gran número de colaboradores y de recursos de OpenERP es uno de los factores que provoque que en términos de funcionalidad sea mucho más amplio, este dato se ve reflejado claramente en el número de traducciones de sus módulos, mientras que Tryton dispone de traducciones a 8 lenguajes, OpenERP dispone de equipos de trabajo asignados a 30 lenguajes.

- Tryton tiene soporte para IPv6.

- Tryton facilita en términos de costes la migración de versiones de los módulos, mientras que OpenERP deben ser previamente contratadas.

Teniendo en cuenta todos estos aspectos y analizando las necesidades actuales y futuras a corto plazo de la empresa, se llega a la conclusión de que para esta prueba de concepto lo más interesante es emplear Tryton. Para esta decisión ha sido importante especialmente la existencia de un módulo oficial que OpenERP no ofrece y que es de un gran interés para la empresa, pero que en un primer momento no tiene importancia para este proyecto.

De igual modo también es importante destacar que un cambio de política futuro con una posible migración a OpenERP tampoco supondría un coste elevado ya que la arquitectura es prácticamente la misma y no acarrearía excesivos cambios para ello.

## 2.2 Entorno Tryton

### *2.2.1 Introducción a Tryton*

Tryton es una plataforma informática de alto nivel de propósito general de tres capas escrita en Python [21] y que emplea PostgreSQL como motor de base de datos.

El núcleo de Tryton ofrece todas las funcionalidades necesarias de una plataforma de aplicaciones completa: persistencia de datos, amplia modularidad, administración de usuarios, workflow y motor de reportes, servicios web e internalización. Se constituye por tanto en una plataforma de aplicación completa que permite ser empleada para cualquier propósito relevante.

En la actualidad los módulos disponibles en la suite de Tryton cubren los siguientes campos: contabilidad, facturación, administración de ventas, administración de compras, contabilidad analítica y administración de inventario; pero además establece una base sencilla capaz de abstraer conceptos clave para cualquier adaptación de negocio.

### *2.2.2 Plataforma de tres capas*

Una arquitectura basada en tres capas se encuentra estructurada típicamente de la siguiente forma:

- Capa de presentación, aquella capa que ve el usuario, presenta la aplicación al usuario, le muestra la información y captura los datos a tratar por la aplicación. Esta capa se comunica exclusivamente con la denominada capa de negocio.

- Capa de negocio o aplicativa, donde se alojan los programas con la lógica de negocio que se ejecutan, recibe peticiones y envía respuestas tras la ejecución de estos programas, está capa se comunica tanto con la capa de presentación al recibir solicitudes, como con el gestor de base de datos para solicitar interacciones de entrada y obtención de información.

- Capa de datos o de base de datos, lugar donde residen los datos y es la encargada de acceder a los mismos. Como ya se ha mencionado reciben solicitudes de almacenamiento y recuperación de información por parte de la capa de negocio.

### *2.2.3 Modelos y Vistas*

En este apartado se presenta una breve descripción de la forma de declarar los elementos de los módulos y se muestra algunas opciones que Tryton otorga, no con el fin de ofrecer un manual de referencia sino para explicar algunos aspectos básicos para poder entender el desarrollo del proyecto. Para ello se entrará en detalle de cómo se definen los modelos y las vistas.

**Modelos**

Para Tryton un modelo representa una única lógica de negocio o un concepto. Un modelo está compuesto por campos y define el comportamiento de sus registros. Es habitual que cada registro de un modelo se encuentre almacenado en una tabla en la base de datos.

Características de los modelos:

- Cada modelo es una clase Python que hereda de la clase trytond.model.model.Model
- Los campos son definidos como atributos de los modelos.
- Tryton genera automáticamente la definición de las tablas.
- Tryton provee un API para acceder a los registros de los modelos.

A continuación se expone un ejemplo sencillo de creación de un modelo, de esta forma se instancia el modelo en el servidor, en la base de datos y es almacenada dentro del Pool de Tryton, espacio en el que se almacenan modelos, asistentes y reportes.

```
from trytond.model import ModelView, ModelSQL, fields

class Persona(ModelSQL, ModelView):
   _description = __doc__
   _name = "persona.persona"
   nombre = fields.Char('Nombre')
   apellidos = fields.Char('Apellidos')

Persona()
```

En el ejemplo puede observarse un modelo de tipo “Persona” que presenta dos campos; nombre y apellidos, en este caso de tipo carácter. También se puede ver los atributos que comienzan con guion bajo, que representan propiedades propias de la clase.

*Campos*

Los campos definen el comportamiento de los datos de los registros del modelo. Existen diferentes opciones a la hora de definirlos, a continuación se detallarán algunos de los empleados en este proyecto tanto su definición como su comportamiento. Algunas de estas opciones son comunes para todos los tipos de campos existentes.

- *Field.string*: define una cadena para la etiqueta del campo.

- *Field.help*: registra una ayuda de varias líneas para el campo.

- *Field.required*: determina si es obligatorio que el campo contenga contenido si su valor es “True”. Por defecto su valor es “False”.

- *Field.readonly*: determina si es modificable el contenido del campo si su valor es “False”. Por defecto su valor es “True”.

- *Field.domain*: establece que una restricción es aplicada al contenido del campo. Actualmente solo es aplicable a campos de tipo *Many2One*, *One2Many* y *Many2Many*.

- *Field.on_change*: establece un listado de nombres de campos, de forma que si cualquiera de ellos es modificado se invoca al método *on_change* definido del campo que corresponda.

Ya se ha podido observar que existe el tipo de campo *Char*, pero existen algunos otros que se detallan a continuación:

- *Boolean:* campo cuyo contenido es “True”/”False”

```
class trytond.model.fields.Boolean(string[,**options])
```

- *Integer*: campo cuyo contenido es un número entero

```
class trytond.model.fields.Integer(string[,**options])
```

- *BigInteger*: campo cuyo contenido es un entero largo

```
class trytond.model.fields.BigInteger(string[,**options])
```

- *Char*: campo cuyo contenido es una línea de caracteres

```
class trytond.model.fields.Char(string[,size[,translate[,**options]]])
```

Este tipo de campo dispone de algunas opciones especiales que se pueden aplicar, por ejemplo para establecer el tamaño máximo del campo o de si se debe traducir en función del lenguaje del sistema.

- *Sha*: campo cuyo contenido debe ser almacenado codificado con algún algoritmo de seguridad de la familia SHA[9] [22].

```
class trytond.model.fields.Sha(string[,**options])
```

[9] Secure Hash Algorithm

- *Text*: define un tipo de campo en el que su contenido es un texto de varias lineas.

```
class trytond.model.fields.Text(string[,size[,translate[,**options]]])
```

Este tipo de campo dispone de las mismas opciones existentes para el tipo *Char*.

- *Float*: campo con contenido de tipo coma flotante, será representado en Python como una instancia de float.

```
class trytond.model.fields.Float(string[,digits [,**options]])
```

Dispone de una opción adicional con la que se permite especificar el número de dígitos tanto de la parte entera como de la parte decimal.

- *Numeric*: campo similar a *Float* pero con un número fijo de decimales, es representado en Python como una instancia de decimal y dispone de la opción adicional explicada previamente para el tipo *Float*.

```
class trytond.model.fields.Numeric(string[,digits [,**options]])
```

- *Date,DateTime y Time*:campo en el que se almacena una fecha, fecha y hora u hora simplemente para ellos se utiliza los tipos de datos de Python date, datetime y time respectivamente.

```
class trytond.model.fields.(Date|DateTime|Time)(string[,**options])
```

- *Binary*: permite almacenar datos binarios empleando para ello la instancia str de Python, es especialmente útil para almacenar ficheros.

```
class trytond.model.fields.Binary(string[,**options])
```

- *Selection*: presenta un campo cuyo contenido queda restringido a una selección. Por ello dispone de un argumento obligatorio, selection, con las tuplas de los valores disponibles.

```
class trytond.model.fields.Selection
(selection,string[,sort[,translate[,**options]]])
```

Adicionalmente dispone de una opción para establecer el orden en el que aparecen los diferentes valores de la selección y otra para definir si se debe traducir su valor al lenguaje que corresponda si es posible.

- *Many2One, One2Many, Many2Many, One2One*: permite establecer las dependencias típicas de la lógica en base de datos. Para declarar un campo de tipo *One2Many* es obligatorio disponer de un opuesto *Many2One* en el modelo objetivo.

Las opciones que Tryton nos ofrece son difícilmente abarcables en este documento, pero estas nociones son suficientes para comprender la parte de modelaje del proyecto.

## Vistas

Otro aspecto a destacar en la estructura de los módulos de Tryton es el de las vistas. Gracias a ellas se especifica cómo se muestran al usuario los registros de nuestros modelos. Las instancias de los modelos pueden ser mostrados de diferentes maneras. El flujo es el siguiente, una acción abre una ventana y define que vista mostrar.

Estas vistas son construidas en XML[10] [23] y son almacenadas en la base de datos bajo la tabla de vistas ir.ui.view.

Existen tres clases de vistas: de tipo formulario, árbol y gráfico. Para el proyecto únicamente se emplean formularios y arboles por lo que no entraremos en detalle de cómo se utilizan las vistas de tipo gráfico, aunque si se explicará su utilidad.

[10] eXensible Markup Language

*Formularios*

Un tipo de vista formulario se emplea para representar un registro de un objeto. Los diferentes elementos de la vista se colocan en la pantalla de la siguiente forma:

- Los elementos son colocados de izquierda a derecha y desde arriba a abajo acorde al orden establecido en la definición de la vista siguiendo el patrón de una tabla.

- La pantalla se compone de un número fijo de columnas y con las filas necesarias para mostrar todos los elementos.

- Los diferentes elementos pueden necesitar de una o varias columnas al ser posicionadas en la tabla. En caso de que no fuera posible ubicarla en la fila correspondiente será emplazado al principio de la siguiente fila.

Cada vista de formulario debe comenzar por el tag <form> y puede tener varios atributos, en la definición, por ejemplo el atributo *string* empleado por defecto en la pestaña del formulario dentro de la ventana o el atributo *col* que define el número de columnas de la vista.

Cada campo en el formulario puede mostrar una etiqueta para describir de una forma breve su contenido, para ello se emplea el tag <label> donde su atributo *string* especifica el texto que se mostrará en la etiqueta.

Como elemento principal de los formularios se encuentra el tag <field>, gracias al cual se indica el contenedor donde se mostrarán o se introducirán los datos que configuran un

objeto de un modelo determinado. Como atributos para este elemento están entre otros el atributo *name* con el que se hace referencia al campo del modelo que estemos referenciando, widget para establecer el tipo de marco que se empleará en la vista, otros empleados para especificar el ancho y el alto del contenedor como *width*, *height*, *yexpand*, *xfill*, *yexpand*, *yfill* o *colspan* son utilizados habitualmente.

También se puede mostrar en el formulario una imagen, únicamente especificandolo en el XML del formulario con el tag <image>, en el que también están disponibles atributos de tamaño y colocación de la propia imagen que mostrar.

Para trabajar con la colocación de los diferentes elementos se emplea el tag <newline>, lo que permite bajar a la siguiente fila de la tabla y empezar a ubicar componentes en la nueva ubicación.

También es muy común establecer una separación lógica entre elementos superiores e inferiores, para ello se utilizad el tag <separator>  ya que crea un separador horizontal visible en la pantalla.

Estos elementos descritos son los principales que se han empleado en la implementación del proyecto pero se disponen de otros muchos para dar una mayor complejidad a las vistas como puede ser trabajar con botones, añadir un widget de paginado y trabajar con varias pestañas o crear subgrupos.

A continuación se muestra un ejemplo sencillo en el que se puede observar cómo se estructura una vista de tipo formulario trabajando con un modelo sencillo:

```
   <form string="Persona" col="6">
       <label name="nombre"/>
       <field name="nombre" xexpand="1"/>
       <label name="apellidos"/>
       <field name="apellidos"/>
       <label name="active"/>
       <field name="active" xexpand="0" width="100"/>
       <notebook colspan="6">
           <page string="Datos personales">
               <field     name="direccion"     mode="form,tree"     colspan="4"
view_ids="persona.direccion_view_form, persona.direccion_view_tree_sequence"/>
               <label name="tipo"/>
               <field name="tipo" widget="selection"/>
               <label name="idioma"/>
               <field name="idioma" widget="selection"/>
               <label name="direccion web"/>
               <field name="direccionweb" widget="url"/>
               <separator string="categorias" colspan="4"/>
               <field name="categoria" colspan="4"/>
           </page>
           <page string="Datos fisiologicos">
               <label name="peso"/>
               <field name="peso"/>
               <label name="altura"/>
               <field name="altura"/>
           </page>
       </notebook>
   </form>
```

En este ejemplo se observa la definición de una vista de tipo formulario compuesto a grandes rasgos de un componente de tipo notebook con dos pestañas, una para mostrar datos generales y otra con datos de altura y peso. Se puede observar cómo se introduce un separador entre el campo *direccionweb* y el campo *categoria*. Por último es importante destacar la definición antes de los campos de sus respectivas etiquetas para facilitar la labor del usuario y detallar que dato se muestra en la vista.

*Árbol*

El tipo de vista árbol se emplea para mostrar los objetos almacenados en forma de árbol o lista. Habitualmente será mostrado como una lista salvo en el caso de que existan campos hijos que será mostrado como un árbol.

De forma similar a como se especificó previamente en las vistas de tipo formulario, en este caso, las columnas también se colocan de izquierda a derecha en la pantalla según se hayan declarado en el fichero XML.

Cada vista de tipo árbol debe comenzar con el tag <tree> y como atributos puede contener una serie de valores para configurarlo. El atributo *string* es usado para determinar el título de la vista dentro de la pantalla o la vista en la que se encuentre insertado.

El atributo editable nos permite configurar si deseamos que los registros mostrados en dicha vista sean editables y en el caso de que así fuera nos permite también decidir si los registros modificados los ubica en la parte superior o en la inferior del listado.

Para reflejar los campos que se desean mostrar en la vista se emplea el tag <field> de forma similar a como se realizaba para la vista de tipo formulario. En este caso las posibilidades de configuración son menores, como importantes cabe destacar *name* para indicar el campo que se pretende mostrar, *readonly* para indicar si es editable, *widget* para mostrar el marco que interese para tal campo en lugar del establecido por defecto, u otros de edición del tamaño de la columna como por ejemplo *width* para fijar el ancho o *expand* para fijar el ancho en función del contenido de la columna.

A continuación se muestra un ejemplo sencillo en el que se puede observar cómo se estructura una vista de tipo árbol para un modelo básico:

```
<tree string="Asignaturas " sequence="id">
       <field name="nombre"/>
       <field name="curso"/>
       <field name="departamento"/>
       <field name="tipo"/>
       <field name="id" tree_invisible="1"/>
</tree>
```

En este código se observa cómo se crea una vista en la que se listan una serie de asignaturas cuyas columnas mostradas son el nombre, el curso, el departamento al que pertenece y el tipo. Es importante notar que el listado es ordenado por el id de asignatura y que este dato no es mostrado en la vista.

*Gráfico*

Otra vista disponible en Tryton para ser empleada en las vistas es la de insertar en la pantalla gráficos a partir de datos de diferentes registros.

Para ello basta con nombrar dicha vista comenzando su definición con el tag <graph>, dentro se puede especificar algunas características generales del gráfico como pueden ser el tipo de gráfico con el atributo *type*, el nombre del gráfico con *string*, los colores con los atributos *background* o color, así como si deseamos que figure una leyenda con el atributo *legend*.

Para introducir los datos se emplea el tag <x> y el <y> acompañado en su interior por <field> para especificar los campos a representar en la tabla.

A modo de ejemplo sencillo se adjunta a continuación la definición de una vista de tipo gráfico de barras en donde se muestra la evolución de las ganancias de una empresa a lo largo de los años.

```
<graph string="Beneficios anuales" type="vbar">
  <x>
      <field name="Anyo"/>
  </x>
  <y>
      <field name="beneficios"/>
  </y>
</graph>
```

## 2.3 Diseño de módulo ERP

Una vez establecida la base de conocimientos necesarios, a continuación se describen los requerimientos impuestos por la empresa y sus expectativas, para la parte del módulo ERP a desarrollar y que constituyen los objetivos que no pueden ser modificados a lo largo de la ejecución del proyecto y que deben ser satisfechos.

Una vez que se dispongan de los requisitos se pasa a detallar el proceso de diseño de los modelos para conformar la estructura de nuestro módulo al igual que las vistas para hacer visible toda la información requerida.

### *2.3.1 Requisitos*

Para esta prueba de concepto se requieren desarrollar las siguientes entidades con la siguiente información cada una de ellas:

**Empresa**

Se requiero almacenar en el módulo a desarrollar todas las empresas proveedoras de servicios, así como modificar y eliminar las no deseadas de modo que se disponga de las siguientes características de cada una de ellas:

- Razón Social y CIF
- Logo de la empresa
- Dirección de la empresa
- Teléfono y correo electrónico de contacto de la empresa
- Descripción del área geográfica de cobertura para las instalaciones
- Usuario y contraseña único por empresa para autenticarse en la aplicación Android

- Posibilidad de habilitar o deshabilitar esta empresa como usuario de la aplicación móvil.

**Responsable**

Se requiere almacenar en el módulo a desarrollar todas las personas responsables de las diferentes empresas ya guardadas en base de datos, así como modificar y eliminar las no deseadas de modo que se disponga de las siguientes características de cada una de ellas:

- Nombre y apellidos
- NIF
- Teléfono y correo electrónico de contacto
- Cargo en la empresa
- Sexo

**Instalador**

Se requiere almacenar en el módulo todos los instaladores disponibles y con capacidad de realizar las instalaciones que se requieran, así como modificar y eliminar los

necesarios de modo que se disponga de las siguientes características de cada uno de ellos:

- Nombre y apellidos
- NIF
- Foto
- Teléfono y correo electrónico de contacto
- Sexo
- Descripción del área geográfica de cobertura de trabajo
- Preferencia de horario
- Modelo del teléfono móvil con el que realizará las instalaciones y su versión de sistema operativo Android

**Pedido**

Se requiere almacenar en el módulo todos los pedidos que lleguen del cliente, se deberá dar la posibilidad tanto de dar de alta los pedidos de forma automática como

manualmente. Para ello se deberá realizar una revisión de una cuenta de correo a la que llegan los pedidos al menos dos veces diarias y en función del resultado de la revisión crear los nuevos pedidos. De igual modo que para las anteriores entidades solicitadas se debe permitir modificar y eliminar los pedidos que convengan. La información necesaria en cada uno de los pedidos es la siguiente:

- Código Bidi [24] como identificador único de pedido, cuyo contenido se establecerá respetando el siguiente patrón, "*5469656E6461*|*identificador_bluetooth*|*identificador_bidi*|*fecha_de_creación*", donde la fecha de creación debe presentar el formato "YYYYmmdd".

- Tipo de instalación, donde únicamente podrá ser de dos tipos, de un dispositivo Bluetooth solitario o acompañado de una pegatina promocional. A continuación se describe en detalle cada uno de los diferentes elementos posibles a instalar:

    - *Dispositivo Bluetooth*

      En este punto se describe cómo funciona el dispositivo con el que se trabajará. Físicamente su aspecto es una caja rectangular, en cuyo interior se encuentran ubicados dos módulos Bluetooth; el primero es un módulo Bluetooth 2.0 y el segundo es un Bluetooth 4.0 [25], la función de estos módulos no se detallará, únicamente explicar que al enlazarse con estos módulos se obtiene un identificador para cada uno, que se refiere a una dirección MAC[11] [26], únicos para cada módulo que se instalan. En la misma carcasa se encuentra en su superficie una etiqueta con un código Bidi, donde se encuentra codificado una cadena de caracteres con el mismo contenido al que ofrece el Bluetooth 4.0. En la Fig. 2.1. , puede observarse el aspecto de uno de estos dispositivos con los que se trabajan.

[11] Media Access Control

*Figura 2.1. Dispositivo Bluetooth*

- *Pegatina promocional.*

La pegatina que debe ser colocada en el tipo de instalación 2 consiste simplemente en un adhesivo con publicidad de nuestro cliente y un código Bidi cuyo contenido representa, como es habitual en el sistema que se está describiendo, una dirección MAC. Se puede observar una de estas pegatinas en la Fig 2.2.


*Figura 2.2. Pegatina promocional*

Todos los códigos Bidi, con direcciones MAC codificadas, y los módulos Bluetooth; sirven para que las personas que estén interesadas en el producto de nuestro cliente reciban con sus teléfonos estos identificadores de tiendas, demostrando que se encuentran en el interior del local, y puedan interactuar con lo que el producto del cliente ofrezca.

- Número de identificación de la tienda donde se llevará a cabo la instalación

- Nombre de la tienda donde se llevará a cabo la instalación así como el nombre del centro comercial en el que se ubica en el caso de que corresponda.

- Datos de contacto de persona responsables de la tienda

- Cargo de la persona de contacto de la tienda

- Dirección de la tienda

- Número de identificación del Bluetooth que se instalará

- Comentario con detalles sobre la ubicación dentro del local donde se llevará a cabo la instalación del Bluetooth

- Número de identificación de la pegatina promocional

- Comentario con detalles sobre la ubicación dentro del local donde se llevará a cabo la instalación de la pegatina promocional

- Campo con observaciones especiales sobre la instalación a efectuar

- Fecha de creación del pedido en el sistema

- Fecha pactada de la instalación en el local por la empresa

- Estado del pedido, donde siempre deberá especificar alguno de los siguientes valores: finalizado, asignado o pendiente y se debe mantener dicho estado en función de la evolución de forma automática.

**Instalación**

Se requiere almacenar en el módulo todas las instalaciones que hayan sido finalizadas por los proveedores. Para ello se deberán dar de alta las instalaciones automáticamente en el momento en que llegué el aviso del instalador mediante la aplicación móvil. La información necesaria en cada uno de los pedidos es la siguiente:

- Numero secuencial, para identificar cada instalación
- Fecha de instalación
- Información ofrecida por el dispositivo Bluetooth instalado
- Información ubicada en el dispositivo Bluetooth
- Información ofrecida por la pegatina promocional
- Comentarios introducidos por el instalador para ofrecernos más información en el caso en el que se considere oportuno
- Foto de la instalación

Para todas estas entidades definidas se requiere que en los campos en los que su contenido sea bien un CIF/NIF, un número de teléfono o un correo electrónico este sea validado superficialmente, a nivel de desarrollo, antes de ser dado de alta o modificado dicho registro.

Las diferentes vistas del módulo deben ser capaces de enlazarse internamente y visualmente entre las entidades que por su lógica así lo requieran, por ejemplo es importante que entre otras, desde una vista de una empresa, el módulo sea capaz de mostrar todos los responsables pertenecientes a dicha empresa y del mismo modo acceder a ellos y modificarlos.

Es imprescindible que la estética de las diferentes pantallas sea lo más clara posible ofreciendo en la ejecución del proyecto plena libertad en el modo de obtener dicho objetivo.

El código deberá estar debidamente comentado para agilizar futuras actualizaciones y de este modo evitar los problemas que puedan ser causados por un código mal explicado.

### *2.3.2 Descripción de tareas*

Trabajando en base a los requisitos entregados por la empresa y en una serie de tareas que a continuación se detallan, se pretende obtener unos resultados satisfactorios de forma que el módulo quede plenamente funcional.

1. Fase de análisis de requisitos

Basando el futuro módulo en los requisitos recibidos, se analizan y se lleva a cabo una reunión para disipar toda duda respecto a los aspectos que necesiten de un desarrollo más detallado y/o de una aclaración.

En este proyecto realmente no existen unos requisitos realmente estrictos, sino que en la mayor parte de ellos se deja libertad mientras la funcionalidad no se vea alterada.

En cuanto a la funcionalidad, se prevé que a medida se vayan generando los diferentes logros estos se muestren al responsable del proyecto para obtener su validación y continuar con las siguientes tareas.

2. Diseño de los modelos y sus interrelaciones

Durante esta fase el objetivo será el de obtener un diagrama con el modelado completo del módulo, donde se podrán observar las interrelaciones entre modelos. En esta fase también se definen que campos deberán ser obligatorios y cuales opcionales de las entidades ya explicadas. Gracias al modelado relacional que se llevará a cabo nos permitirá visualizar, especificar, construir y documentar el sistema de una forma clara y sencilla de entender para cualquier persona ajena al proyecto.

3. Diseño de las vistas del módulo

El objetivo de esta fase será el de lograr que el módulo sea lo más claro y ordenado con el fin de hacer que las interacciones del usuario sean lo más intuitivas posibles. En este sentido, Tryton nos ofrece una importante ayuda ya que el marco y los diferentes widgets vienen predefinidos, por lo tanto, el objetivo será el de colocar, maquetar, estructurar, definir las vistas y la información que debe figurar en cada una de ellas.

4. Validación de la fase de diseño

En este momento del proyecto dispondremos de la documentación necesaria para implementar la solución adoptada durante las fases de diseño, cuya solución es presentada al jefe del proyecto valorando alguna posible mejora.

En el momento en el que se disponga del visto bueno del responsable se comenzará con la implementación del módulo.

5. Creación del módulo

Durante esta tarea se creará la estructura básica del módulo definiéndose como un nuevo elemento del sistema ERP, incluyendo los ficheros necesarios y definiéndose tanto las dependencias como la información de referencia del módulo.

6. Implementación de los modelos

Con la documentación ofrecida por la comunidad de Tryton, cuyo resumen se ha redactado en una sección anterior en esta memoria, y partiendo del diseño previo se desarrollará el código necesario para hacer real nuestro modelado de datos.

## 7. Implementación de las vistas

En base a las vistas bocetadas en la fase de diseño, se desarrollarán todas ellas siguiendo fielmente lo acordado. En este punto debe figurar también la creación de las hojas de pedido que posteriormente se remitirán a la empresa proveedora asignada, para ello se creará la plantilla con el mismo formato con el que se venía trabajando antes de la ejecución de este proyecto.

## 8. Implementación de la generación automática de pedidos

Se creará un cron escrito en Python para acceder a la bandeja de la cuenta de correo que se establezca para recibir las notificaciones del cliente con solicitudes de alta de instalaciones. Este cron se debe configurar para que sea ejecutada al menos dos veces al día y en caso de que en la bandeja existan notificaciones nuevas, según el formato esperado, se creará en el sistema un pedido con la información que figure en el correo en cuestión. Adicionalmente a esto se debe crear, dentro de la vista que se dedicará para mostrar cada instancia de pedidos, una plantilla para poder generar un archivo con formato PDF[12] que se remitirá a la empresa que ha sido asignada para la tarea de ejecución.

[12] Portable Document Format

9. Validación del módulo completo

Una vez finalizado todas las tareas se presentan los resultados en una reunión donde se valorará si es necesario realizar alguna modificación o por el contrario todo está implementado según lo esperado. En caso de que la implementación haya sido exitosa se iniciará un periodo de testeo del sistema donde se estudiará la estabilidad del sistema, la sencillez de uso, se probará toda la funcionalidad, para posteriormente dar de alta en la base de datos la información de la que se requiera.

### *2.3.3 Diseño de los modelos*

A lo largo de este apartado se muestra el proceso seguido junto a las medidas tomadas para la obtención de toda la lógica diseñada de los modelos del módulo ERP bajo los condicionantes que impone la arquitectura de Tryton. Se describirá entidad a entidad el modelado total del sistema y finalmente se ofrecerá una perspectiva global del modelo de datos del módulo para la gestión de instalaciones.

**Empresa**

Se crea una clase específica para esta entidad y a esa clase se la denomina “Empresa”, Fig 2.3., bajo el atributo de clase *_name* se especifica este mismo nombre con la nomenclatura necesaria, en este caso con el nombre del módulo seguido del nombre de la clase en minúsculas, “instaladores.empresa”. Esta clase se registra para almacenar toda la información de interés de colaboradores pertenecientes a la red de proveedores. Por cada empresa colaboradora se dispondrá de una instancia de dicha clase.

Dentro de la clase se dispone de los siguientes atributos modelados según los requisitos que la empresa ha indicado:

La razón social es un campo bajo el atributo denominado “name”, es de tipo *Char* cuyo contenido no está limitado en longitud pero exclusivamente deberá tener una línea, aparte de esto deberá ser obligatorio que presente contenido para cada instancia de la clase ya que no tiene ningún sentido la existencia de una empresa sin este atributo.

El CIF al igual que la razón social es un atributo denominado “cif”, de tipo *Char* y no es necesario que el campo tenga contenido ya que es posible que se necesite crear una empresa y no se disponga de este dato. La longitud debe ser de 9 caracteres en el caso de que el campo tenga información y se realiza una validación para comprobar si la cadena introducida cumple la regla de estos identificadores.

Para modelar la información referente a la dirección de la empresa se opta por crear cuatro campos que se definen de tipo *Char*, no obligatorios y con el contenido no limitado en longitud. Estos atributos se son nombrados como “direccion”, “codPostal”, “provincia” y “localidad”, de este modo se dispone de la ubicación de cada una de las empresas proveedoras.

En referencia a los datos de contacto de la empresa se dispone únicamente de dos campos. En el primero de ellos “telefono”, se almacena el teléfono de contacto de la empresa, se define de tipo *Char* y su contenido queda limitado a 9 caracteres numéricos, para este atributo se debe realizar una validación ya que no es posible que su contenido empiece por una cifra diferente a 6,8 o 9. En el segundo se recoge un correo electrónico para contactar con la empresa vía email, este campo se denomina “email”, se define de tipo *Char*, con longitud ilimitada y es validado según el patrón de definición de una cuenta de correo electrónico.

También se requiere almacenar el logo de la empresa, para este requerimiento Tryton ofrece la posibilidad de almacenar en un campo, datos de tipo binario. Por esto se crea un atributo denominado “logo” de tipo *Binary*.

Para poder disponer de una autentificación de los instaladores que accedan a la aplicación Android, cada empresa debe poseer una tupla usuario y contraseña bajo las cuales se debe logar. Este usuario debe pertenecer a la clase *Empresa*, ya que se requiere que el usuario sea compartido por todos los instaladores de la misma empresa. De esta forma se definen dos campos obligatorios “usuario” y “password” de tipo *Char* y sin limitación alguna en su contenido. Adicionalmente a estos atributos de la clase se añade un atributo para controlar si este usuario se le ofrece permiso para acceder a la aplicación móvil, esto se modela simplemente definiendo el campo borrado de tipo *Boolean*.

Como se define en los requisitos, es importante para la empresa disponer de información sobre qué áreas geográficas es capaz de dar cobertura y bajo qué condiciones, para ello se define un campo nombrado como “descripcionAreaTrabajo” de tipo *Text*, ya que se permite introducir un contenido multilínea, y en él se redactará cualquier información de interés al respecto.


*Figura 2.3. Diagrama de clase Empresa*

Con esto se completa el modelado de los atributos, únicamente falta definir las relaciones con el resto de entidades, para ello se necesita disponer del diseño de algunas de las demás entidades, por ello se aplaza este asunto hasta que sea posible tratarlo.

**Responsable**

Al igual que para la entidad empresa, para esta entidad también se define una clase específica y a esa clase se la denomina “Responsable”, Fig 2.4. , bajo el atributo de clase *_name* se especifica este mismo nombre con la nomenclatura ya comentada, “instaladores.responsable”. Esta clase es creada para almacenar toda la información de interés de las personas responsables de cada empresa. Por ejemplo es muy posible que exista una o más personas encargadas de gestionar a sus trabajadores para este asunto, por esta razón se requiere disponer de esta información para que en caso de necesidad se pueda disponer ella y se facilite la comunicación con estas personas.

Dentro de la clase se dispone de los siguientes atributos modelados según los requisitos listados anteriormente en la memoria:

Se solicita almacenar el nombre y los apellidos de las personas responsables de cada empresa, por ello se opta por definir dos campos de tipo *Char*, “name” y “apellidos”, el primero de ellos se decide que al menos deba ser obligatorio ya que es posible que no se disponga de más información que el del nombre de un responsable.

El NIF se declara del mismo modo que el CIF para la clase Empresa, de tipo *Char* y no es necesario que el campo tenga contenido ya que es posible que se necesite crear una instancia de *Responsable* y no se disponga de este dato, el atributo lo nombraremos

como “nif”. La longitud debe ser de 9 caracteres en el caso de que el campo tenga información y se realiza una validación para comprobar si la cadena introducida cumple la regla de estos identificadores.

En alusión a los datos de contacto de la persona responsable se requiere en este caso de tres campos. En dos de ellos “telefono1” y “telefono2” se almacena un teléfono de contacto del responsable, se opta por ofrecer dos atributos con números de teléfono para disponer de la posibilidad de almacenar el número de teléfono fijo y el móvil, aunque no necesariamente deben ser de esta naturaleza. Se definen de tipo *Char* y su contenido queda limitado a 9 caracteres numéricos, para este atributo se debe realizar una validación ya que no es posible que su contenido empiece por una cifra diferente a 6,8 o 9. El otro atributo se empleará para recoger un correo electrónico de contacto, este campo se denomina “email”, se define de tipo *Char*, con longitud ilimitada y es validado según el patrón de definición de una cuenta de correo electrónico como ya se realizó para la clase *Empresa*.

También se ha solicitado guardar en cada instancia de la entidad *Responsable*, en caso de que sea necesario, el cargo de esta persona o algún tipo de información referido a su puesto de trabajo en la empresa proveedora. Para ello se declara un atributo “cargo” de tipo *Text* dando de este modo la posibilidad de disponer de varias líneas para rellenar con información de interés relacionada con las tareas de las que está encargada por su empresa.

Otro dato que se nos ha solicitado almacenar es el de si las instancias hacen referencia a un hombre o a una mujer. Para ello creamos un campo “sexo” donde se permita seleccionar entre “Hombre y Mujer”, esto se define en Tryton como un campo de tipo *Selection*.

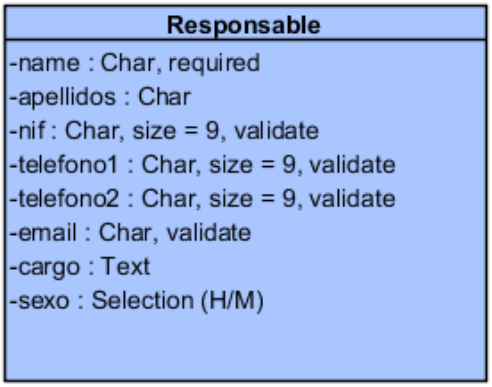



*Figura 2.4. Diagrama de clase Responsable*

Como relación lógica del módulo se va a establecer que una instancia de la clase *Empresa* puede tener asociadas de ninguna a varias instancias de la clase Responsable *(0,n)* y que una instancia de la clase *Responsable* puede tener asociado o una instancia de la clase *Empresa* o ninguna *(0,1)*. Para ello, según la forma de definir este tipo de relaciones en módulos Tryton, se emplea un campo de tipo *One2Many* de la clase Empresa que se define con el nombre de “contactos_id” que se relaciona con el atributo que se denomina “empresa_id” de la clase *Responsable*. En este caso el atributo “empresa_id” es de tipo *Many2One* y se relaciona con el atributo “id”, que contiene un valor secuencial que consta como clave primaria de la clase *Empresa* y que es creado automáticamente en la generación de las tablas en base de datos por Tryton. Una vez aclarado este asunto, se añade dicha relación a las clases *Empresa* y *Responsable* quedando los modelos como pueden observarse en la Fig 2.5.

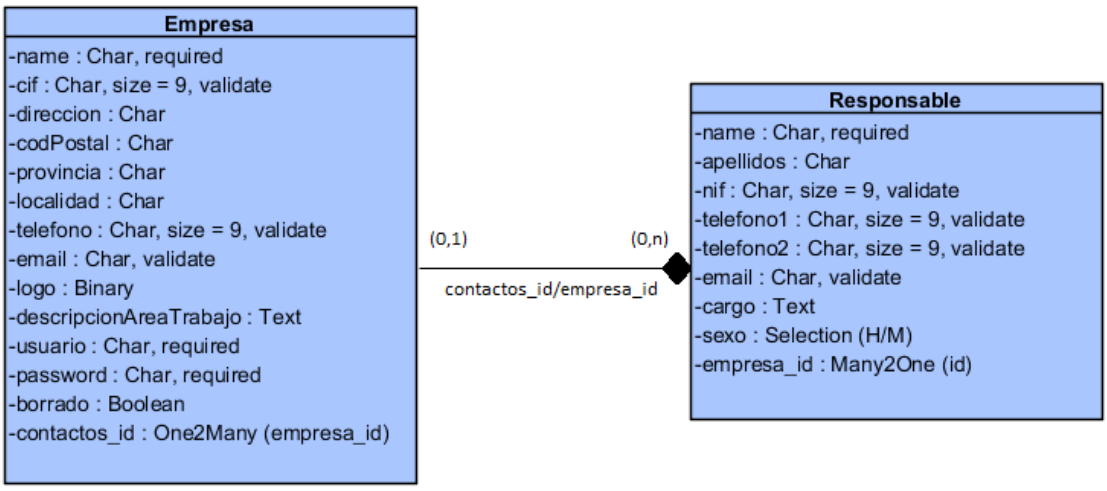



*Figura 2.5. Relación de clase Empresa y Responsable*

## Instalador

Para diseñar la entidad instalador, la opción que se toma es la de crear una clase nueva que se llamará “Instalador”, Fig 2.6. , bajo el atributo de clase *_name* se le asigna el nombre “instaladores.instalador” como viene establecido en las especificaciones de Tryton. Esta clase es creada para almacenar toda la información de interés de las personas encargadas de realizar cada una de las instalaciones que se asignen a la empresa a la que pertenezcan. Al igual que sucedía para los responsables de las empresas es muy posible que existan varios instaladores asignados para hacer efectiva la tarea de colocar el dispositivo en los locales por cada empresa proveedora. Para la empresa es importante disponer de esta información, así como de llevar un control por ejemplo de reportes de cada instalador o simplemente para analizar lo efectiva de sus instalaciones.

Dentro de la clase se dispone de los siguientes atributos, muy similares a los ya definidos para la entidad responsable, modelados según los requisitos listados anteriormente en la memoria:

Se solicita almacenar el nombre y los apellidos de todas las instancias de la clase Instalador, por ello se opta por definir dos campos de tipo *Char*, “name” y “apellidos”, el primero de ellos se decide que al menos deba ser obligatorio ya que es posible que no se disponga de más información que el del nombre de un instalador por diferentes motivos en los que no entraremos.

El NIF se declara de tipo *Char* y no es necesario que el campo tenga contenido ya que es posible que se necesite crear una instancia de Instalador y no se disponga de este dato, el atributo lo nombraremos como “nif”. La longitud debe ser de 9 caracteres en el caso de que el campo tenga información y se realiza una validación para comprobar si la cadena introducida cumple la regla de estos identificadores.

En alusión a los datos de contacto del personal instalador se requiere en este caso de tres campos para la clase. En dos de ellos “telefono1” y “telefono2” se almacena un teléfono de contacto del instalador, se opta por ofrecer dos atributos con números de teléfono para disponer de la posibilidad de almacenar el número de teléfono fijo y el móvil, aunque no necesariamente deben ser de esta naturaleza. Se definen de tipo *Char* y su contenido queda limitado a 9 caracteres numéricos, para este atributo se debe realizar una validación ya que no es posible que su contenido empiece por una cifra diferente a 6,8 o 9. El otro atributo se empleará para recoger un correo electrónico de contacto, este campo se denomina “email”, se define de tipo *Char*, con longitud ilimitada y es validado según el patrón de definición de una cuenta de correo electrónico como ya se realizó en clases anteriores.

Otro dato que se nos ha solicitado almacenar es el de si las instancias hacen referencia a un hombre o a una mujer. Para ello creamos un campo “sexo” donde se permita seleccionar entre “Hombre y Mujer”, esto se define en Tryton como un campo de tipo *Selection*.

Como se define en los requisitos, se debe disponer de información sobre qué áreas geográficas es capaz de dar cobertura dicho instalador y bajo qué condiciones, para ello se define un campo nombrado como “descripcionAreaTrabajo” de tipo *Text*, ya que se permite introducir un contenido multilínea, y en él se redactará cualquier información de interés al respecto.

También se requiere almacenar una foto de cada instalador en caso de que se disponga la posibilidad de obtenerla. Como ya se definió en el caso del atributo “logo” para la clase Empresa se crea también en este caso un atributo denominado “photo” de tipo *Binary* para manejar las imágenes.

Para registrar la preferencia de horario de cada instalador se empleará el tipo de campo *Time* que hasta el momento no se había requerido. Se crean dos atributos uno para definir la hora de inicio, “horainicio” y otro para la hora de fin, “hora fin” donde se establece el margen de preferencia de realización de trabajo para cada instalador. Para mantener la concordancia de los datos se debe realizar una validación para comprobar que la hora de fin es mayor a la hora de inicio, ya que el horario de trabajo debe ser según el horario comercial y no es posible que se dé el caso de que la hora de fin sea menor a la hora de inicio.

La empresa también necesita disponer de información sobre el número de versión del firmware Android y del modelo de teléfono móvil con el que se realizarán los registros de las instalaciones, con el objetivo de poder depurar específicos errores que se puedan encontrar en la aplicación para determinados modelos. Para estos campos se incluirán en el modelo dos atributos “terminal” y “versionAndroid” de tipo *Char* sin ninguna limitación en su contenido.

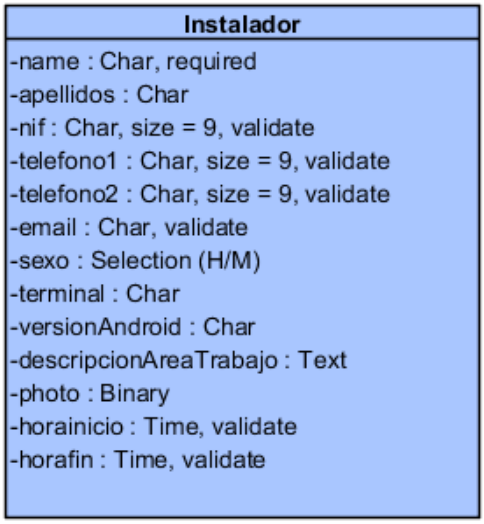



*Figura 2.6. Diagrama de clase Instalador*

Como ya se dejó constancia para la relación entre las clases *Empresa* y *Responsable*, para el caso de la clase Instalador se va a definir una relación equivalente a aquella. Para el proyecto, la funcionalidad requerida implica que una empresa puede tener en su plantilla desde ninguna a múltiples instancias de la clase *Instaladores (0,n)* y que una instancia de la clase *Instaladores* puede tener asociado o una instancia de la clase *Empresa* o ninguna *(0,1)*. Por esto se emplea un campo de tipo *One2Many* de la clase *Empresa* que se define con el nombre de “instaladores_id” que se relaciona con el atributo que se denomina “empresa_id” de la clase Instalador. En este caso el atributo “empresa_id” es de tipo *Many2One* y se relaciona con el atributo “id”, se recuerda que este atributo era un número secuencial que Tryton emplea internamente como clave primaria. Se añade dicha relación a las clases *Empresa* e *Instalador* quedando los modelos como pueden observarse en la Fig 2.7..

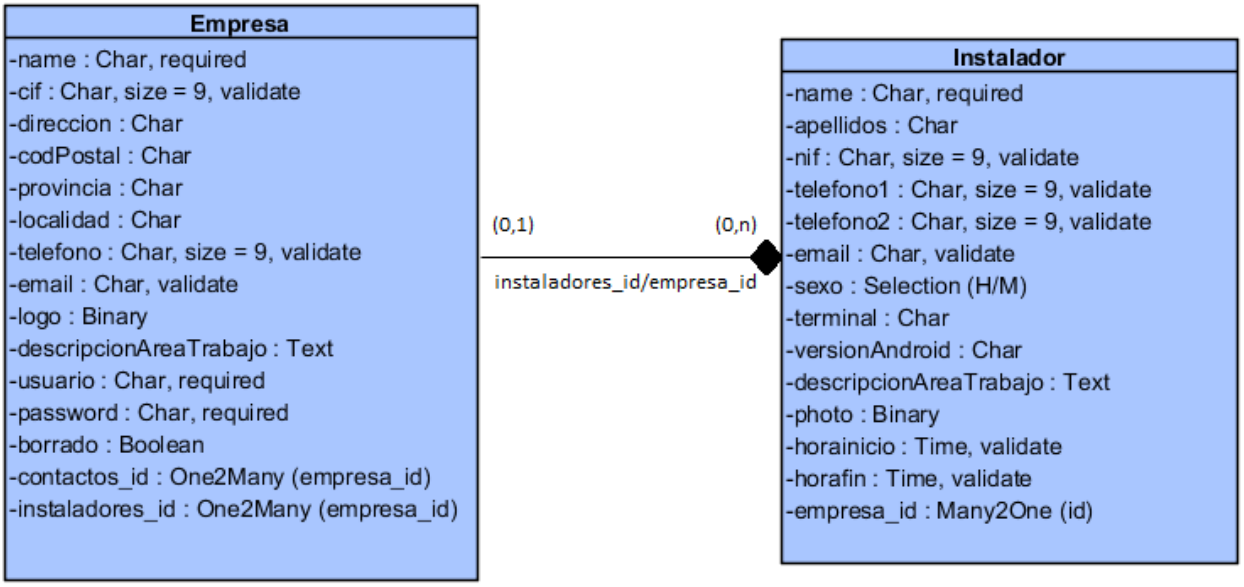



*Figura 2.7. Relación de clase Empresa e Instalador*

**Pedido**

La clase que a continuación se define representa el núcleo del módulo ERP que se está implementando; por lo tanto, dada la importancia que tiene, se realizará una descripción detalla del sentido de cada uno de los campos que se deciden incluir para que posteriormente se entienda el sistema completo sin excesiva dificultad.

Se decide crear una clase “Pedido” y como atributo de clase *_name* “instaladores.pedido”. La función de esta clase es la de facilitar al módulo modelar todos los pedidos que entren en el sistema, por lo tanto se tiene que dar cabida a los dos tipo de instalaciones que exclusivamente se permiten realizar. A continuación se explican las dos clases de instalaciones:

*Instalación tipo 1: Dispositivo Bluetooth*

La empresa proveedora en este caso asigna esta tarea a una persona, cuyo trabajo consiste en colocar el dispositivo desarrollo por Infortrex en una zona en la que sea accesible para los terminales móviles del público. Normalmente se realiza en el techo del local pero depende del permiso que conceda el propio responsable de la tienda.

*Instalación tipo 2: Dispositivo Bluetooth y pegatina promocional*

La empresa proveedora, para esta segunda clase, asigna de igual modo una persona que será la encargada de realizar la tarea. Este trabajo consiste en colocar el dispositivo Bluetooth de la misma forma que se ha explicado anteriormente para el tipo de instalación 1, con la condición que adicionalmente debe colocar una pegatina en algún lugar visible en el local, habitualmente en el escaparate o bien en un mostrador, de forma que sea accesible para las personas que accedan a la tienda.

Una vez explicado esto se pasa a detallar el modelado de los atributos que las instancias de la clase Pedido deben poseer en base a los requisitos que ha ofrecido la empresa.

Se requiere almacenar un identificador propio del sistema del cliente, a grandes rasgos su función es la de determinar un local en particular donde debe ser realizada la tarea de instalación. Para ello se establece un campo de tipo *Char*, y se denomina como “idtienda” siendo necesario que disponga de contenido en cualquier caso, ya que sin este identificador no es posible realizar su instalación.

Adicional a este identificador es necesario también almacenar el nombre de la tienda objetivo de sufrir una instalación; y en el caso de que se encuentre ubicado dentro de un centro comercial también necesitamos almacenar el nombre de la gran superfice. Para este fin se reservan dos campos; uno denominado “comercio” de tipo *Char* y cuyo contenido es obligatorio y otro llamado “centrocomercial” también de tipo *Char* pero en este gasto no contemplamos su obligatoriedad por lo que ya se ha comentado previamente.

Para poder ofrecer a los proveedores información sobre donde se ubica dicho local se debe guardar en el sistema su dirección completa (dirección, localidad, provincia y código postal) para este asunto se crean como es habitual varios atributos cuyos campos son de tipo *Char* y de carácter obligatorio. Estos atributos se denominan “direcciontienda”, “localidad”, “provincia” y “codpostal” respectivamente.

También se necesita almacenar información sobre una persona de contacto de la tienda, para poder acordar una fecha y las condiciones de la instalación en cuestión. Para ello se requieren el nombre, los apellidos, al menos un teléfono de contacto de esta persona, un correo electrónico y el cargo de esta persona en la tienda. Todos los campos se definen de tipo *Char* y se denominan del siguiente modo “nombre”, “apellidos”, “telefono1”, “telefono2”, “email” y “cargo”, pero solamente los campos “nombre”, “apellidos” y “telefono1” son obligatorios para poder crear una instancia de esta clase, siendo limitados el contenido de los campos reservados para almacenar números de teléfono a nueve caracteres. Es importante resaltar que el sistema debe validar el contenido de los campos que almacenan teléfonos y correo electrónico para impedir guardar contenido erróneo, con las reglas ya explicadas en casos similares de otras clases.

Según las especificaciones es necesario guardar, para cada pedido, que tipo de instalación es requerida, siendo posible únicamente tomar una de las dos clases de instalaciones explicadas. Para ello se define un campo de tipo *Selection* que se denomina “tipo” con únicas opciones posibles de almacenar “BT” para una instalación tipo 1 o bien

“BT+Bidi” para una *instalación tipo2* siendo obligatoria la selección de alguna de las alternativas dadas.

Una vez definido el tipo de instalación que se requiere se pasa a declarar los datos necesarios para cada tipo. En cuanto a los datos referentes al dispositivo Bluetooth se solicita guardar información cuyo contenido es un identificar propio del cliente, al igual que para el identificador tampoco se necesita conocer su utilidad pero principalmente sirve para reconocer el dispositivo instalado con el sistema propio del cliente y un breve comentario para informar al proveedor los detalles sobre dónde y cómo se realizará la instalación de los Bluetooth en el local. Para ello se crean dos atributos con campos de tipo *Char* llamados “idbluetooth” y “lugarBluetooth” cuyo contenido debe ser obligatorio para el atributo en el que se almacenará el identificador del dispositivo.

En el caso de que se haya registrado el pedido como una tarea de *instalación de tipo 2* es necesario, aparte de guardar la información del *tipo1*, almacenar la información equivalente para la pegatina promocional a la de los dispositivos Bluetooth, es decir un identificador propio del cliente y un comentario con información para el instalador donde se puede detallar donde ubicar dicha pegatina. Se emplean dos campos de tipo *Char* “idbidi” y “lugarBidi” para ello, donde se debe comprobar en cualquier alta de pedido o en cualquier modificación que los datos son concordes al tipo de instalación, no permitiéndose por ejemplo almacenar un “idbidi” cuando el tipo de instalación registrada es de tipo Bluetooth o que no exista contenido en el campo “idbidi” cuando el tipo de instalación es de tipo Bluetooth y Bidi promocional.

Adicionalmente a esta información se debe dar la posibilidad de almacenar una breve descripción del trabajo a realizar con información sobre el margen de horario disponible para realizar la tarea o cualquier otro asunto que se considere oportuno comunicar al instalador, por ello se reserva un campo denominado “observaciones” de tipo *Text*, para disponer de esta funcionalidad.

Un objetivo importante para esta clase, aparte de llevar el control y gestionar los pedidos, es la de ser capaz de crear una hoja de pedido en formato PDF con los datos que contiene cada instancia para ser enviada al proveedor que corresponda. Dentro de esa hoja de pedido debe figurar una imagen con otro código Bidi según el patrón que se ofreció en las especificaciones. Para ello se define un atributo “bidi” de tipo Binary que se deberá de actualizar cada vez que algún dato de los que componen dicha imagen sufra alguna modificación. En este caso “idbluetooth” y “idbidi”, siendo “0” “idbidi” en el caso de que el tipo de instalación sea exclusivamente de Bluetooth.

También se requiere disponer para el estudio y la gestión de las instalaciones, de dos fechas, estas fechas deben ser la fecha de creación de la instancia en el sistema y la fecha pactada de su instalación por un proveedor. Se almacenan en dos campos llamados “fecharecepción” y “fechaasignacion”, de tipo *Date* para el caso de la fecha de creación y de tipo *DateTime* para el caso de la fecha y hora asignada para ser concluida.

Para que la empresa pueda llevar un control más global de los pedidos, se requiere disponer de un campo que se deberá ir actualizando automáticamente para registrar en que momento del ciclo de vida se encuentra el pedido, en este caso se crea un campo de tipo *Selection* llamado “estado” y son bloqueadas las posibilidades de modificación directamente por el usuario. En el caso de que ya haya sido realizada el estado será “Finalizado”, en el caso de que haya sido asignado a una empresa de la red de proveedores el estado será como “Asignado” y para el resto de los casos como “Pendiente”. En la Fig 2.8. puede verse el resumen de atributos de la clase.

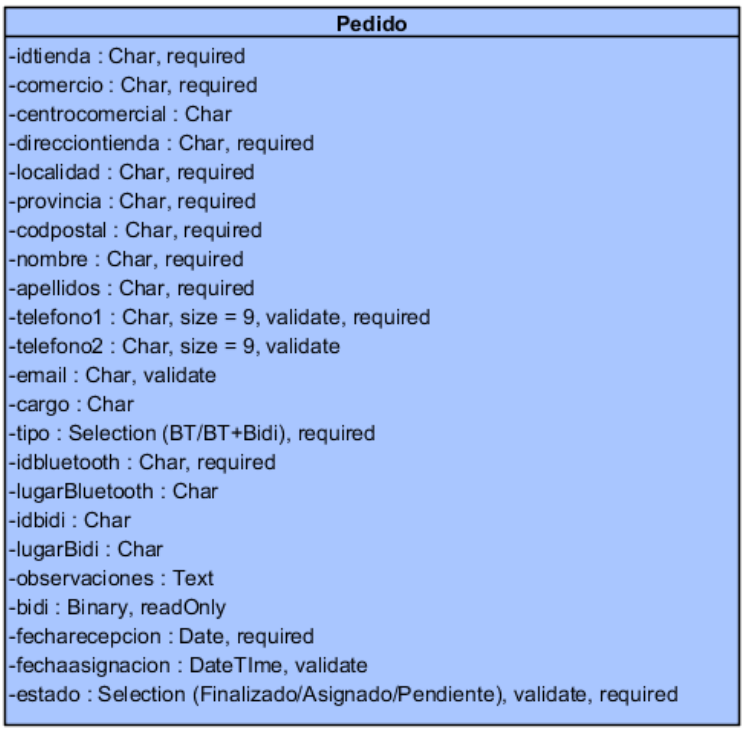



*Figura 2.8. Diagrama de clase Pedido*

Para poder llevar a cabo la asignación de una empresa proveedora a un pedido determinado se necesita establecer una relación entre las clases “Empresa” y “Pedido”. Para el proyecto, la funcionalidad requerida implica que una empresa puede tener en su calendario desde ninguna a múltiples instancias de la clase Pedido *(0,n)* y que una instancia de la clase “Pedido” puede tener asociado o una instancia de la clase “Empresa” o ninguna *(0,1)*. Por esto se emplea un campo de tipo *One2Many* de la clase “Empresa” que se define con el nombre de “pedidos_id” que se relaciona con el atributo que se denomina “empresa_id” de la clase “Pedido”. En este caso el atributo “empresa_id” es de tipo *Many2One* y se relaciona con el atributo “id” el número secuencial que ya se empleó para otras relaciones. Se añade dicha relación a las clases “Empresa” y “Pedido” quedando los modelos como pueden observarse en la Fig 2.9..

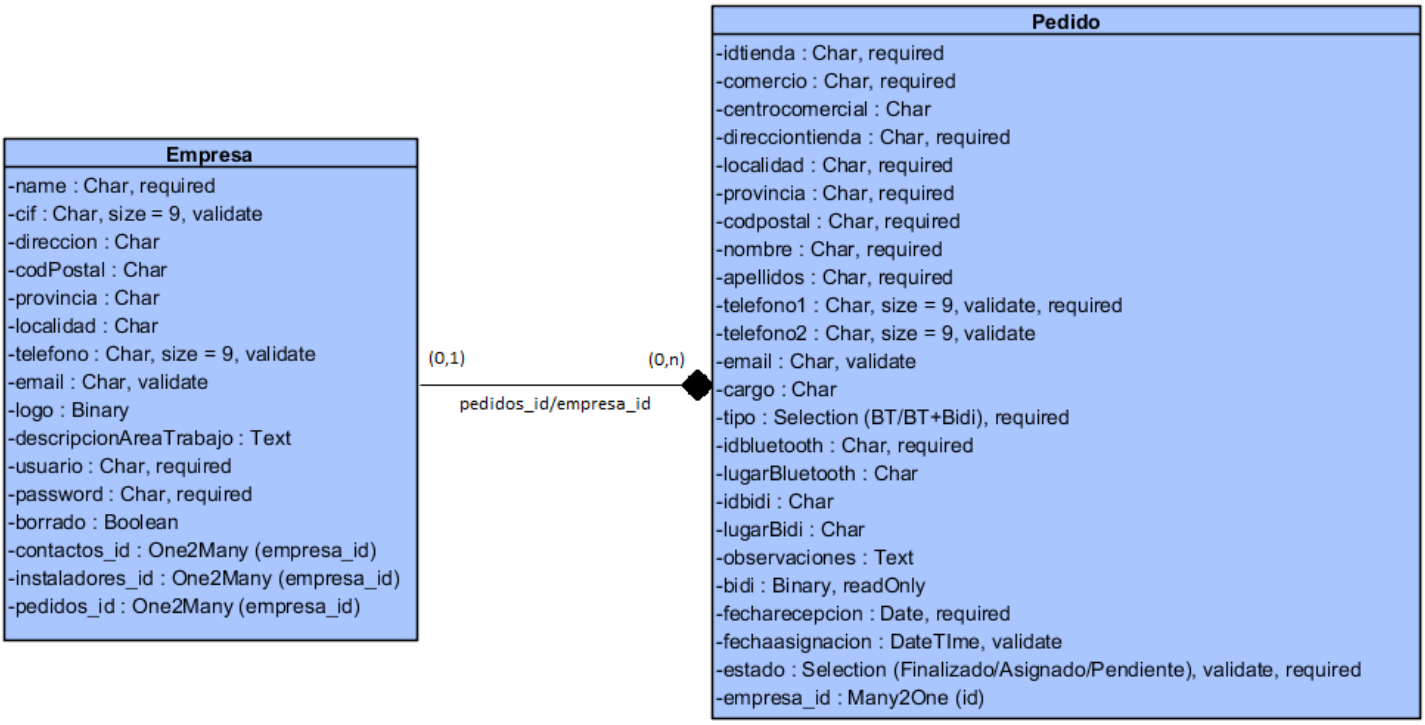



*Figura 2.9. Relación de clase Empresa y Pedido*

## Instalación

A continuación se diseña la entidad que se encarga de gestionar las instalaciones ya realizadas, esta clase posee una característica especial en comparación con las clases ya creadas hasta el momento ya que las instancias de esta clase serán las que se creen gracias a las interacciones con los terminales móviles de los instaladores. La creación de instalaciones no será posible de realizar con el cliente Tryton, ya que para los campos que conforma la clase serán establecidos como de solo lectura. Se decide crear una clase "Instalacion", Fig 2.10., y como atributo de clase *_name,* "instaladores.instalacion".

Para almacenar las instalaciones se requiere guardar principalmente los datos del dispositivo que se instala, es decir, la información que proporciona el Bluetooth 2.0, la carcasa del dispositivo y la pegatina promocional. No se requiere almacenar la

información del Bluetooth 4.0 ya que en estos momentos no se dispone en el mercado de suficientes modelos con el sistema operativo Android capaces de interactuar con esta versión de módulos Bluetooth. Por este motivo la información del Bluetooth 4.0 viene codificada en el código Bidi de la carcasa del dispositivo aunque está previsto que en el caso de que las nuevas versiones dispongan de esta funcionalidad se pasará a utilizar en el módulo la interacción directa con esta versión de Bluetooth. Se crean tres campos denominados “macbluetooth2”, “macbluetooth4” y “macbidi” de forma que para almacenar esta información se definen como campos de tipo *Char* y se establece que su contenido no sea posible de modificar por el cliente Tryton.

Para obtener el número secuencial que se establece en los requisitos ofrecidos por la empresa se utiliza el atributo que se crea en Tryton por defecto en cada modelo. De esta forma tenemos un identificador único para cada instalación realizada, como ya se ha comentado en la clase “Empresa”, se denomina con el nombre de “name” y también se establece como no modificable al tratarse de un valor crítico de la clase.

Para guardar la fecha y la hora en la que se produce la creación de una instancia de instalación, es decir la fecha de instalación por parte del proveedor, se emplea un campo de tipo *DateTime*, “fechainstalacion” y se establece como el resto de campos de la clase “Instalador” que no se permita modificar bajo ninguna circunstancia.

Desde la aplicación móvil se va a solicitar al instalador en cuestión que realice una fotografía de la tarea que ha realizado, de forma que la empresa pueda corroborar que el estado del trabajo es satisfactorio para el cliente. Para ello se emplea un tipo de campo *Binary*, que denominamos “photo” y cuyo contenido no es posible modificar. También se solicita, en el caso de que el instalador lo considere oportuno, introducir en la aplicación un comentario con la instalación que ha finalizado. Para ello se establece un campo de tipo *Text* que se nombra como “comentarios” y cuyo contenido también se considerará para el cliente Tryton como de solo lectura.

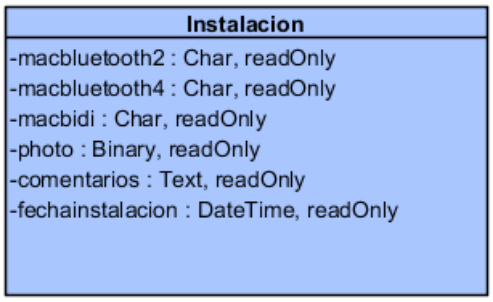



*Figura 2.10. Diagrama de clase Instalacion*

Todas las instancias de la clase “Instalacion” requieren de manera obligatoria una relación de uno a uno, con la clase “Pedido” ya que no es posible, para la lógica del sistema, que una instalación realizada no disponga previamente de un pedido en el que se haya basado el trabajo. Para las relaciones de tipo *One2One* en Tryton es necesario crear una clase adicional que será la encargada de enlazar ambas clases. A esta clase se la nombrará como “One2OnePedidoInstalacion”, bajo el atributo *_name* contendrá el identificador de clase “instaladores.one2one.relation” y se estable que el atributo al que hará referencia será el *id* tanto para la clase “Pedido” como para la clase “Instalacion” de forma que queda identificada la relación de manera unívoca en ambos sentidos. Estos atributos deben ser de tipo *Many2One* en la clase “One2OnePedidoInstalacion” y si se desea tener esta referencia en las clases enlazadas se debe crear su atributo *One2One* correspondiente, en el caso que estamos diseñando únicamente interesa crear este atributo en la clase “Pedido” y se establece bajo el nombre de *instalación_id*.

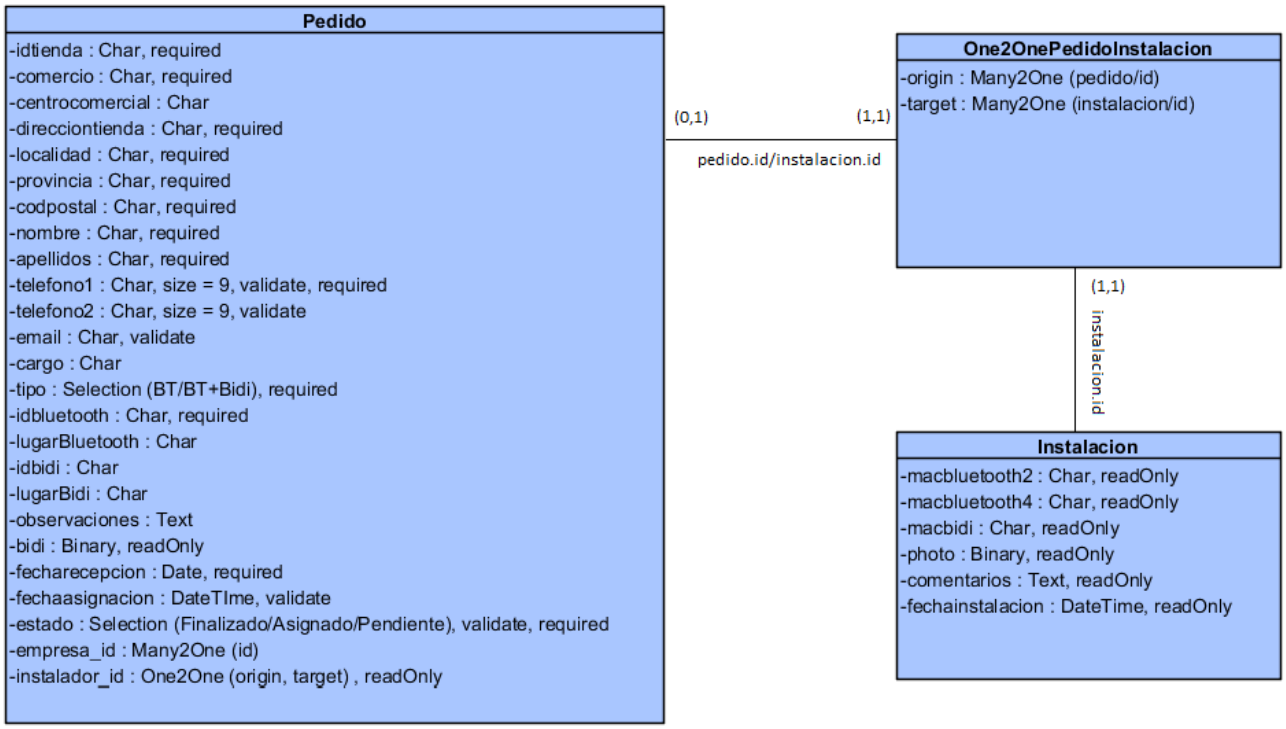



*Figura 2.11. Relación de clase Instalacion y Pedido*

De esta forma queda completado el diseño del modelado del módulo ERP, todas las entidades quedan explicadas en detalle siendo el esquema de datos el que se expone en la Fig 2.11..

En la Fig 2.12. puede verse el esquema de clases completo para el módulo ERP.

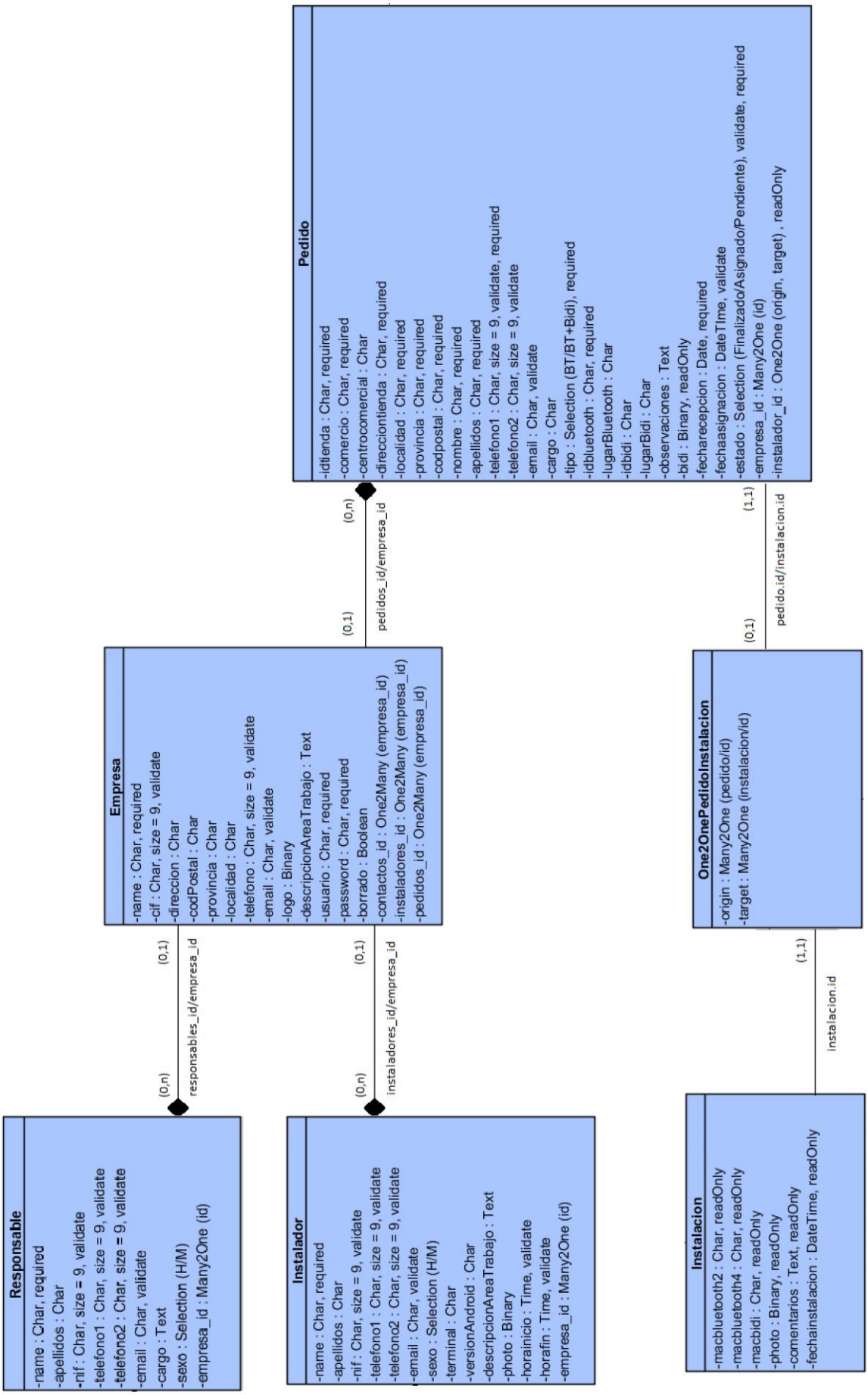



*Figura 2.12. Diagrama de clases del sistema completo*

### *2.3.4 Diseño de las vistas*

A lo largo de este apartado se describe el diseño de las vistas del módulo Tryton, es decir, se define como interpretará el cliente Tryton la generación de las pantallas de la interfaz de usuario. Para ello se diseña la forma en que se gestionará el lanzamiento de las diferentes vistas y se crearán dos tipos de vistas para cada entidad.

Se opta por diseñar primeramente una vista tipo tabla, para observar de una forma resumida todas las instancias creadas con información relevante en sus columnas. Mediante esta tabla se debe ser capaz de acceder a cada una de las instancias de manera independiente y consultar los datos que contenga. Posteriormente se diseña la vista de formulario para cada entidad de forma que cada vista estará compuesta de varias secciones, donde en cada una de ellas se ubican los campos con información relacionadas por sus características de contenido. Basando el diseño en estas condiciones se presentan los bocetos de cada vista.

**Empresa**

Para la vista tipo lista se decide que los datos importantes que deben figurar son primeramente los CIF de las empresas, seguido de las razones sociales y posteriormente los datos de contacto (el correo electrónico y el teléfono), de esta forma se obtiene la información de contacto necesaria de todas las empresas en una misma vista. En la Fig 2.13. se observa el boceto de la vista en la que se basará la posterior implementación, donde la sección “Cabecera Tryton” representa las opciones disponibles por defecto para la vista tipo lista, donde se ubican iconos por ejemplo para crear un nuevo registro, cambiar de vista, refrescar pantalla o guardar cambios entre otras posibilidades. La sección “Menú Tryton” contiene el árbol de los módulos disponibles para el usuario que ha

realizado el logado en el sistema, además está también la sección de configuración del sistema.

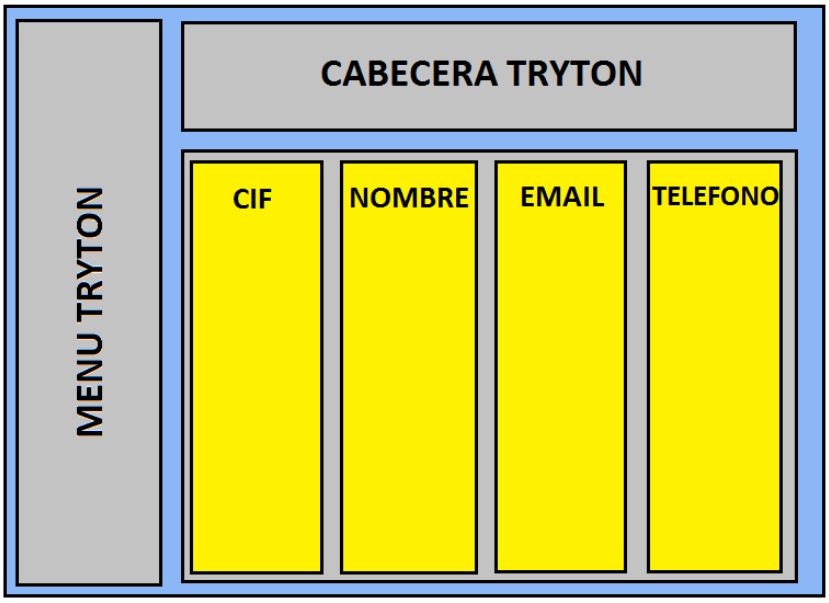



*Figura 2.13. Diseño vista listado de clase Empresa*

Para la vista de formulario se establece la apariencia de una ficha descriptiva de las empresas como se observa en la Fig 2.14.. Primeramente se ubica una sección con los datos que se muestran en la vista tipo lista, pero en este caso en formato formulario presentando los datos con el estilo etiqueta y cuadro de texto acompañado de los datos de registro de la empresa en la aplicación Android, al mismo nivel se adjunta el logo de la empresa. A continuación se muestra información referente a la dirección de la sede, oficina o lugar de trabajo en una sección individual. Como tercera y cuarta sección figuran dos vistas de tipo lista iguales a las que se diseñaran posteriormente para mostrar tanto los responsables de los que disponga la empresa como de todo el personal instalador que disponga en plantilla pudiendo acceder a las propias vistas de Instalador y Responsable seleccionando la que se desee, y finalmente, se reserva espacio para mostrar la descripción de la empresa así como sus restricciones en formato texto.

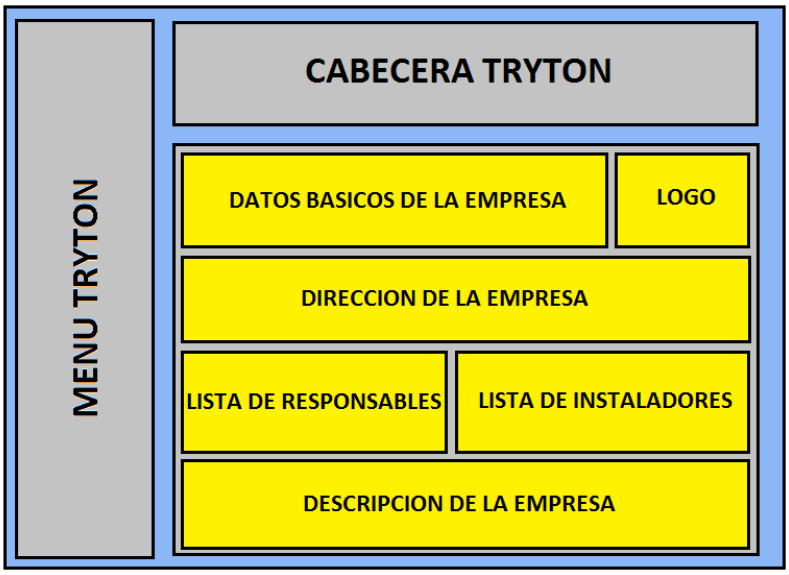



*Figura 2.14. Diseño vista formulario de clase Empresa*

**Responsable**

Para la vista tipo lista se pretende resumir los datos de contacto de todas las personas que sean instancia de la clase Responsable, por este motivo se opta por mostrar los teléfonos de contacto y el correo electrónico acompañado del nombre completo de cada persona y su NIF, de esta forma se obtiene la información de contacto necesaria de todas las responsables del sistema en una misma vista. En la Fig 2.15. se observa el boceto de la vista en la que se basará la posterior implementación.

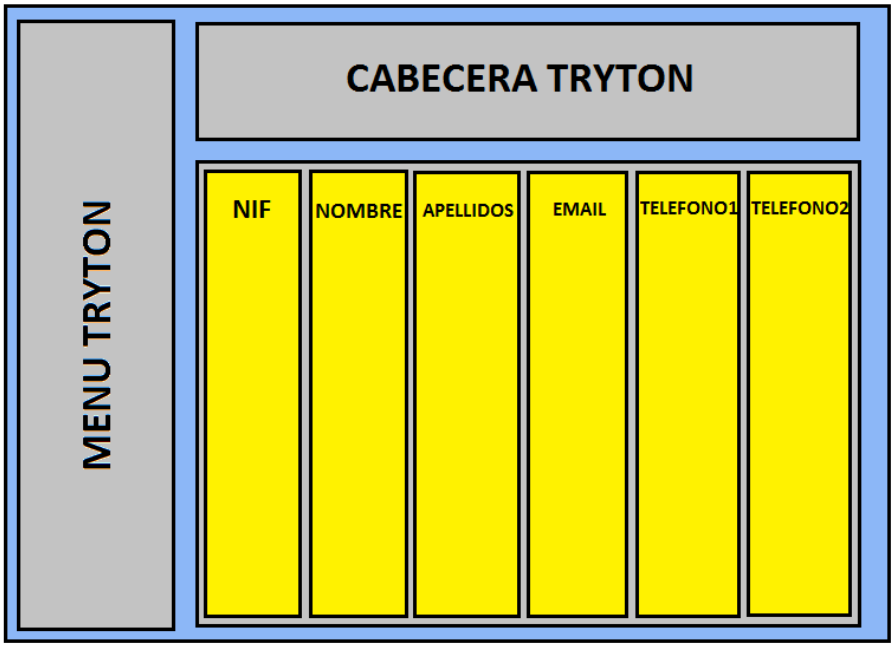



*Figura 2.15. Diseño vista listado de clase Responsable*

Para el diseño de la vista tipo formulario, Fig 2.16., se opta por dividir la pantalla en dos áreas. La primera sección se reserva para mostrar la información que hace referencia a la empresa a la que pertenece dicho responsable, es decir, tanto el nombre de la empresa, con la posibilidad de abrir la vista de formulario de la empresa si se desea como el puesto de trabajo que ocupa esta persona en la empresa. La sección inferior se emplea para mostrar el resto de datos personales de la persona responsable instanciada.

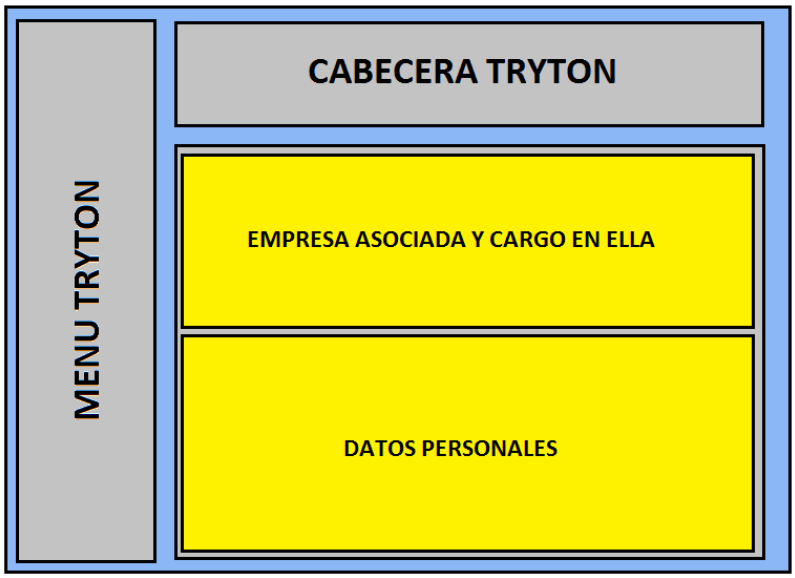



*Figura 2.16. Diseño vista formulario de clase Responsable*

**Instalador**

Como boceto para implementar la visto tipo lista, al desearse mostrar los mismos datos que para el caso de las personas responsables de cada empresa que son el NIF, el nombre, los apellidos, el correo electrónico y los teléfonos de contacto; se emplea esta misma estructura mostrada en la Fig 2.16., por lo tanto dispondremos de una vista equivalente a la explicada para listar todas las instancias de la clase Instalador.

La estructura que presentará la vista tipo formulario será  al estilo de la ya bocetada para el caso de la empresa como puede observarse en la Fig 2.17.. La foto se debe ubicar en la parte superior derecha de la pantalla y a su izquierda se mostrarán los datos personales así como los datos más específicos del modelo de móvil que empleará dicha persona para realizar las tareas que se le encomienden. Para terminar de mostrar todos los datos de interés para la empresa, en la parte posterior de la vista se ubicarán la información relacionada con la empresa para la que trabaja, de entre todas las registradas en el sistema, y las preferencias del instalador para realizar las instalaciones.

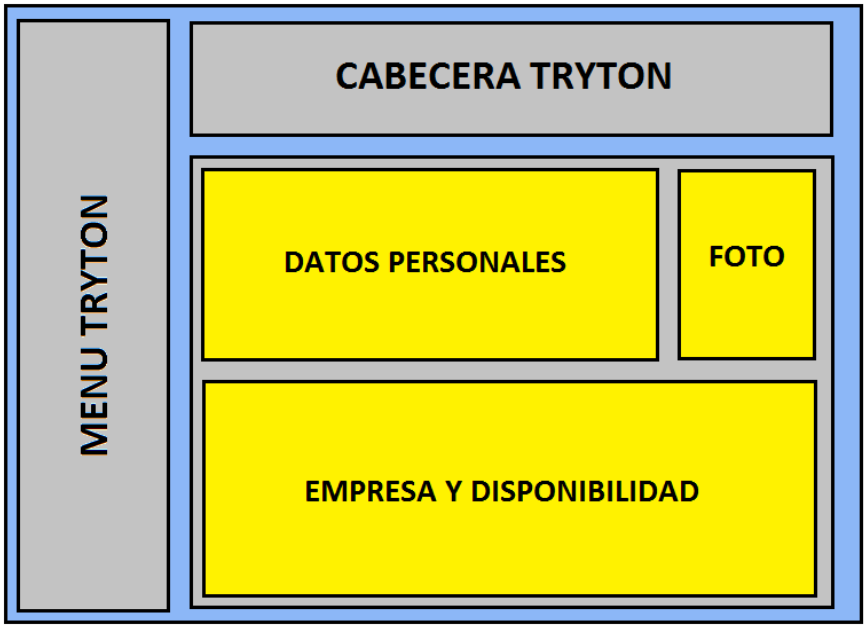



*Figura 2.17. Diseño vista formulario de clase Instalador*

**Pedido**

Para la vista con el estilo de lista se establece que los datos importantes que deben figurar son primeramente el id de pedido, el nombre del local y su dirección para reconocer de que comercio se está haciendo referencia ya que es muy probable la existencia en el sistema de varios comercios con el mismo nombre, por ejemplo en el caso de que se hable de una cadena de tiendas, el tipo de instalación que el cliente demandaba o demanda, el estado del pedido y en el caso de que el pedido ya haya sido realizado por algún proveedor también se debe mostrar el identificador de la tarea realizada. En la Fig 2.18. se observa el boceto de la vista en la que se basará la posterior implementación.

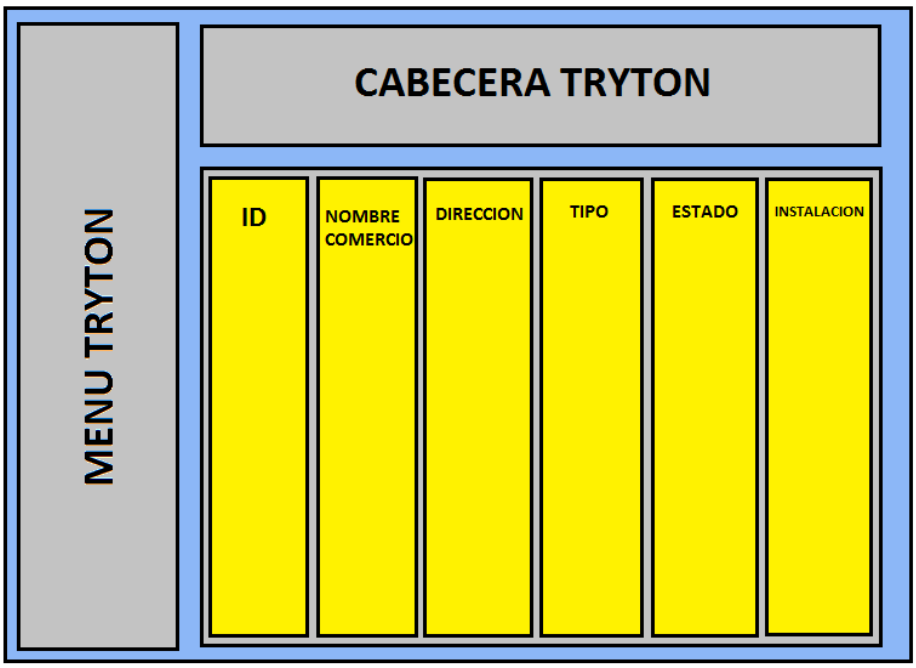



*Figura 2.18. Diseño vista listado de clase Pedido*

El diseño de la vista formulario se refleja en la figura Fig 2.19, se opta por dividir la pantalla esencialmente en tres secciones. En la primera de ellas se muestran datos generales de la instancia, como por ejemplo la fecha de creación del pedido o su estado acompañado del código Bidi que se debe autogenerar en Tryton dependiendo de los datos adicionales de la instancia. En la segunda de la secciones presentes en la vista se debe mostrar los datos del local en el que se pretende hacer efectiva la tarea que el cliente nos encarga, siendo estructurada el área en dos subapartados empleando el primero para los datos propios del local y un segundo para adjuntar los datos de la persona de contacto de la tienda. La sección inferior se dedicará para reflejar los detalles que deberán ser utilizados por la aplicación Android y también para poder generar el código Bidi adjunto en el pedido. Como aspecto a resaltar de la vista es que se debe diseñar para que el usuario del módulo ERP asigne dicho pedido visible en la vista a la empresa que se desee y por otro lado se debe ser capaz de visualizar la instancia de la instalación que haya sido creada por la aplicación del terminal móvil del instalador en el caso de que se hubiera sido realizado el trabajo.

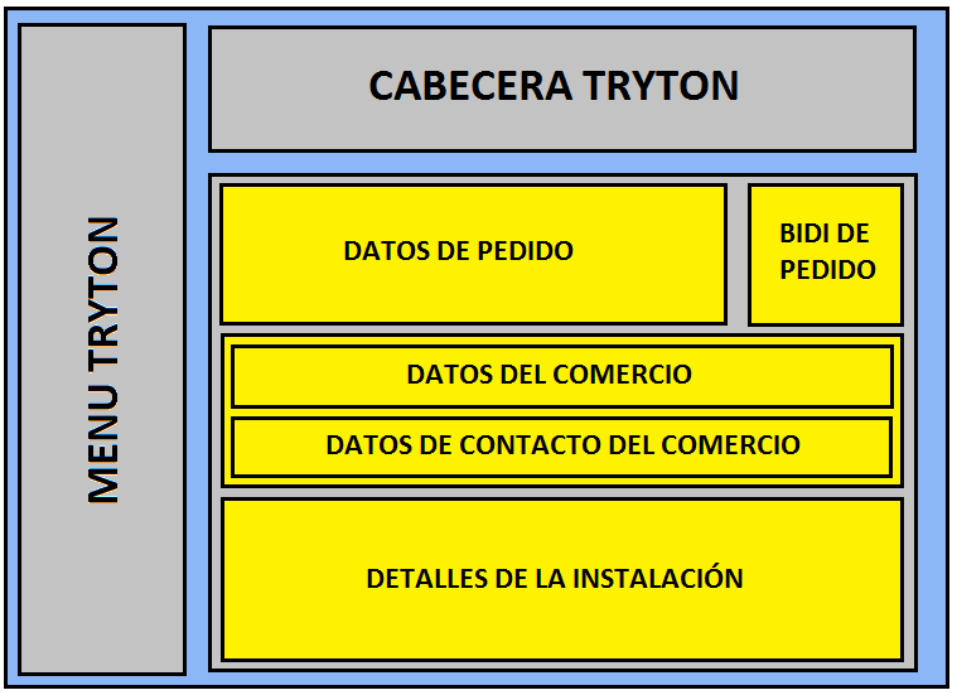



*Figura 2.19. Diseño vista formulario de clase Pedido*

**Instalación**

Para el caso de la entidad Instalación, en la vista tipo tabla, no se considera que deban figurar datos con demasiados detalles por ello, como puede verse en la Fig 2.20., únicamente se exponen los identificadores de todas las instancias del modelo y la fecha de su ejecución, con el objetivo de poder contabilizar de alguna manera y hacerse a la idea del volumen de instalaciones que se han concluido en un margen de fechas.

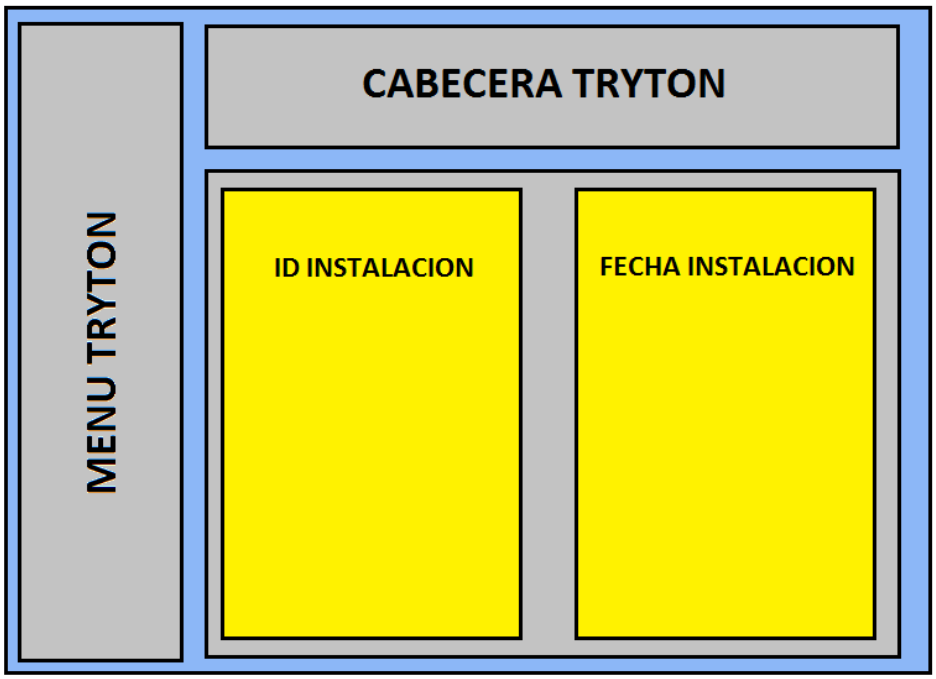



*Figura 2.20. Diseño vista listado de clase Instalacion*

Toda la información que el instalador, el cual ha sido encargado por su empresa para realizar la labor, necesite registrar en el sistema se debe poder observar en esta vista que se diseña a continuación y que puede verse a modo de esquema en la Fig 2.21.. A parte de la información que a la empresa le interesa registrar para valorar que toda la instalación ha sido ejecutada correctamente, aquí se encuentran la mayor parte de los datos junto a alguno, que más adelante se explicará, de la entidad “Pedido” que el cliente requiere para poder continuar con su negocio y que desde nuestro sistema debemos transmitirle simultáneamente al registro en el sistema que se está describiendo en esta memoria; de forma que por parte del cliente se pueda dar por finalizada. En la parte superior de la vista se deberán mostrar los datos que aparecían en la vista con aspecto de lista es decir el identificador y la fecha de realización. A su derecha deberá aparecer la foto que deberá ser tomada por el instalador en el momento de realizar el trabajo mostrando, con la mayor claridad posible, la manera y el lugar de como ha sido ubicado el dispositivo Bluetooth en dentro del local. Y para finalizar el boceto, en la parte inferior de la vista se mostrarán todas las cadenas que el instalador ha debido recoger con la aplicación móvil.

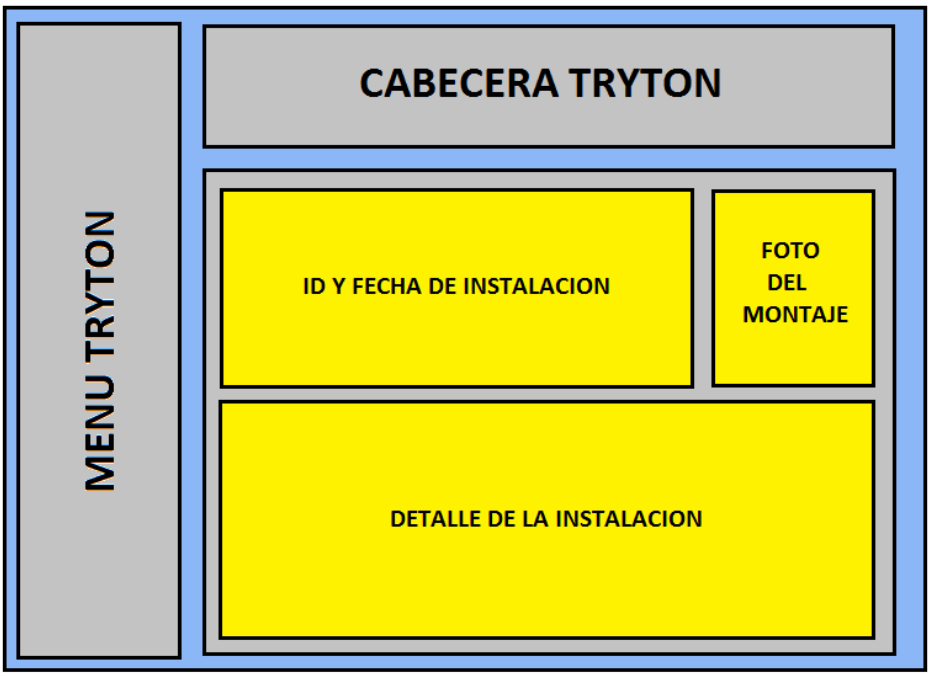



*Figura 2.21. Diseño vista formulario de clase Instalacion*

# 2.4 Implementación del módulo

Una vez que se dispone de todas las especificaciones, tanto de funcionalidad como de diseño, del módulo Tryton que se debe implementar en este apartado se pasa a explicar de qué manera ha sido desarrollado el módulo completo desde su definición dentro de la plataforma Tryton hasta las validaciones de los campos que así lo requieren.

## *2.4.1 Creación del módulo*

Para definir un nuevo módulo en el entorno Tryton primeramente se debe crear una carpeta dentro del servidor, más específicamente en la ruta *trytond/modules*. Para el módulo que estamos desarrollando nombraremos esta carpeta como “instaladores” de forma que bajo este directorio se guardaran todos los recursos que se necesiten.

Primeramente se debe crear un archivo llamado *_init_.py*. Este fichero Python importará a todos los demás archivos escritos en Python dependientes del módulo instaladores. En este caso son todos los ficheros que se encuentren en la carpeta ya que el cron que se empleará para la generación automática de pedidos se aloja en un primer momento en otra máquina o directorio. Por lo tanto el contenido de este archivo debe ser el siguiente:

```
from instaladores import *
```

Posteriormente el módulo necesita disponer de un nuevo fichero que se debe llamar *_tryton_.py*, donde se especifican algunos datos de configuración del módulo, traducción en el caso de que se disponga del correspondiente fichero de traducción o de información del propio módulo tal como el autor, la versión o una breve descripción. Este fichero escrito en Python debe contener exclusivamente de un diccionario y con unas palabras clave determinadas por la arquitectura de Tryton. El contenido del módulo es el siguiente:

```
{
    'name' : 'Instaladores Infortrex',
    'version' : '1.0.0',
    'author' : 'fpeiro',
    'email': 'fpeiro@outsidebuilder.com',
    'website': 'http://www.infortrex.es',
    'description': 'Modulo para la gestion de instalaciones',
    'depends' : [
        'ir',
    ],
    'xml' : [
        'instaladores.xml',
    ],
    'translation': [
    ],
}
```

Entre toda la información, a parte de los datos como el nombre del módulo, la versión, el autor el correo electrónico de contacto, pagina web y la descripción, se debe hacer referencia al campo *depends* que especifica los módulos de los que depende este propio módulo. El módulo que figura en el fichero llamado 'ir' es un módulo instalado por defecto en el servidor Tryton y que siempre va a acompañar a todos los módulos Tryton ya que sirve de apoyo para generar las vistas y los modelos. Otro campo importante es el de XML donde se referenciará a todos los ficheros que se necesiten del módulo, es decir todas las vistas ya que se definen como ya se ha comentado en XML.

### *2.4.2 Implementación de los modelos*

Para la implementación de todos los modelos, que se han diseñado previamente, se va a emplear un fichero llamado "instaladores.py" y basando el desarrollo en las condiciones tanto de la fase de requisitos como la de diseño se trabaja en ello. Es importante destacar que en este fichero se encuentran implementados todos los modelos definidos pero a modo descriptivo se va a entrar a fondo en el modelo que se ha implementado para la entidad "Pedido" ya que resulta ser el más completo y el esqueleto del módulo. A continuación se adjunta la porción de código, debidamente comentado, que contiene la definición de la clase Pedido:

```
#Definicion de la clase Pedido
class Pedido(ModelSQL, ModelView):

   #Nombre para la clase
   'Pedido'
   #Nombre de la tabla que se generará en BBDD
   _name = 'instaladores.pedido'
   _description = __doc__
```

Posteriormente se añade la declaración de todos los campos que describen la clase. Se adjunta esta declaración para campos de diferentes tipos exclusivamente ya que no se considera necesario adjuntar el código al completo.

```
    #Campos que conforman la clase Pedido

    #Identificador del cliente para cada tienda
    idtienda = fields.Char('Numero de Tienda',required=True)

    #Tipo de instalación a llevar a cabo
    tipo = fields.Selection([('BT', 'Bluetooth'),('BT+Bidi', 'Bluetooth y
Bidi'),],'Tipo de Instalacion',required=True)

    #Comentarios especiales para la empresa proveedora o el instalador
    observaciones= fields.Text('Observaciones')

    #Fecha en la que ha sido creado el registro
    fecharecepcion = fields.Date('Fecha recepcion de Pedido',required=True)

    #Fecha pactada con la tienda para realizarse la instalacion
    fechaasignacion = fields.DateTime('Fecha pactada de instalacion')

    #Referencia a una instancia de la clase Instalacion (One2One) a
    #través de la clase instaladores.one2one.relation
    instalacion_id =
fields.One2One('instaladores.one2one.relation','origin','target','Instalacion',r
eadonly=True)

    #Referencia a una instancia de la clase Empresa (Many2One)
    empresa_id = fields.Many2One('instaladores.empresa','id')

    #Bidi->imagen con el bidi que figurara en la hoja de pedido
    bidi= fields.Binary('Bidi', readonly=True)
```

Únicamente falta por controlar las diferentes validaciones de la clase. Para ello se define dentro del método *init* de la clase las restricciones a tener en cuenta y los mensajes de error a mostrar. En este momento se adjunta el código para una validación, por ejemplo la validación del número de teléfono.

```
    #Método init
    def __init__(self):
        super(Pedido, self).__init__()

        #Se definen restricciones para cada validación que sea necesaria,
```

```
        #el primer parámetro hace referencia al método que debe ser
        #ejecutado el segundo el mensaje en caso de no superar la
        #validación
        self._constraints += [
            ('check_telefono1', 'telefono_incorrecto'),
        ]

     #Se definen los mensajes de error de las validaciones
        self._error_messages.update({
            'telefono_incorrecto': 'Numero de telefono incorrecto'
        })
```

Fuera del método *__init__* se define el método que realiza la comprobación, en este caso se denomina *check_telefono1*,

```
    #Validacion de telefonos, en el caso de
    #que el telefono introducido no comience por 6,8 o 9 y sus demás
    #caracteres no sean numéricos se debe alertar con un mensaje de error
    def check_telefono1(self, ids):
        for pedido in self.browse(ids):
            telefono = pedido.telefono1
            numeros="689"
             numeros2="0123456789"
            if (len(telefono)==9):
                if numeros.find(telefono[0]) >=0:
                      if
((numeros2.find(telefono[1])>=0)and(numeros2.find(telefono[2])>=0)and(numeros2.f
ind(telefono[3])>=0)and(numeros2.find(telefono[4])>=0)and(numeros2.find(telefono
[5])>=0)and(numeros2.find(telefono[6])>=0)and(numeros2.find(telefono[7])>=0)and(
numeros2.find(telefono[8])>=0))
                        return True
                      else:
                        return False
                else:
                    return False
            else:
                if len(telefono)==0:
                    return True
                else:
                    return False
```

Finalmente se registra la clase en el Pool de clases de forma que se permite la creación de objetos y su referenciación.

```
   #registro de la clase en el Pool de clases
```

```
Pedido()
```

### 2.4.3 *Implementación de las vistas*

Como ya se ha explicado anteriormente, a modo de resumen, los objetos pueden ser presentados con diversas vistas y mediante una acción se abre una ventana y se establece que vista mostrar. Para definir las vistas se emplea fielmente los bocetos creados durante la fase de diseño de vistas. Para esta parte de la memoria se describirá primeramente como se maquetan las dos vistas diseñadas, tipo árbol y tipo formulario para posteriormente se explicar las acciones que lanzan las vistas y la definición en la barra de navegación de Tryton del módulo implementado. Todas las vistas se crean en un fichero XML como ya se pudo observar en el fichero de definición del módulo *_tryton_.py* y este fichero XML se nombra como *instaladores.xml*. Se explicará, al igual que se realizó con los modelos, todas las vistas y las acciones en las que interviene la clase Pedido, al tratarse de la clase más compleja.

Finalmente se explicará cómo se crea la plantilla para generar el documento de cada pedido, que será remitido a los proveedores ya que hasta el momento no se ha entrado en detalle de este asunto y resulta ser un aspecto muy importante para completar la funcionalidad del módulo y del sistema completo.

En la siguiente sección de código del archivo *instaladores.xml* se muestra la implementación de la vista árbol realizada para la clase Pedido, junto a sus comentarios para entender sin problemas el desarrollo y posteriormente ser capaz de ampliar está explicación a las demás clases.

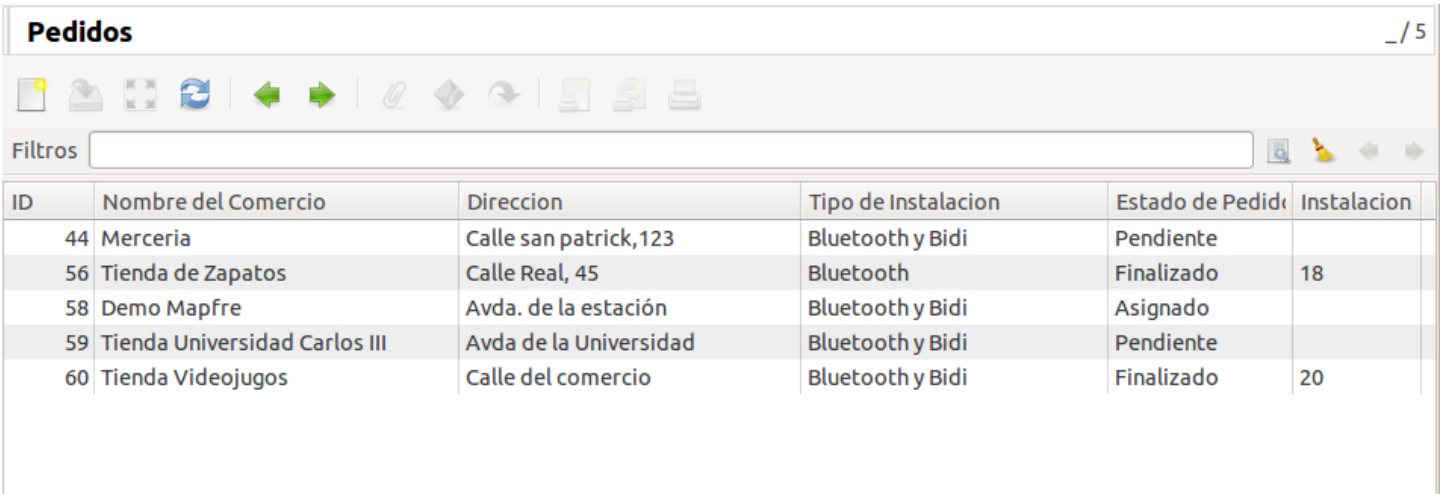
Primeramente se adjunta el código con el que se define la vista, se relaciona con el modelo que se presenta en la vista, el tipo de vista y como se define el contenido, en este caso con XML.

```
<!-- Definición de la vista -->
<record model="ir.ui.view" id="pedidos_view_tree">

      <!—Modelo al que hace referencia la vista-->
      <field name="model">instaladores.pedido</field>

      <!—Definición de tipo de vista, en este caso tipo arbol-->
      <field name="type">tree</field>

      <!—Definición del detalle de la vista mediante xml-->
      <field name="arch" type="xml">
```

Finalmente se incorpora que campos se desea mostrar en la vista y bajo qué orden,

```
<![CDATA[
      <!—Nombre de la vista tipo arbol-->
      <tree string="Pedidos">

          <!—Listado de atributos, cada columna vendrá nombrada por el primer
parámetro en su definición -->
          <field name="id"/>
          <field name="comercio"/>
          <field name="direcciontienda"/>
          <field name="tipo"/>
          <field name="fechapedido"/>
          <field name="estado"/>
          <field name="instalacion_id"/>
      </tree>
]]>
</field>
</record>
```

A continuación se adjunta la Fig 2.22., la pantalla de cómo puede observarse esta vista desde el cliente Tryton con registros ya incluidos en el sistema. El resto de vistas de tipo lista de las demás clases podrán verse en el Anexo II, manual de uso del módulo ERP, de forma que se pueda tener una perspectiva visual más completa de todo el módulo.




| ID | Nombre del Comercio | Direccion | Tipo de Instalacion | Estado de Pedid | Instalacion |
|---|---|---|---|---|---|
| 44 | Merceria | Calle san patrick,123 | Bluetooth y Bidi | Pendiente | |
| 56 | Tienda de Zapatos | Calle Real, 45 | Bluetooth | Finalizado | 18 |
| 58 | Demo Mapfre | Avda. de la estación | Bluetooth y Bidi | Asignado | |
| 59 | Tienda Universidad Carlos III | Avda de la Universidad | Bluetooth y Bidi | Pendiente | |
| 60 | Tienda Videojugos | Calle del comercio | Bluetooth y Bidi | Finalizado | 20 |

*Figura 2.22. Vista formulario de clase Pedido*

En la siguiente sección de código, al igual que se ha realizado con la vista árbol, se muestra la implementación del formulario que detalla un registro ya guardado o que permite la introducción y modificación de nuevos datos en el sistema. Es interesante el destacar que para este tipo de vista es necesario mostrar una descripción de cada campo ya que no se muestra automáticamente como sí se realizaba en la vista anterior desarrollada, de manera que se sepa a que corresponde cada campo y ofrecer mayor sencillez en el módulo. En este caso no se adjunta el código que proporciona la definición de la vista ya que es idéntica al caso de vista tipo árbol sustituyendo donde corresponde el tipo de vista.

Se estructura la vista, agrupando los diferentes campos por funcionalidad y una vez agrupados se maqueta la vista. En la siguiente porción de código se muestra la definición de uno de los grupos, compuesto a su vez de dos entidades, con el objetivo de disponer de una referencia a como se definen los campos a mostrar en la vista de formulario.

```
<!—Seccion 1 para mostrar datos básicos del pedido y la foto del código bidi de
pedido autogenerado-->
<group string="" id="pedido_datos">

      <!—Seccion 1.1 datos básicos de pedido-->
      <group string="" id="instaladores_basico">

            <!—campos y etiquetas a mostras dos filas con dos campos cada una-->
           <label id="idlabel" string="Id. Pedido"/>
            <field name="id"/>
            <label name="fecharecepcion"/>
            <field name="fecharecepcion"/>
            <newline/>
            <label id="idlabelpedido" string="Fecha Pedido"/>
            <field name="create_date"/>
            <label name="estado"/>
            <field name="estado"/>
      </group>

      <!—Seccion 1.2 widget para mostrar la foto del bidi de pedido-->
      <group string="FotoBidi" id="instaladores_photobidi">
            <field name="bidi" widget="image"/>
      </group>
</group>
<newline/>
```

Se adjunta la Fig 2.23., la imagen de cómo se observa esta vista desde el cliente Tryton para el caso de la clase Pedido y un registro ya guardado. El resto de vistas de esta clase también se pueden ver en el Anexo II, manual de uso del módulo ERP.

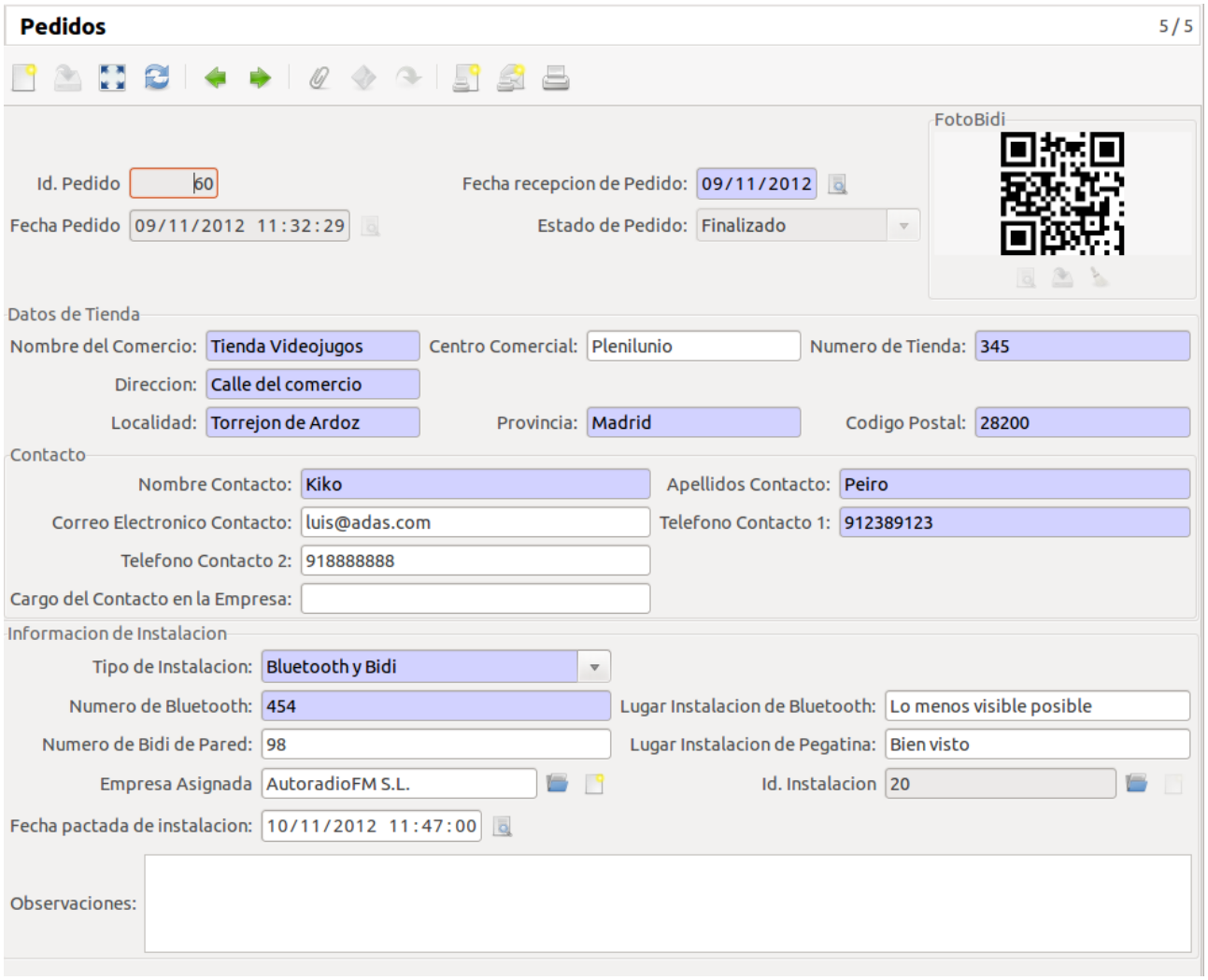



*Figura 2.23. Vista formulario de clase Pedido*

Como se comentó al principio de este apartado, Tryton permite como punto fuerte generar documentos dinámicos basados en plantillas, definiendo por ejemplo en un archivo "*.odt"[13], [27] el formato del documento que se desea y posteriormente generar automáticamente el reporte completo, en PDF, basándose en la información existente en el sistema.

Tryton para esto emplea como motor de reportes un sistema basado en Relatorio [28], proyecto de código abierto que permite la generación de informes de modo sencillo

[13] Open Document Text

en varios formatos desde objetos Python. La manera en la que se define la plantilla que se empleará, es introduciendo una etiqueta que realiza un bucle de tipo “for” para cada elemento que se desee generar el documento resumen del pedido de esta forma

```
<FOR EACH = “PEDIDO IN OBJECTS”> </FOR>
```

Y en el interior de estas líneas todos los campos y la ubicación donde se deseen, además del formato que se decidan. Para este proyecto la empresa facilitó una modelo de documento que se venía utilizando hasta el momento con los proveedores, por lo tanto se tuvo que crear la nueva plantilla de manera que no se notara visualmente el cambio de un documento a otro y de un sistema a otro. En la Fig 2.24. puede verse la plantilla que se creó para que el módulo fuera capaz de generar las hojas de pedido correspondientes.


*Figura 2.24. Plantilla de hoja de pedido*

Una vez ya explicados los aspectos más complejos del fichero “instaladores.xml” en el que se definen las vistas, se adjunta el código eliminando las partes que ya han sido detalladas anteriormente teniendo la fotografía completa de cómo definir las vistas completas de un módulo. En este código puede verse la implementación de las acciones que permiten el lanzamiento de las vistas y la creación del menú en la barra de Tryton que

permite que el módulo sea accesible desde el cliente. Se ha simplificado el código a la vista de tipo árbol ya que es completamente equivalente para la vista de tipo formulario.

```
<!—Creación del enlace al módulo desde el menú Tryton-->
<menuítem name="Gestion de Instalaciones" id="menu_instaladores" sequence="10"/>

<!—Codigo vistas árbol de cada clase-->

<!-- Vista para el menu principal y el evento -->
<record model="ir.action.act_window" id="act_pedidos_form">
    <field name="name">Pedidos</field>
    <field name="res_model">instaladores.pedido</field>
</record>

<!—Acción que conecta el menu con la vista árbol, el valor de sequence refleja
la prioridad a la hora de mostrarse, se puede cambiar de vista si se pulsa el
botón de cambio de vista en la barra de herramientas en el cliente Tryton -->
<record model="ir.action.act_window.view" id="act_pedidos_form_view1">
    <field name="sequence" eval="10"/>
    <field name="view" ref="pedidos_view_tree"/>
    <field name="act_window" ref="act_pedidos_form"/>
</record>


<!—Se añade un fragmento adicional para enlazar las vistas anteriores con el
menu-->
      <menuítem parent="menu_instaladores" sequence="1" id="menu_pedidos_form"
icon="tryton-list" action="act_pedidos_form"/>
```

### *2.4.4 Generación de pedidos*

Para la generación de las instancias del modelo Pedido, las alternativas son dos. La más común será la de generarse automáticamente y la otra alternativa es de forma manual. Para entender a grandes rasgos como se comunican todas las partes que se ven involucradas en el proceso de recepción de peticiones de pedidos se define el siguiente diagrama de la Fig 2.25..

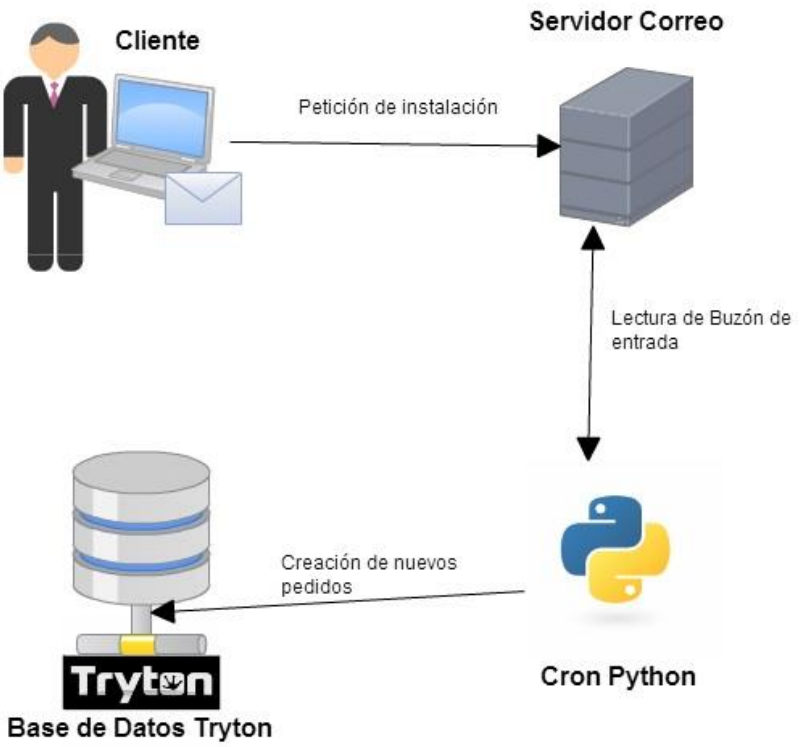



*Figura 2.25. Diagrama de la generación automática de pedido*

Primeramente el cliente debe enviar a una cuenta de correo de la empresa, creada específicamente para la recepción de este tipo de solicitudes, un correo electrónico por instalación con un formato definido y que ya se venía empleando en la empresa antes de comenzar este proyecto. Esta pasarela de comunicación con el cliente viene establecida como una especificación del sistema a pesar de existir otras opciones que podrían resultar más interesantes tales como invocar un *Web Service* o lanzar una petición a un *Servicio Restful* [29], pero asumiendo que las condiciones son las definidas no se entrará más en detalle sobre este asunto.

Estas peticiones llegarán de manera aleatoria y a lo largo de todo el día, por esto se opta por implementar un cron escrito en Python que será ejecutado de manera periódica y que comprobará si existen correos nuevos con el asunto y el formato adecuado. En el caso de que dispongamos de estas condiciones el propio cron se conectará a la base de datos Postgres de Tryton y lanzará las sentencias correspondientes para crear un registro

nuevo para el modelo Pedido, para el desarrollo de esta prueba de concepto se ha creado una cuenta de correo en “Gmail” y se empleará para realizar la demostración.

A continuación se muestran Fig 2.26 y Fig 2.27., donde se pueden analizar los diagramas de flujo del cron que se está tratando.

Para la creación del código Bidi se emplea el siguiente código que será también el que se utilice en el propio módulo en el caso de que alguno de los datos guardados deban ser modificados o incluso dar de alta un pedido nuevo de forma manual. Se deben importar las librerías *qrencode* [30] y *psycopg2* [31] para generar la imagen y para poder guardarla como binario en la base de datos. A modo de recordatorio, el contenido de este código Bidi contiene la cadena compuesta por el prefijo “5469656E6461” seguido por los identificadores, propios del cliente, del dispositivo Bluetooth y la pegatina promocional.

```
#Se establece el tamaño de la imagen
size = 250

#Si no disponemos del dato idbidi significa que el tipo de instalacion es del
tipo 1. Y debe figurar como idbidi=0
if idbidi == '':
      bidi='0'

#Se obtiene la fecha actual con el format %Y%m%d
hoy = datetime.datetime.now()
hoy=hoy.strftime("%Y%m%d")

#Se genera la imagen
version, src_size, im = encode("5469656E6461|"+bluetooth+"|"+bidi+"|"+hoy)
im = im.resize((size, size), Image.NEAREST)
pad = 0
ret = Image.new("L", (size, size), 255)
ret.paste(im, (pad, pad))

#Se obtiene los datos de la imagen para poder insertarla en base de datos
binary = StringIO.StringIO()
ret.save(binary,format="PNG")
contents = binary.getvalue()
binary.close()
```

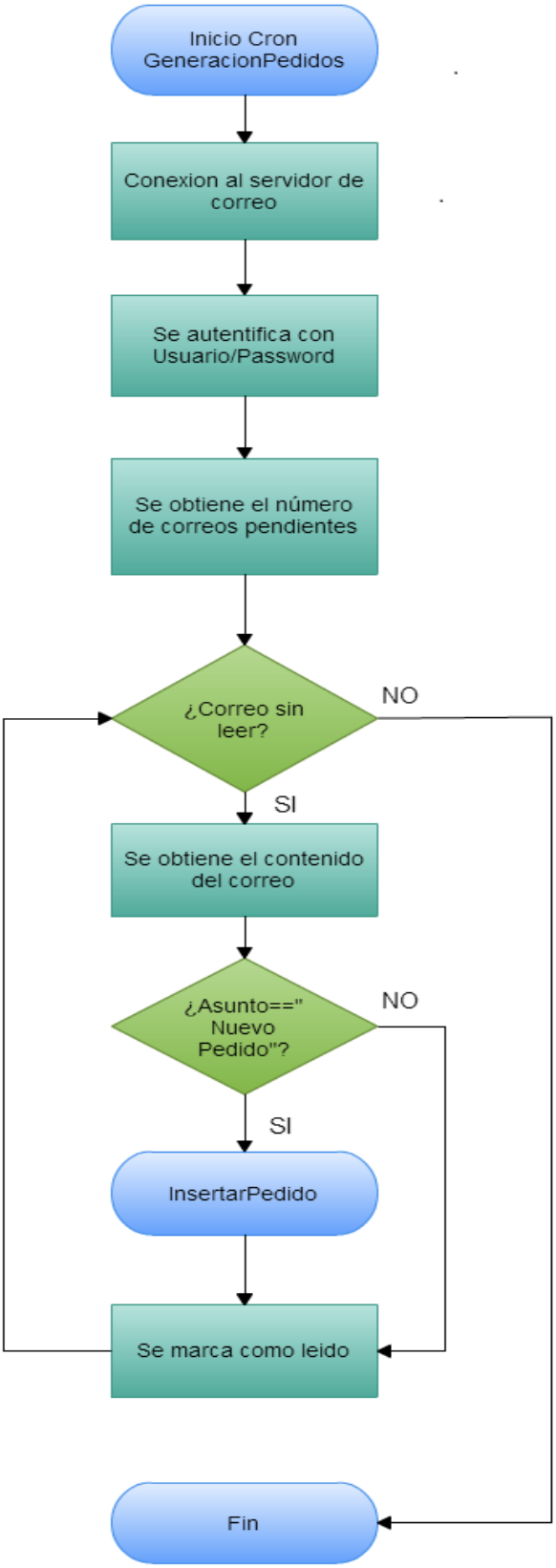



*Figura 2.26. Diagrama de flujo de lectura de bandeja de entrada de correo*

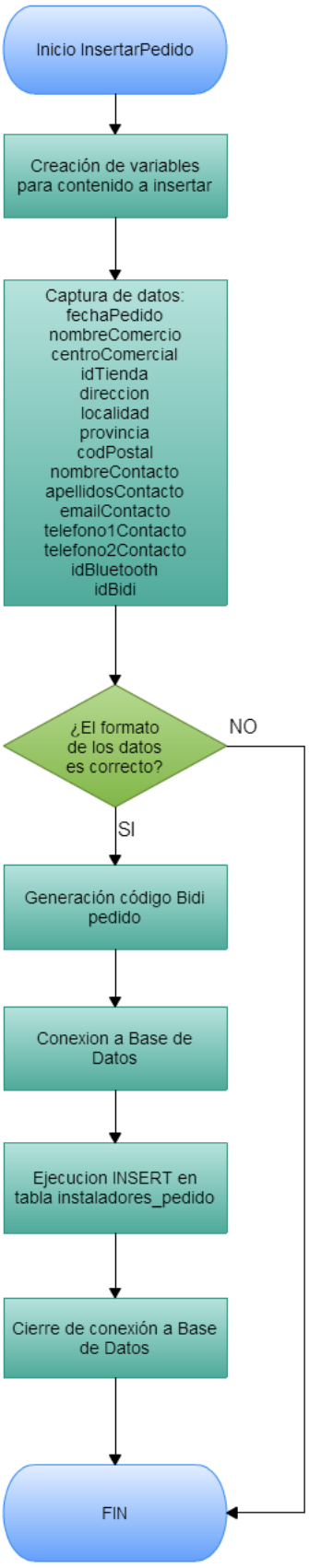



*Figura 2.27. Diagrama de flujo de creación de pedido*

# 3. Aplicación Android para instaladores

Los terminales móviles se han convertido en partes cruciales de las vidas cotidianas de todas las personas, tanto en el entorno doméstico como en el empresarial. Todo el mundo, desde adolescentes hasta ancianos, tiene un teléfono móvil. Sin embargo, los móviles que existieron hace algunos años, grandes y pesados pensados únicamente para hablar por teléfono no se parecen en nada a los actuales donde el término de comunicación se ha quedado bastante ligero.

Android se ha posicionado en pocos años como el rey indiscutible en dispositivos móviles inteligentes, ya sean *smartphones* o *tablets*. En algunos países este sistema operativo ha barrido del mercado a otros que llevaban tiempo siendo los usados habitualmente por los usuarios y también ha superado con relativa facilidad a otras plataformas con más repercusión mediática y que hasta la llegada de Android era considerada como el culmen en sistemas operativos móviles, se habla de Apple y su sistema IOS [32].

Android fue desarrollado por Android Inc., empresa que en 2005 fue comprada por Google, aunque no fue hasta 2008 cuando se popularizó, gracias a la unión al proyecto de Open Handset Alliance [33], un consorcio formado por 48 empresas de desarrollo hardware, software y telecomunicaciones, que decidieron promover el software libre. Pero ha sido Google quien ha publicado la mayor parte del código fuente del sistema operativo.

Dado que Android está basado en el núcleo de Linux, tiene acceso a sus recursos, pudiendo gestionarlo gracias a que se encuentra en una capa por encima del Kernel, núcleo del sistema operativo, accediendo así a recursos como los controladores de pantalla, la cámara, el flash o al Bluetooth entre otros.

En la Fig 3.1. se muestran las capas que conforman el sistema operativo Android.


*Figura 3.1. Arquitectura Android*

Es importante conocer cómo está estructurada la arquitectura de este sistema operativo. En el caso de Android, está formada por varias capas que facilitan al desarrollador la creación de aplicaciones. Además esta distribución permite acceder a las capas más bajas mediante el uso de librerías para que él desarrollador no tenga que programar a bajo nivel las funcionalidades necesarias para que una aplicación haga uso de los componentes hardware de los teléfonos. A continuación se explican cada una de las capas empezando por la base de la pila.

## Kernel de Linux

Actualmente el núcleo del sistema operativo Android está basado en el kernel de Linux versión 3.0, hablando de la versión de Android JellyBean, para el caso de versiones anteriores se basaban en la versión de kernel 2.6, similar al que puede incluir cualquier distribución de Linux, como Ubuntu, solo que adaptado a las características del hardware en el que se ejecutará Android, es decir, para dispositivos móviles.

El núcleo actúa como una capa de abstracción entre el hardware y el resto de las capas de la arquitectura. El desarrollador no accede directamente a esta capa, sino que debe utilizar las librerías disponibles en capas superiores. De esta forma también se evita el hecho de conocer las características precisas de cada teléfono. Si se necesita hacer uso de la cámara, el sistema operativo se encarga de utilizar la librería que incluya el teléfono, sea cual sea. Para cada elemento de hardware del teléfono existe un controlador dentro del propio kernel que permite utilizarlo desde el propio software.

El kernel también se encarga de gestionar los diferentes recursos del teléfono tales como la batería y la memoria, y del sistema operativo en sí como pueden ser los procesos o los elementos de comunicación.

## Librerías

La siguiente capa que se sitúa justo sobre el kernel la componen las bibliotecas nativas de Android, también llamadas librerías. Están escritas en lenguajes C y/o C++ y se encuentran compiladas para la arquitectura hardware específica del teléfono. Estas normalmente están realizadas por el fabricante, quien también se encarga de instalarlas

en el dispositivo antes de ponerlo a la venta. El objetivo de las librerías es proporcionar funcionalidad a las aplicaciones para tareas que se repiten con frecuencia, evitando tener que codificarlas cada vez y garantizando que se llevan a cabo de la forma más eficiente.

Entre las librerías incluidas habitualmente se encuentran típicamente OpenGL (motor gráfico) [34], bibliotecas multimedia (formatos de audio, imagen y video), Webkit (navegador), SSL[14] (cifrado de comunicaciones) [35], FreeType (fuentes de texto) [36] y SQLite (base de datos).

**Entorno de ejecución**

Como se puede apreciar en el diagrama expuesto previamente, el entorno de ejecución de Android no se considera una capa en sí misma, dado que también está formado por librerías. Aquí encontramos las librerías con las funcionalidades habituales de Java así como otras específicas de Android.

El componente principal del entorno de ejecución de Android es la máquina virtual Dalvik. Las aplicaciones se codifican en Java y son compiladas en un formato específico para que esta máquina virtual las ejecute. La ventaja de esto es que las aplicaciones se compilan una única vez y de esta forma estarán listas para distribuirse con la total garantía de que podrán ejecutarse en cualquier dispositivo Android que disponga de la versión mínima del sistema operativo que requiera la aplicación.

Cabe aclarar que Dalvik es una variación de la máquina virtual de Java, por lo que no es compatible con el bytecode Java, código intermedio entre el código fuente y el código máquina. Java se usa únicamente como lenguaje de programación, y los ejecutables que

[14] Secure Socket Layer

se generan con el SDK de Android tienen la extensión “.dex” que es específico para Dalvik, y por ello no podemos correr aplicaciones Java en Android ni viceversa.

**Framework de aplicaciones**

La siguiente capa está formada por todas las clases y servicios que utilizan directamente las aplicaciones para realizar sus funciones. La mayoría de los componentes de esta capa son librerías Java que acceden a los recursos de las capas anteriores a través de la máquina virtual Dalvik. Entre ella se encuentran:

*Activity Manager*: Se encarga de administrar la pila de actividades de una aplicación así como su ciclo de vida.

*Windows Manager*: Se encarga de organizar lo que se mostrará por pantalla. Básicamente crea las superficies en la pantalla que posteriormente pasarán a ser ocupadas por las actividades.

*Content Provider*: Esta librería se emplea para crear una capa que encapsula los datos que se compartirán entre aplicaciones de forma que se obtenga un control sobre cómo se accede a la información.

*Views*: En Android, las vistas conforman los elementos que nos ayudarán a construir las interfaces de usuario: botones, cuadros de texto, listas y hasta elementos más avanzados como un navegador web o un visor de Google Maps.

*Notification Manager*: Engloba los servicios para notificar al usuario cuando algo requiera su atención mostrando alertas en la barra de estado. Un dato importante es que esta biblioteca también permite jugar con sonidos, activar el vibrador o utilizar los LEDs[15] del teléfono en caso de tenerlos.

*Package Manager*: Esta biblioteca permite obtener información sobre los paquetes instalados en el dispositivo Android, además de gestionar la instalación de nuevos paquetes. Con el término paquete se hace referencia a la forma en que se distribuyen las aplicaciones Android, estos contienen el archivo ".apk", que a su vez incluyen los archivos ".dex" con todos los recursos y archivos adicionales que necesite la aplicación, para facilitar su descarga e instalación.

*Telephony Manager*: Con esta librería se permite realizar llamadas o enviar y recibir SMS[16]/MMS[17], aunque no admite reemplazar o eliminar la actividad que se muestra cuando una llamada está en curso.

*Resource Manager*: Con esta librería se pueden gestionar todos los elementos que forman parte de la aplicación y que están fuera del código, es decir, cadenas de texto traducidas a diferentes idiomas, imágenes, sonidos o layouts.

*Location Manager*: Permite determinar la posición geográfica del dispositivo Android mediante GPS[18] o redes disponibles y trabajar con mapas.

*Sensor Manager*: Gracias a este componente se permite manipular los elementos de hardware del teléfono como el acelerómetro, giroscopio, sensor de luminosidad, sensor de

[15] Light Emmiting Diode
[16] Short Message Service
[17] Multimedia Messaging System
[18] Global Positioning System

campo magnético, brújula, sensor de presión, sensor de proximidad y al sensor de temperatura en el caso de que el teléfono disponga de ello.

*Cámara*: Con esta librería se permite hacer uso de la cámara o cámaras del dispositivo para tomar fotografías o para grabar vídeo.

*Multimedia*: Permiten reproducir y visualizar audio, vídeo e imágenes en el dispositivo.

**Aplicaciones**

En la última capa se incluyen todas las aplicaciones del dispositivo, tanto las que tienen interfaz de usuario como las que no, las nativas (programadas en C o C++) y las administradas (programadas en Java), las que vienen preinstaladas en el dispositivo y aquellas que el usuario ha instalado.

En esta capa se encuentra también la aplicación principal del sistema el Home o el Launcher porque es la que permite ejecutar otras aplicaciones mediante una lista y mostrando diferentes escritorios donde se pueden colocar accesos directos a aplicaciones o incluso widgets, que son también aplicaciones de esta capa.

Como podemos ver, Android nos proporciona un entorno sumamente poderoso para que podamos programar aplicaciones que hagan cualquier cosa. Nada dentro de Android es inaccesible y podemos jugar siempre con las aplicaciones de nuestro teléfono para optimizar cualquier tarea.

El potencial de Android se sitúa en el control total que se le da al usuario para que haga de su teléfono un dispositivo a su medida.

# 3.1 Entorno Android

En esta sección se comenta a modo de referencia algunos conceptos básicos relacionados con el entorno de desarrollo de Android; una descripción de los componentes que se requieren para poder desarrollar código, testear y depurar aplicaciones, la estructura de un proyecto de estas características y algunos conceptos básicos de los elementos que intervienen en cualquier aplicación Android.

Para el desarrollo de las aplicaciones se va a utilizar un potente y moderno entorno de desarrollo. Al igual que Android, todas las herramientas están basadas en software libre, aunque existen varias alternativas para desarrollar aplicaciones, en este proyecto se emplean los siguientes componentes que se detallan a continuación.

**Java Runtime Enviroment 6.0**

Gracias a este software se va a permitir ejecutar código Java en el equipo de trabajo. A la máquina virtual Java también se le conoce como entorno de ejecución Java, JRE[19] o JVM[20].

[19] Java Runtime Enviroment
[20] Java Virtual Machine

**Eclipse (Eclipse IDE[21] for Java Developers)**

Eclipse ha resultado ser el entorno de desarrollo más recomendable para Android ya que es libre y además es soportado por Google, ya que ha sido empleado por los desarrolladores de Android para crear el propio Android.

**Android SDK[22] (Google)**

Este software corresponde al kit de desarrollo de Google para su sistema operativo, y gracias a él se puede virtualizar de una forma sencilla un sistema operativo Android en un PC cualquiera. Comprende un depurador de código, biblioteca, un simulador de un terminal, documentación, ejemplos de código y tutoriales.

**Eclipse plug-in (Android Development Toolkit,ADT)**

Este software disponible para Eclipse, creado por Google, instala una serie de complementos, de forma que el entorno de desarrollo se adapte al desarrollo de aplicaciones para Android ampliando las capacidades de Eclipse para la puesta en marcha de nuevos proyectos, proporcionando una asombrosa sencillez en la creación de nuevos componentes basados en la API[23] de Android y sobre todo permitirá crear la aplicación a partir del código fuente y sus recursos entre otras funcionalidades.

[21] Integrated Development Enviroment
[22] Software Development Kit
[23] Application Programming interface

Para diseñar una aplicación en Android, es necesario tener claros los elementos que la componen y la funcionalidad de cada uno de ellos. Android trabaja en Linux [37], y cada aplicación utiliza un proceso propio. Se distinguen por el ID, un identificador que obliga a que solo este proceso pueda tener acceso a sus recursos. Los dispositivos habitualmente tienen un único foco, la ejecución principal, que se puede reconocer por ser la aplicación que está siendo mostrada por pantalla, pero es normal que se encuentren corriendo varias aplicaciones en un segundo plano, cada una con su propia pila de tareas.

Esta pila corresponde a la secuencia de ejecución de procesos en Android y están compuestos de actividades que se van apilando según son invocadas, y solo pueden terminarse cuando las tareas que tienen encima se terminan, o cuando son destruidas por falta de recursos en el sistema. El sistema siempre elimina la actividad que lleve más tiempo inactiva y por ejemplo en el caso de ser necesaria mucha memoria, si la aplicación no está en el foco, puede ser eliminada por completo exceptuando su actividad principal.

Otra de las características principales del diseño en Android consiste en la reutilización de componentes entre aplicaciones, es decir dos aplicaciones diferentes podrían compartir un mismo componente, de modo que se consiga evitar la repetición de código y su correspondiente ocupación en memoria. Los componentes son los elementos básicos con los que se construyen los proyectos Android pero son realmente las actividades las que componen la aplicación, habrá una actividad por cada pantalla distinta que tenga la aplicación y de hecho los componentes por si solos no pueden hacer funcionar una aplicación necesitan de la ayuda de los *Intents*, pero por ahora no se entrará en más detalle.

### *3.1.1 Estructura de un proyecto Android*

Un proyecto Android está formado principalmente por un descriptor de la aplicación, es decir, un fichero que se denomina “AndroidManifest.xml” y que más adelante se explica, el

código fuente y una serie de ficheros con recursos. Cada uno de estos elementos se almacena en una carpeta específica y para explicar la forma en que se organizan los proyectos se va a explicar los puntos claves que son necesarios entender previamente empleando para ello la estructura del clásico “Hola Mundo” presente en la Fig 3.2..

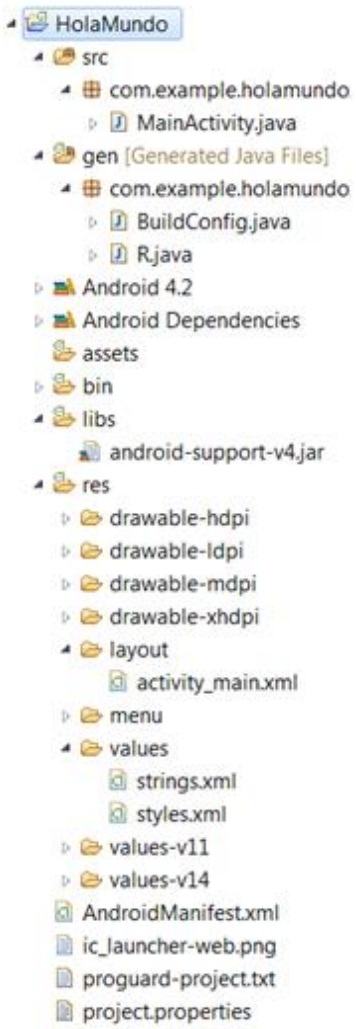



*Figura 3.2. Estructura de carpetas de proyecto Android*

**Carpeta src**

Esta es por defecto la carpeta donde se deposita el código fuente Java. Todo el código que se ubique en la carpeta “src” será compilado cuando se requiera. Y también como en

los proyectos tradicionales de Java, el código se organiza en carpetas que resultan ser paquetes. Es importante recordar que durante el proceso de creación del proyecto se debe definir el paquete que lo identificará, generalmente la estructura del proyecto partirá de esta raíz como puede verse.

**Carpeta res**

La carpeta "res" puede resultar la más compleja del proyecto debido a los subdirectorios que existen en su interior, pero para resumir se dirá que contienen los recursos usados por la aplicación como imágenes, layouts, textos y que se van a gestionar estos recursos para diferentes dispositivos con características y configuraciones distintas. Sus subcarpetas pueden tener un sufijo para el caso de que únicamente interese cargar sus recursos en el caso de que se cumpla una condición en particular. A modo de ejemplo el sufijo "–hdpi" significa que sólo se han de cargar los recursos en el caso de que el dispositivo en el que se ejecuta la aplicación dispone de una densidad gráfica alta, en este caso mayor a 180 puntos por pulgada, y el sufijo -14 corresponde a la condición de requerir un dispositivo con un nivel de API de 14.

Las posibles subcarpetas sin sufijos se listan a continuación:

*drawable*: en esta carpeta se almacenan los ficheros de imágenes (JPEG[24] o PNG[25]) e incluso descriptores de imágenes en XML.

*layout*: contiene ficheros XML con vistas de la aplicación. Las vistas se emplean para configurar las diferentes pantallas que compondrán la interfaz de usuario de la aplicación.

[24] Joint Photographic Experts Group
[25] Portable Network Graphics

*menu*: fichero XML con los menús de cada actividad.

*values*: también se emplea ficheros XML para indicar valores de tipo *texto*, color o estilo. De esta manera se permite cambiar los valores sin necesidad de recurrir al código fuente y facilitar por ejemplo una traducción de idioma.

*anim*: contiene ficheros XML con animaciones “Tween”, las cuales permiten realizar a algunos componentes una serie de transformaciones sencillas definidas, como cambiar de posición, cambiar de tamaño o de transparencia por ejemplo, en el momento y con la duración que se indique.

*animator*: contiene ficheros XML con animaciones de propiedades, que consiste en animaciones pero no aplicables en exclusiva a componentes de tipo View como sucedía con las animaciones “Tween” y al finalizar la animación la visualización del componente se mantiene al terminar, mientras que con una “Tween” vuelve al estado inicial.

*xml*: otros fichero XML requeridos por la aplicación.

*raw*: ficheros adicionales que no se encuentran en formato XML.

**Carpeta gen**

Esta carpeta contiene código fuente como la carpeta “src”, pero no se debe agregar o modificar los archivos que contiene ya que se generan automáticamente por el plugin

(complemento de una aplicación) de Android a partir de la carpeta “res”. Dentro se encuentra:

*Buildconfig.java*: Define la constante “DEBUG” para que desde Java se pueda conocer si la aplicación está en fase de desarrollo

*R.java*: Este archivo java se emplea para indexar e identificar a todos y cada uno de los recursos de la carpeta “res”, es decir asocia a los recursos con identificadores únicos y de esta forma los recursos podrán ser accedidos desde Java.

**Carpeta bin**

El interior de esta carpeta, que se genera automáticamente como la carpeta “gen”, es utilizada por el compilador para preparar los archivos para el empaquetado final en forma de APK[26]. Esta carpeta se puede borrar ya que nuevamente se generará al recompilar por lo tanto no es necesario en el caso de que se trabaje con un sistema de repositorio incluirla en la sincronización evitando así almacenar más memoria de la necesaria.

**Carpeta libs**

En esta carpeta se guardan las librerías que se necesitan enlazar para usar en el proyecto, estas librerías se incluyen en formato JAR[27] habitualmente.

[26] Application PacKage File
[27] Java Archive

**Carpeta Android x.x**

Aquí puede encontrarse el código en formato JAR del API de Android que se ha seleccionado para el proyecto.

**Android Dependencies**

En el interior de esta carpeta se podrán observar las librerías asociadas al proyecto.

**Carpeta assets**

Esta carpeta puede contener una serie arbitraria de ficheros o carpetas que pueden ser empleados por la aplicación. También serán empaquetados en el APK final y pueden ser considerados como recursos, al igual que el contenido de la carpeta res con la diferencia de que el contenido de la carpeta en este caso no será indexado ni compilado de modo que el acceso a estos recursos es menos eficiente. Destacar que para el contenido de esta carpeta no es necesario aplicar la política de nomenclatura y distribución explicada anteriormente para la carpeta res.

**Fichero AndroidManifest.xml**

Con este fichero se describe la aplicación Android, se puede decir que es el archivo principal y que todas las aplicaciones o librerías deben contenerlo en la raíz del proyecto. En él se definen las características del proyecto como el nombre, paquete o los permisos

que va a requerir la aplicación. También se declaran las actividades, Intents, servicios y proveedores de contenido de la aplicación

**Fichero ic_launcher-web.png**

Esta imagen representa el icono de la aplicación en gran tamaño, para ser empleado por una página Web. El nombre puede diferir si se indicó un nombre distinto en el momento de la creación del proyecto. Debe tener una resolución de 512x512 con capa alfa (capa que define la opacidad de la imagen) incluida.

**Fichero proguard-project.txt**

Este fichero configura la herramienta ProGuard [38], con la que se optimiza y ofusca el código que se ha generado. Simplemente se genera un APK más compacto y en con el que hacer ingeniería inversa es más complejo.

**Fichero default.properties**

Este es un fichero que se genera automáticamente por el SDK. No es recomendable modificarlo y es empleado para realizar comprobaciones cuando se instala la aplicación en el terminal como por ejemplo la versión del API.

### *3.1.2 Componentes de una aplicación Android*

Existen una serie de elementos clave que resultan imprescindibles para desarrollar aplicaciones en Android. A lo largo de este capítulo se realiza una descripción de algunos de los más importantes, que se emplean para este proyecto y que conviene destacar.

**Vistas (Views)**

Las vistas representan los elementos que componen la interfaz de usuario de cualquier aplicación. Todos los objetos son descendientes de la clase “View” y así se pueden definir con código Java. Sin embargo lo más común es emplear código XML y delegar la tarea de creación de dichos objetos al sistema a partir de estos ficheros. Entre otros muchos se encuentran los elementos que pueden observarse en la Fig 3.3..

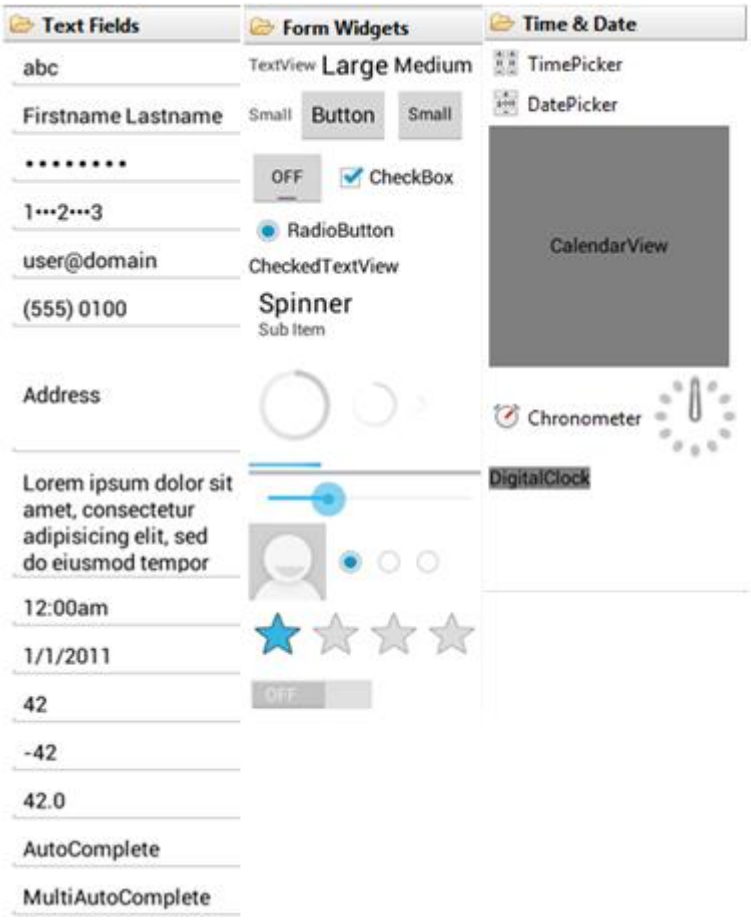



*Figura 3.3. Vistas básicas Android*

Todos los elementos se pueden encontrar en la sección *Graphical Layouts*, dentro del entorno de desarrollo, al abrir un fichero XML creado para diseñar un Layout.

## Layout

Un layout se compone por un conjunto de vistas agrupadas de una forma en particular. Se dispone de varios modelos de layouts para organizar las vistas de forma lineal, en cuadricula o indicando sus posiciones absolutas. Los layouts también son descendientes

de la clase “View" y al igual que para las vistas también es posible la definición de layouts en código Java pero lo más recomendado es realizarlo también con código XML.

Es destacable que un layout puede a su vez contener otros layouts a medida que se forma la interfaz. A continuación se listan los modelos de layout más comunes en una aplicación Android:

*Linear Layout* (Fig 3.4.): Es el modelo más sencillo de todos, dispone los elementos uno tras otro y le podemos indicar si la orientación debe ser horizontal o vertical.

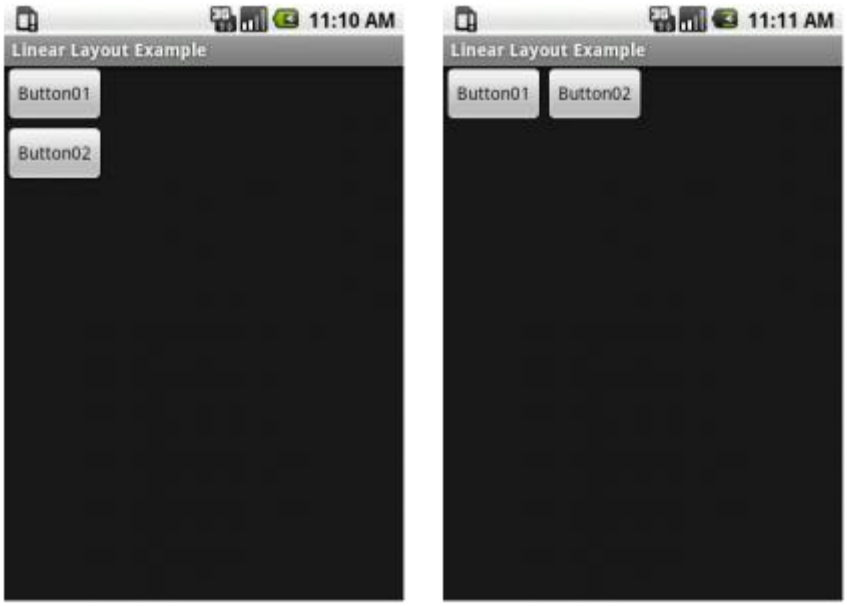



*Figura 3.4. Ejemplo Lineal Layout*

A continuación se adjunta la porción de código necesaria para obtener el layout vertical. Para obtener el de orientación horizontal únicamente es necesario declararlo en el atributo *orientation* con el valor “horizontal”.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical" android:layout_width="fill_parent"
android:layout_height="fill_parent">
```

```
      <Button android:text="Button01" android:id="@+id/Button01"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" />

      <Button android:text="Button02" android:id="@+id/Button02"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" />

</LinearLayout>
```

En esta porción de código se pueden observan dos atributos muy empleados tanto para los layouts como para los elementos, en este caso botones. Son *layout_width* y *layout_height* que se declaran con el fin de determinar el tamaño de los componentes a lo largo del ancho de la pantalla y el largo en función del ancho y el largo del contenedor padre (“fill_parent”) o del contenido (“wrap_content”). En este caso el layout es definido para que ocupe toda la pantalla del terminal y en los botones se restringe su tamaño al contenido en su interior.

*Table Layout* (Fig 3.5.): Representa los elementos en forma de tabla, se indica los saltos de fila y siguiendo el orden los elementos son colocados en la fila en la que estén definidos.


*Figura 3.5. Ejemplo Table Layout*

*Relative Layout* (Fig 3.6.): Este modelo representa el layout más flexible de todos los básicos que se definen en Android ya que permite coloca los elementos en relación a otros elementos o a otro layout.


*Figura 3.6. Ejemplo Relative Layout*

*Absolute Layout*: En este modelo de layout se definen las posiciones de los elementos de manera absoluta, esta forma de ordenación de los componentes no es recomendable ya que la gran variedad de dispositivos puede hacer que existan grandes diferencias de visualización de unos a otros.

*Frame Layout*: Permite el cambio dinámico de los elementos que contiene la vista en una misma posición ya que permite el solapamiento de componentes. Todos los componentes definidos en este Layout se posición en la esquina superior izquierda. Con el atributo *visibility* se permite el hacer visible uno y no el resto.

Además de los modelos de layout ya descritos existen otros muchos modelos, algo más complejos, que se pueden utilizar pero no se entrará en este asunto ya que en la implementación del proyecto no se utilizan y no se ha considerado oportuno.

**Actividad (Activity)**

Las aplicaciones en Android como ya se ha adelantado anteriormente están compuestas por un conjunto de elementos de visualización, es decir pantallas de la aplicación. Cada una de estas pantallas se denomina actividad. Su función es la de crear la interfaz de usuario, normalmente una aplicación presenta varias actividades para crear esta interfaz. Las diferentes actividades trabajan de manera independiente aunque entre ellas formarán la lógica de la aplicación. Cada una de las actividades definidas son descendientes de la clase “Activity”.

El concepto de actividad en Android representa una pantalla de la aplicación con interactividad del usuario. Una aplicación se forma habitualmente con un conjunto de actividades y el usuario navega de una a otra. Un ejemplo clásico es el botón “atrás” que permite volver a la actividad anterior, la actividad que creó a la actual.

Toda actividad debe disponer una vista asociada que se emplea como interfaz de usuario, normalmente será un layout aunque no es imprescindible. Los pasos necesarios para añadir una nueva actividad a una aplicación son los siguientes:

- Crear un nuevo Layout para la nueva actividad.

- Implementar una clase descendiente de “Activity”, indicándole que la vista a visualizar es el nuevo Layout creado en el punto anterior.

- Activar dicha actividad a partir de otra actividad.

- Registrar la actividad dentro del fichero *AndroidManifest.xml*.

Es muy importe tener siempre en mente el ciclo de vida de las actividades. Las actividades se ejecutan de forma similar a como lo haría una tarea. Cuando una actividad nueva es iniciada se coloca al inicio de la pila de ejecución y se convierte en la actividad en ejecución mientras que la actividad que estuviera ejecutándose anteriormente nunca volverá al primer plano mientras la nueva actividad exista. En la Fig 3.7 se puede observar el ciclo de vida de cualquier “Activity” Android

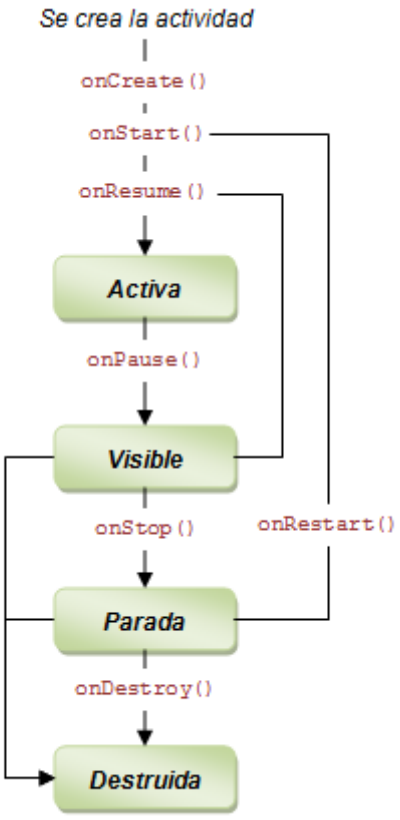



*Figura 3.7. Ciclo de vida de Activity Android*

## Intent

Un “Intent” representa la voluntad de realizar una acción como por ejemplo iniciar una llamada de teléfono o visualizar una página Web. Los Intent se emplean cada vez que se

requiere: iniciar una nueva actividad o servicio, iniciar un anuncio Broadcast, lanzar alguna comunicación con un servicio o transmitir información entre componentes.

Cuando una actividad ha de lanzar otra actividad en multitud de casos es necesario enviar cierta información. Para ello Android ofrece el siguiente mecanismo, a continuación se muestra un ejemplo del lanzamiento de una nueva actividad con envío de información desde una actividad lanzadora.

```
Intent intent = new Intent(this, MI_CLASE.class);
intent.putExtra("info1", "Informacion1");
intent.putExtra("info2",2);
startActivity(intent);
```

Y en el lado de la actividad lanzada se recibe esta información del siguiente modo:

```
Bundle extras = getIntent().getExtras();
 String s = extras.getString("info1");
 int i = extras.getInt("info2");
```

**Receptor de anuncios (Broadcast receiver)**

Este receptor recibe y reacciona frente a anuncios de tipo Broadcast. Por ejemplo, se puede considerar un anuncio de batería baja o una llamada entrante y en la aplicación pueden estar definidas acciones que serán ejecutadas al recibir alguno de los eventos esperados.

**Servicio (Service)**

Un servicio en Android representa un proceso que corre en el fondo de la aplicación, sin necesidad alguna de recibir interacciones de usuario. En Android se pueden encontrar dos tipos de servicios: locales, donde puede ser utilizado por varias aplicaciones desde el mismo terminal, o remotos en los que pueden ser utilizados desde otros terminales.

**Proveedores de contenido (Content Provider)**

En Android se ha definido un mecanismo con el que las aplicaciones puedan compartir datos sin necesidad de comprometer y vulnerar la seguridad del sistema de ficheros. Con esta funcionalidad se permite acceder a información de otras aplicaciones o enviar información a otras.

## 3.2 Diseño de la aplicación

Una vez descritos algunos aspectos fundamentales de los componentes básicos de un proyecto Android, a continuación se describen los requerimientos impuestos por la empresa y sus expectativas, para la parte de la aplicación móvil a desarrollar y que constituyen los objetivos que no pueden ser modificados a lo largo de la ejecución del proyecto y que deben ser satisfechos plenamente.

Una vez que se dispongan de los requisitos se pasa a detallar el proceso de diseño de las actividades que conforman la lógica de la aplicación y todos los layouts y vistas que constituyen la parte visible de la aplicación.

### *3.2.1 Requisitos*

Para esta prueba de concepto se requieren implementar una aplicación para terminales basados en el sistema operativo Android con las siguientes características:

- Nombre de la aplicación: InfortrexInstaladores
- Nombre del proyecto: InfortrexInstaladores
- Nombre del paquete: es.infortrex.instaladores
- Mínimo SDK requerido: API 9, Android 2.3 (GingerBread)
- Terminales móviles con posibilidad de ser visualizado correctamente en dispositivos con formato de tableta.

En lo referente a la funcionalidad que se demanda se adjuntan los requisitos previos ofrecidos por la empresa para la realización de la aplicación, acompañados de un diagrama funcional descriptivo, Fig 3.8., en el que se deberá basar la implementación de la aplicación Android.

- Servicio de autenticación de usuario ya que gracias a ese proceso de logado se registrarán las instalaciones al usuario correspondiente así como la posibilidad de visualizar las instalaciones ya realizadas y de bloquear a usuarios.

- Posibilidad de realizar el registro de cualquiera de los dos tipos de instalaciones ya explicadas.

    - Para la instalación de tipo Bluetooth, el personal deberá primeramente escanear el código Bidi presente en la hoja de pedido donde, en cuyo contenido están reflejados los identificadores del cliente sobre los dispositivos que debe ubicar, posteriormente deberá adjuntar una fotografía de la tarea que haya realizado en el comercio objetivo, enlazarse vía Bluetooth al dispositivo que acaba de instalar y escanear el código Bidi que se encuentra adherido al dispositivo a instalar. Es importante tener en cuenta que estos pasos se deben realizar para cada instalación obligatoriamente.

    - Para la instalación de tipo Bluetooth con pegatina promocional, el personal deberá cumplir todos los pasos descritos en el tipo de instalación anterior y además deberá escanear un nuevo código Bidi, este será el que se encuentre en la pegatina promocional.

- Una vez realizado todos estos pasos se debe permitir revisar la información capturada, visualizar la fotografía tomada, posponer la instalación para otro momento en caso de que no se disponga de cobertura en el local y enviar los datos al sistema de Infortrex.

- Por otro lado, se debe dar la posibilidad de revisar todas las instalaciones ya recibidas por Infortrex realizadas bajo el mismo identificador de empresa, con el fin de que el instalador puede asegurarse de que el registro se ha realizado correctamente; y también ofrecer un acceso directo que permita poder contactar vía llamada telefónica con personal de Infortrex en caso de algún problema con el funcionamiento de la instalación.

- En caso de que una instalación no se haya podido realizar, debido a problemas de conexión, la información capturada se debe quedar almacenada en la memoria del terminal y se debe lanzar periódica y automáticamente peticiones de registro contra el sistema de Infortrex hasta que se consiga registrar la instalación, notificando al usuario cuando haya sido finalizado el registro.

- Las comunicaciones con el sistema de Infortrex deberán realizarse vía conexiones HTTP, basándose en servicios RESTFUL y los resultados se recibirán con formato JSON[28] [39], bajo una implementación en Java. El único requisito estricto para la programación del back-end de la aplicación es que se deberá recibir las peticiones sobre un mismo punto, un Servlet que redirigirá las acciones a los servicios que correspondan en cada caso. De modo que sólo es posible acceder desde el exterior mediante el paso por ese Servlet.

- Es imprescindible que el código se encuentre debidamente comentado, para ahorrar tiempo ante futuros cambios o frente a errores encontrados.

[28] JavaScript Object Notation

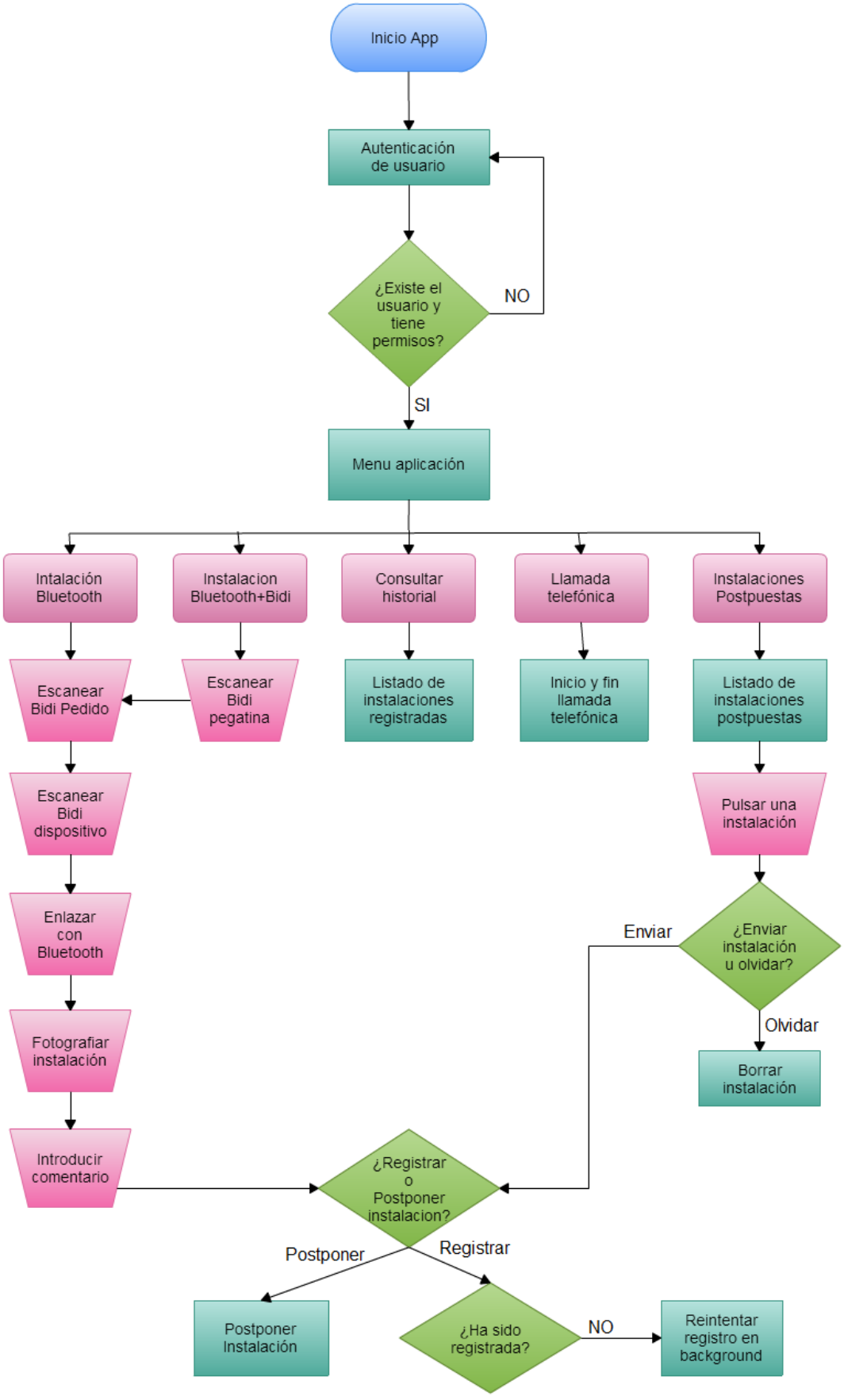



*Figura 3.8. Diagrama funcional de la aplicación Android*

### *3.2.2 Descripción de tareas*

Trabajando en base a los requisitos entregados por la empresa y en una serie de tareas que a continuación se detallan, se obtendrá el resultado que posteriormente se describe.

1. Fase de análisis de requisitos

Basando la creación del proyecto de Android en los requisitos recibidos, estos se analizan y se utiliza la misma sesión inicial que se empleó para la sección del proyecto que empleaba Tryton, para repasar los conceptos e incluso profundizar en algunas ideas importantes que requieren de aclaración de las especificaciones ofrecidas por la empresa y que se han expuesto anteriormente.

En este proyecto realmente no existen unos requisitos realmente estrictos en cuanto a diseño de interfaz de usuario, sino que se ofrece una gran libertad mientras que no se vea entorpecida la funcionalidad ya comentada.

En cuanto a la funcionalidad, se prevé que a medida se vayan generando los diferentes logros estos se muestren al responsable del proyecto para obtener su validación y continuar con las siguientes tareas.

2. Diseño de los layouts de la aplicación

El objetivo de esta fase será el de lograr que la aplicación obtenga los niveles de sencillez y claridad que la empresa ha solicitado de forma que permita al proyecto alcanzar la funcionalidad que se requiere. Se necesita que las diferentes pantallas resulten, a los instaladores o cualquier persona que vaya a utilizarla, especialmente intuitivas con la finalidad de no explicar su funcionamiento a todos los usuarios.

Para el diseño, simplemente, se bocetan las diferentes pantallas a modo esquemático para disponer de una visión a gran escala de cómo se podrá observar la aplicación en un futuro.

3. Diseño de la lógica de las actividades

Durante esta fase el objetivo será el de obtener varios diagramas representando las interacciones entre todas las actividades que se decidan crear y también se describirán los métodos y atributos a implementar para cada actividad. En esta tarea se deberá comenzar a estructurar los servicios REST que se deben implementar y por lo tanto invocar desde la aplicación. Este punto es importante dejar bien establecido ya que todo el desarrollo de la aplicación se basará en lo que durante esta fase quede fijado.

4. Validación de la fase de diseño

En este momento del proyecto dispondremos de la documentación necesaria para implementar la solución adoptada durante las fases de diseño, cuya solución es presentada jefe del proyecto valorando alguna posible mejora.

En el momento en el que se disponga del visto bueno del responsable se comenzará con la implementación de la aplicación al completo empezando por los layouts y actividades y terminando con los servicios a invocar.

5. Creación del proyecto en Eclipse

Durante esta tarea se creará el proyecto Android mediante el IDE Eclipse, se establecerá las propiedades básicas de la aplicación; el nombre de la aplicación y del proyecto, el API de Android que se empleará y el logo de la aplicación. Una vez creado el proyecto ya se podrá comenzar a desarrollar todos y cada uno de sus componentes.

6. Implementación de los layouts

Con el diseño realizado previamente y con la documentación disponible en la web de desarrolladores de Android, donde se ofrece toda la referencia necesaria, se crearán todas las pantallas. Es importante destacar que se debe visualizar correctamente para dispositivos con diferentes tipos de pantalla, este hecho será fundamental a lo largo de la realización de esta tarea ya que las imágenes y los layouts podrán sufrir modificaciones para obtener una correcta visualización en según qué tipo de terminales.

7. Implementación de las actividades

En esta tarea se trabajará en la implementación de todas las actividades, es decir la implementación de la lógica de la aplicación y su funcionalidad definida anteriormente, control de errores, invocación de llamadas a servicios REST para interactuar con la base

de datos PostgreSQL empleada por el módulo Tryton creado y la recepción de la información recibida principalmente. El contenido de la llamadas HTTP a los servicios se pospondrá al momento de creación de estos servicios aunque si es importante establecer que será necesario enviar en cada llamada y los datos que convendrán obtener.

8. Implementación de la generación automática de pedidos

Se creará un cron escrito en Python para acceder a la bandeja de la cuenta de correo que se establezca para recibir las notificaciones del cliente con solicitudes de alta de instalaciones. Este cron se debe configurar para que sea ejecutada al menos dos veces al día y en caso de que en la bandeja existan notificaciones nuevas, según el formato esperado, se creará en el sistema un pedido con la información que figure en el correo en cuestión. Adicionalmente a esto se debe crear, dentro de la vista que se dedicará para mostrar cada instancia de pedidos, una plantilla para poder generar un documento en PDF que se remitirá a la empresa que ha sido asignada para la tarea de ejecución.

9. Implementación del Servlet de entrada de peticiones y los servicios REST

Esta tarea consistirá en la implementación de los métodos clásicos de los servicios REST: GET, POST y/o PUT de cada servicio siendo estos los encargados de realizar las consultas y modificaciones en la base de datos e implementar la lógica de enrutado de llamadas al servicio que corresponda. Para ello nos basaremos en la experiencia en proyectos anteriores en la empresa adaptando el funcionamiento de trabajos anteriores al que se requiere en este proyecto.

En este punto también será importante trabajar en los errores que se mostrarán en el fichero de logs del servidor de aplicaciones para poder detectar fallos en el funcionamiento del sistema de una forma cómoda para la persona encargada del mantenimiento de la plataforma.

10. Validación del módulo completo

Una vez finalizado todas las tareas se presentan los resultados en una reunión donde se valorará si es necesario realizar alguna modificación o por el contrario todo está implementado según lo esperado. En caso de que la implementación haya sido exitosa se iniciará un periodo de testeo del sistema donde se estudiará la estabilidad del sistema, la sencillez de uso, se probará toda la funcionalidad y se iniciará un proceso de traslado de la tecnología a las empresas proveedoras.

### *3.2.3 Diseño de layouts*

A lo largo de este apartado se describe el diseño de las diferentes pantallas que deberá presentar la aplicación al usuario, es decir, se define la apariencia de los layouts, haciendo especial hincapié en los botones, campos y en resumen los diferentes componentes que estarán contenidos dentro de los layouts así como su distribución en la pantalla.

Únicamente se trabajará para una orientación de pantalla en vertical ya que no se necesita que de soporte a formato apaisado, esto es debido a que todas las funcionalidades de la aplicación trabajan de forma natural con el teléfono en esta posición.

Analizando la funcionalidad de la aplicación se deben crear los siguientes layouts que más adelante se describirán más detenidamente:

- *Pantalla de autenticación*. En esta pantalla deberán figurar los campos de texto para que al usuario se le permita introducir su nombre y contraseña, junto a un botón para hacer login.

- *Pantalla con menú inicial*. Se deberán mostrar las siguientes opciones: iniciar instalación de tipo Bluetooth y de tipo Bluetooth junto a pegatina promocional, registrar instalaciones pospuestas, realizar una llamada telefónica a Infortrex y consultar el historial de instalaciones registradas por el usuario.

- *Pantalla de instalación tipo Bluetooth*. En esta pantalla deberán figurar las opciones de escanear el código Bidi de la hoja de pedido, realizar una fotografía de la instalación realizada, enlazarse con el dispositivo Bluetooth instalado, escanear el código Bidi adherido al dispositivo, revisar la fotografía tomada, enviar la instalación, posponer el registro de la instalación y además mostrar la información que vaya siendo tomada a lo largo de la instalación, es decir un resumen de todas las acciones realizadas con la aplicación.

- *Pantalla de instalación tipo Bluetooth junto a pegatina promocional*. Para esta pantalla el contenido requerido debe ser similar al definido para el anterior tipo de instalación salvo que en este caso además deberá disponer de la opción de escaneo el código Bidi de la pegatina promocional a colocar en el establecimiento.

- *Pantalla de enlace con módulo Bluetooth*. En está pantalla se deberá observar el listado de todos los dispositivos Bluetooth encontrados, este listado únicamente deberá mostrar los módulos que cumplan una característica que identifique a los dispositivos que se instalan, ya que no interesa mostrar todos los Bluetooth que el terminal del instalador sea capaz de encontrar en un determinado momento.

- *Pantalla de captura fotográfica*. Esta pantalla simplemente mostrará la imagen en tiempo real que está capturando la cámara y un botón para capturar la imagen deseada.

- *Pantalla de visualización de fotografía*. Esta pantalla será una forma de revisar la fotografía que se ha tomado y que va a ser enviada para realizar el registro de la instalación. Con el fin de que si no es lo suficientemente clara se realice una nueva toma fotográfica.

- *Pantalla de resumen de registros realizados*. En este layout se deberá mostrar un listado con información de todas las instalaciones que ya han sido registradas en base de datos correctamente. Con suficiente información como para que el usuario sea capaz de reconocer cada una de ellas y que realmente sea útil tanto para el instalador como para la persona responsable de revisar las instalaciones con la aplicación.

- *Pantalla resumen de instalaciones pospuestas*. De forma similar a como se espera realizar en la pantalla encargada de mostrar el listado de instalaciones ya registradas, en esta se mostrarán las instalaciones que hayan sido aplazadas en su registro por cualquier razón que el instalador se encuentre.

Es importante destacar que no se necesitará diseñar ni implementar un layout para la pantalla de escaneo de códigos Bidi ya que para ello se empleará el apoyo de una aplicación Android diferente a la que se está desarrollando y se capturarán los datos que esta aplicación externa ofrezca. A continuación pasa a describirse en detalle el diseño de cada una de las pantallas.

### Pantalla de autenticación

El diseño de esta pantalla se abordará de la manera tradicional para el tipo de pantallas de autenticación. Se necesitan dos campos de texto para introducir caracteres en una sola línea, tanto para introducir el usuario como la contraseña. Para este segundo campo de texto se deberá emplear la codificación con asteriscos para ocultar la contraseña que se introduzca y un botón para confirmar el deseo de logarse en el sistema. Adicionalmente se deberá incluir el logo de la empresa y un fondo adecuado al contexto de la aplicación, por el momento se deja el fondo en blanco a espera de acordar un fondo para esta pantalla. El aspecto de esta pantalla puede verse en la Fig 3.9..


*Figura 3.9. Boceto de pantalla de autenticación*

Mientras el usuario espera a que la aplicación dé el visto bueno a su logado debe mostrarse un aviso de que el proceso de entrada a la aplicación está llevándose a cabo para que el usuario no pulse el botón en repetidas ocasiones al no ver actividad en la pantalla. En caso de que la autenticación haya sido válida se le dará la bienvenida al usuario y en caso de error en el proceso, se le comunicará al usuario que los datos enviados son incorrectos.

**Pantalla con menú inicial**

Esta pantalla servirá como origen a la realización de acciones mediante la aplicación y será la siguiente pantalla que se mostrará tras el logado correcto del usuario. Conformará el menú inicial donde el usuario elegirá entre las siguientes acciones mediante el pulsado del botón correspondiente como se observa en la Fig 3.10..

- *Iniciar instalación de Bluetooth*, para comenzar a registrar la información que se requiere en este tipo de instalaciones y su posterior envío a Infortrex.

- *Iniciar instalación de Bluetooth acompañado de pegatina promocional*, para comenzar a registrar la información que se requiere en este tipo de instalaciones y su posterior envío a Infortrex.

- *Iniciar registro de instalaciones pospuestas*, para consultar, enviar o incluso borrar instalaciones de las que aún no constan en la base de datos de la empresa y que han sido almacenadas en el terminal por cualquier razón que pudiera haber sucedido, como que el instalador no disponga de tráfico de datos en ese momento. Es importante destacar que esta opción solo debe figurar en el caso de que se disponga de instalaciones almacenadas en el terminal del instalador que aún no hayan sido registradas.

- *Consultar el historial de instalaciones*, para ver el detalle de las instalaciones que ya se encuentran registradas en el sistema de Infortrex. Puede servir como comprobante por parte del personal instalador de que la instalación registrada se ha realizado correctamente.

- *Llamar por teléfono* a las oficinas de Infortrex, para ponerse en contacto con la persona encargada de gestionar estas instalaciones y poder realizar cualquier tipo de consulta, exponer alguna duda o comentario sobre la tarea que se le ha encomendado realizar.

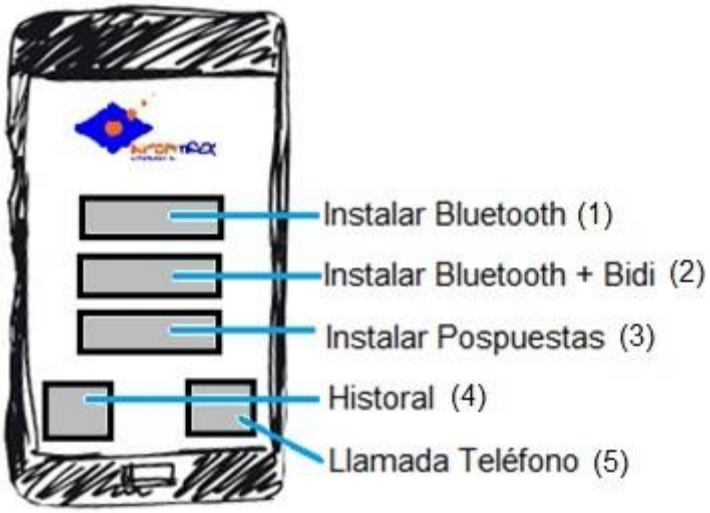



*Figura 3.10. Boceto de pantalla de menú inicial*

Se ha establecido, al igual que en la pantalla de logado, que también figure en la parte superior de la pantalla el logo de la empresa y se emplearán para crear los iconos de los botones de las diversas opciones los que a continuación se exponen en la Fig 3.11..


*Figura 3.11. Iconos de pantalla de menú inicial*

**Pantalla de instalación tipo Bluetooth**

En el caso de que el usuario seleccione la opción de iniciar una instalación de tipo Bluetooth, la pantalla a mostrar será la mostrada en la Fig 3.12.. En esta pantalla se encontrarán los siguientes componentes:

- *Botón para escanear la hoja de pedido*, donde el instalador deberá escanear el código Bidi que figura al inicio de este documento al que se hace referencia, deberá encontrarse en posesión del instalador en el momento de realizar la tarea. Este será el primer paso que deberá realizar el instalador y sin la conclusión de este paso de manera satisfactoria no se deberá mostrar ninguna otra opción en la pantalla.

- *Botón para realizar una fotografía*, gracias al cual el instalador se verá con la posibilidad de realizar una captura donde pueda observarse que la tarea se ha realizado como se refleja en el detalle de la hoja de pedido.

- *Botón para escanear el código Bidi del dispositivo*, donde el instalador deberá escanear el código adherido al módulo que se ha entregado al instalador.

- *Botón para realizar enlace con el módulo Bluetooth*, la funcionalidad de este botón será la de completar la tarea de detectar el dispositivo instalado y obtener la información que proporciona.

- *Campo de resumen de datos*, donde se podrá ir observando la información que se vaya capturando a lo largo de todo el proceso. De esta forma se permite al personal instalador comprobar que los datos están siendo capturados correctamente por la aplicación y que posteriormente van a ser

enviados a Infortrex. Esta área será únicamente a modo informativo por lo tanto no deberá permitirse la escritura en ella.

- *Botón de visualización de la fotografía*, mediante el pulsado de este botón la aplicación mostrará la fotografía que el usuario ha tomado previamente. Es importante destacar que en el caso de que no se haya realizado ninguna captura fotográfica este botón no debe figurar en la pantalla.

- *Botón de registro de instalación*, gracias al cual la aplicación empezará a llevar a cabo el registro de la instalación en la base de datos de Infortrex, previamente se le deberá dar la posibilidad al usuario de introducir comentarios sobre la instalación. Es imprescindible que para que este botón se encuentre habilitado que todas las tareas que debía hacer el usuario se hayan realizado correctamente.

- *Botón de posponer instalación*, gracias al cual la aplicación empezará a llevar a cabo el registro de la instalación en la memoria interna del dispositivo del usuario, previamente se le deberá dar la posibilidad al usuario de introducir comentarios sobre la instalación realizada. Es imprescindible que para que este botón se encuentre habilitado que todas las tareas que debía hacer el usuario se hayan realizado correctamente, de igual modo que para el de registro.

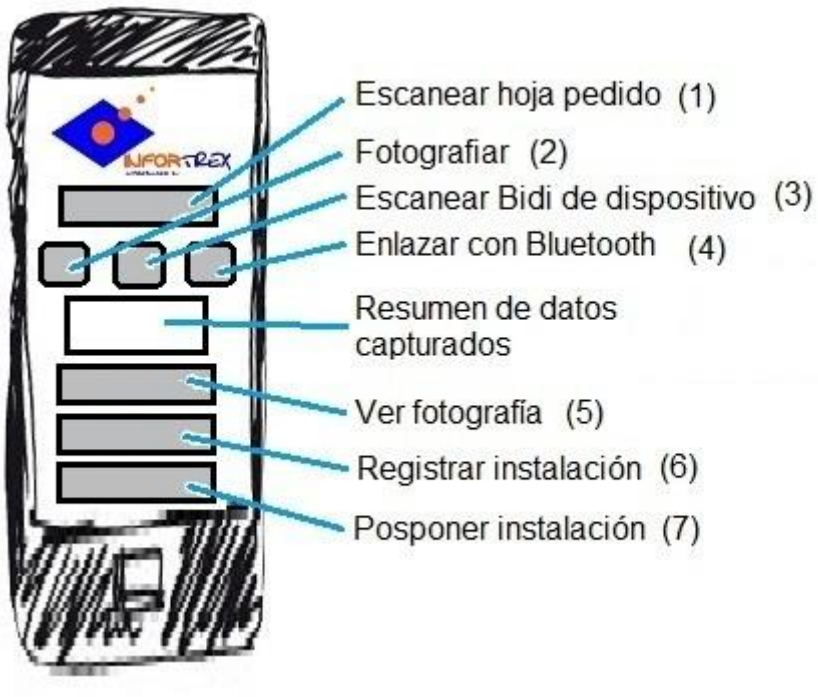



*Figura 3.12. Boceto de pantalla de menú de instalación sencilla*

Como anteriormente se ha establecido en otros layouts, esta pantalla también debe figurar en la parte superior de la pantalla el logo de la empresa y se emplearán para crear los iconos de los botones de las diversas opciones los que a continuación se exponen en la Fig 3.13..


*Figura 3.13. Iconos de pantalla de menú de instalación sencilla*

*(1- Escaneo hoja de pedido, 2- Realizar fotografía, 3- Escaneo Bidi de dispositivo, 4- Enlace con módulo Bluetooth, 5- Previsualizar fotografía, 6- Registrar instalación, 7- Posponer instalación)*

**Pantalla de instalación tipo Bluetooth junto a pegatina promocional**

En el caso de que el usuario seleccione la opción de iniciar una instalación de tipo Bluetooth acompañado de una pegatina promocional, la pantalla a mostrar será la mostrada en la Fig 3.14.. Esta pantalla se compone de la misma funcionalidad que para el tipo de instalación diseñado anteriormente pero añadiendo la opción de escanear el código Bidi que se encuentra en la pegatina promocional que se le ha encomendado al encargado de realizar el trabajo.

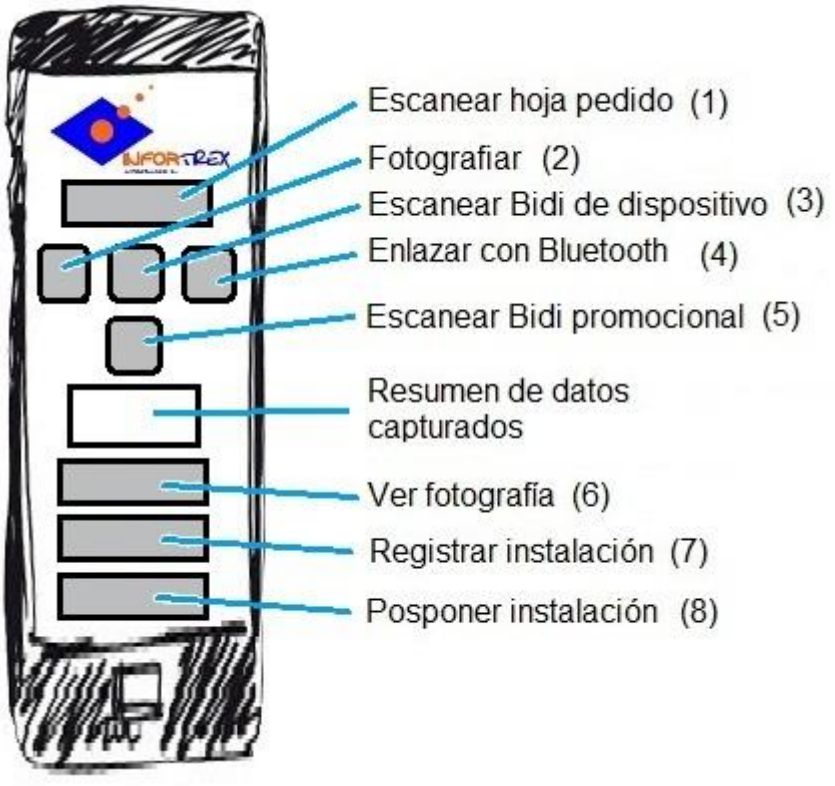



*Figura 3.14. Boceto de pantalla de menú de instalación completa*

Al estar basada está pantalla en el mismo formato que la anteriormente diseñada, los componentes de esta vista serán los mismos; con los mismos iconos y botones pero

añadiendo la nueva imagen para ser empleada en la funcionalidad de escanear el código Bidi de la pegatina promocional. En la Fig 3.15. se encuentran todos estas imágenes que deben figurar en la pantalla.


*Figura 3.15. Iconos de pantalla de menú de instalación completa*

*(1- Escaneo hoja de pedido, 2- Realizar fotografía, 3- Escaneo Bidi de dispositivo, 4- Enlace con módulo Bluetooth, 5- Escaneo de pegatina, 6- Previsualizar fotografía, 7- Registrar instalación, 8- Posponer instalación)*

**Pantalla de enlace con módulo Bluetooth**

Para completar ambas instalaciones se debe enlazar con el módulo de Bluetooth instalado, por ello se empleará la misma pantalla para los dos casos. En cuanto al formato de la pantalla únicamente será necesario incorporar un botón que será el encargado de activar la búsqueda de dispositivos y a continuación se mostrará un listado de todos los Bluetooth encontrados. Dentro del listado deberá figurar para cada dispositivo encontrado una cadena de texto que proporciona el propio módulo, con un formato de dirección MAC, útil para el cliente y que se le deberá proporcionar posteriormente. La pantalla se debe crear con un aspecto similar al mostrado en la Fig 3.16..

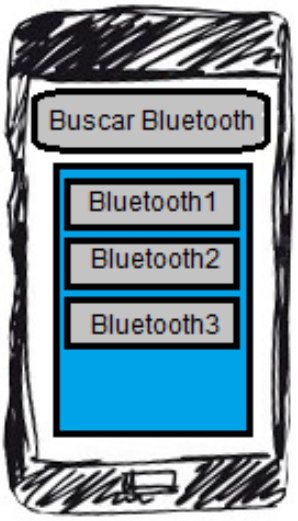



*Figura 3.16. Boceto de pantalla de enlace Bluetooth*

Es importante destacar que se debe mostrar un mensaje en la pantalla a modo informativo de que el proceso de detección de dispositivos se está llevando a cabo mientras el periodo de búsqueda se está llevando a cabo. En cuanto a funcionalidad, el listado deberá permitir el seleccionar uno de los dispositivos que figuren, para continuar con las demás tareas pendientes y registrar en la información de envío a Infortrex el dispositivo instalado.

**Pantalla de captura fotográfica**

Para cumplir con la tarea de la realización de una toma fotográfica del resultado final de la instalación llevada a cabo, se diseña la siguiente pantalla que se boceta en la Fig 3.17.. En ella se deberá observar la imagen visible por la cámara del terminal y un botón que permitirá almacenar la imagen. La pantalla también se empleará para los dos tipos de instalaciones.

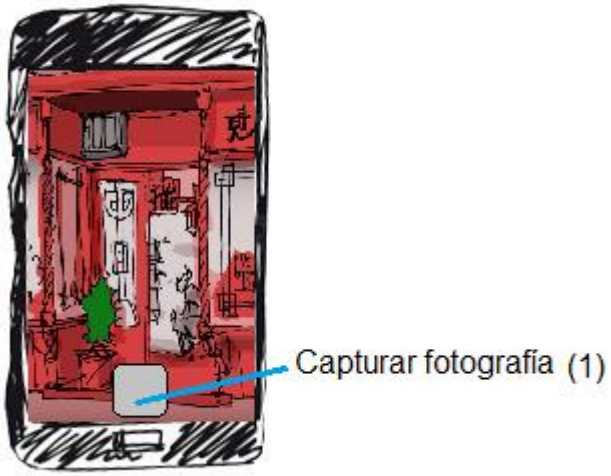



*Figura 3.17. Boceto de pantalla de captura fotográfica*

Para llevar a cabo la captura de la foto se deberá emplear el icono que se muestra en la Fig 3.18. a modo de botón.

*Figura 3.18. Icono de botón de captura fotográfica*

**Pantalla de visualización de fotografía**

En el caso de desear visualizar la fotografía tomada, pulsando el botón correspondiente, la pantalla mostrada únicamente deberá contener la imagen a pantalla completa, sin necesidad de mostrar ningún componente más ya que la funcionalidad de esta pantalla será únicamente la de facilitar una visión de la fotografía que se va a enviar. El aspecto de la pantalla será sencillo tal como se puede observar en la Fig 3.19..

*Figura 3.19. Boceto de pantalla de previsualización de fotografía*

**Pantalla de resumen de registros realizados**

En cuanto al diseño y bocetado de la pantalla de consulta de historial de las instalaciones realizadas, se deberá mostrar con estilo de lista, como se puede ver en la Fig 3.20. y en el orden de más actuales a más antiguas la siguiente información de cada registro. Cada ítem o elemento debe contener dos columnas, en la primera debe figurar la fecha y la hora del registro mientras que en la segunda columna se mostrará el identificador del pedido, la información que fue proporcionada por la pegatina adherida al módulo, los datos del Bluetooth obtenidos al enlazarse con él y la información obtenida

del escaneo de la pegatina promocional en el caso de que la instalación conllevase la colocación de esta pegatina en el local.

En el caso de que se pulsara sobre uno de los ítems se debe mostrar, en caso de que existiera, el comentario introducido por el instalador en el momento de realizar el registro.


*Figura 3.20. Boceto de pantalla de histórico de registros*

**Pantalla resumen de instalaciones pospuestas**

Para el diseño de esta pantalla se emplea el mismo formato que para el caso de la consulta de historial, siendo el boceto el que se muestra en la Fig 3.21.. Con la siguiente diferencia en su contenido, únicamente será necesario mostrar la fecha y hora en la que se pospuso la instalación y el comentario que introdujo la persona que realizó la instalación. En el caso de pulsar sobre cualquiera de los ítems se permitirá al usuario, mediante una ventana emergente, las opciones de borrar la instalación pospuesta o enviar dicha información al sistema de Infortrex.


*Figura 3.21. Boceto de pantalla de instalaciones pospuestas*

### *3.2.4 Diseño de actividades*

A lo largo de esta sección se abordará el diseño del núcleo de la aplicación Android, y en ella se trata de describir el modo en el que se trabajará la funcionalidad a implementar. Se deben especificar las diferentes interacciones entre las diferentes vistas, los diferentes eventos que se deben de tratar, tales como, el pulsado de los diferentes botones de la pantalla o la información que se captura en los diferentes pasos de cualquier instalación; el almacenamiento de datos en la memoria interna del terminal o las llamadas a los servicios REST.

Básicamente lo que se necesita es crear una actividad por layout creado y mediante esa actividad gestionar todas las posibles interacciones del usuario, para esto se expondrá la funcionalidad de cada actividad mediante un diagrama de caso de uso.

Analizando la funcionalidad de la aplicación y los requisitos ofrecidos al inicio del proyecto y posteriormente desarrollados, se deben crear las siguientes actividades que a continuación se describen:

**InstaladoresActivity**

Esta actividad corresponde a la actividad de inicio de la aplicación y deberá tener las siguientes características:

- Al arrancar se debe mostrar el layout de autenticación diseñado anteriormente.

- Consultar si existen datos de logado almacenados en la memoria del móvil, para escribirlos de forma automática en los campos específicos de la pantalla.

- Al pulsar el botón de inicio de sesión se debe enviar una petición a un servicio REST para validar al usuario con el password introducido, para ello se empleará el paquete org.apache.http [40] y el formato de la URL[29] para la petición debe respetar el siguiente formato, siendo este formato el que se empleará para todas las llamadas a los servicios.

[29] Uniform Resource Locator

```
http://<ip:puerto>/DispatcherInstaladores/DispatcherInstaladores?msg=<servicioRest>&param=<parámetro=valor>[straightbar<parámetro=valor>]*
```

- Los espacios en blanco se sustituirán por la cadena ”%20”, ya que no se permite enviar espacios en blancos en una URL.

- Mientras se recibe la respuesta de los servicios REST se debe mostrar una ventana de diálogo informando al usuario de que el proceso de inicio de sesión se está llevando a cabo.

- La estructura de todas las respuestas que se reciban vendrán en formato de JSON, en este caso se requiere únicamente obtener el identificador de usuario, es decir el índice dentro de la base de datos de ese usuario y un valor que indique si el proceso de autenticación se ha realizado correctamente o no.

- En caso de que no se haya llevado a cabo el logado correctamente se debe mostrar un mensaje de error de login.

- Si el logado ha sido satisfactorio, se almacenará el usuario y la clave en la memoria local de la aplicación de modo que, para futuros usos de la aplicación, no se requiera introducir nuevamente los datos de logado. Y posteriormente se realiza una llamada a la siguiente actividad, en este caso la actividad que se encarga de gestionar la pantalla de menú, pasando a esta actividad el nombre del usuario y el identificador de dicho usuario.

- Al pulsar la tecla de regresar del terminal se debe finalizar la aplicación.

En la Fig 3.22. adjunta a continuación se pueden observar todas las interacciones que el usuario debe ser capaz de realizar durante la ejecución de la actividad diseñada, detallándose en el propio diagrama de casos de uso aspectos que conllevarían ejecutar cada uno de ellos a modo de aclaración.

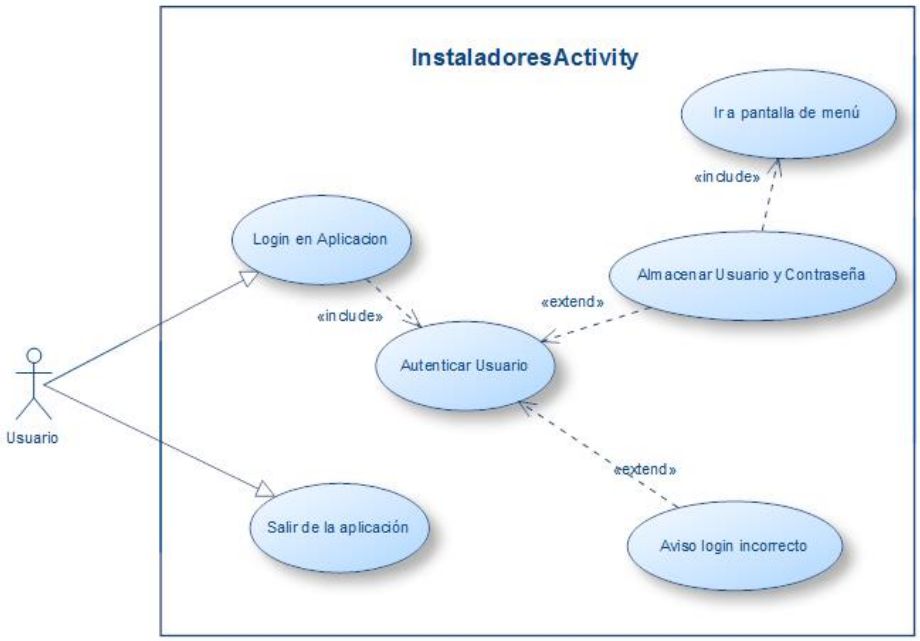



*Figura 3.22. Diagrama de casos de uso de pantalla de autenticación*

**IniciarTrabajoMenu**

Esta actividad es la responsable de ofrecer al usuario las diferentes acciones básicas que puede realizar, como ya se ha comentado anteriormente estas acciones son: realizar una instalación nueva o pospuesta, consultar historial o llamar por teléfono a las oficinas de Infortrex. Las características básicas de funcionalidad corresponden a las siguientes:

- Al iniciarse la actividad se debe mostrar el layout en el que se muestra el menú inicial y en el caso de que se provenga de la pantalla de logado se debe mostrar un mensaje de bienvenida al usuario. En el caso de que se provenga de cualquier otra actividad se deberá, de igual modo, mostrar un mensaje que de una descripción del resultado de la acción realizado anteriormente.

- Se debe comprobar si existe alguna instalación pospuesta en la memoria interna del terminal para mostrar u ocultar el botón de registrar instalaciones pospuestas, ya que no se debe dar la opción de poder registrar instalaciones pospuestas si no existen tales instalaciones.

- Si existieran instalaciones pospuestas estas deberán intentarse quedar registradas en el sistema de forma automática por ello se debe lanzar un hilo que realice tal tarea, además de ofrecer la opción de realizarse manualmente. Como característica del hilo que intenta registrar las instalaciones almacenadas se ha establecido que periódicamente se realicen intentos de registro y que en caso de que se consigan registrar todas las instalaciones almacenadas se emplearán las notificaciones de la barra de estado de Android para tal función.

- Al pulsar el botón de llamada telefónica se requiere invocar una actividad del sistema, la de iniciar una llamada telefónica, pasando como parámetro el número de teléfono al que se desea iniciar la comunicación.

- Al pulsar el botón de consultar historial de instalaciones se debe iniciar la actividad que se creará y se explicará más adelante para cumplir con tal función. Como parámetro a priori se deberá pasar el identificador de usuario. De igual modo se diseña para los botones de iniciar instalación nueva y pospuesta,

se debe invocar a la actividad correspondiente pasando como parámetro el identificador de usuario y el nombre del usuario.

- Si se pulsa el botón del terminal que permite volver hacia atrás la pantalla debe regresar a la pantalla de logado.

En la Fig 3.23. se pueden ver, a modo de gráfico de casos de uso, las posibilidades de interacción del usuario con la aplicación, quedando limitadas a las seis opciones ya explicadas.

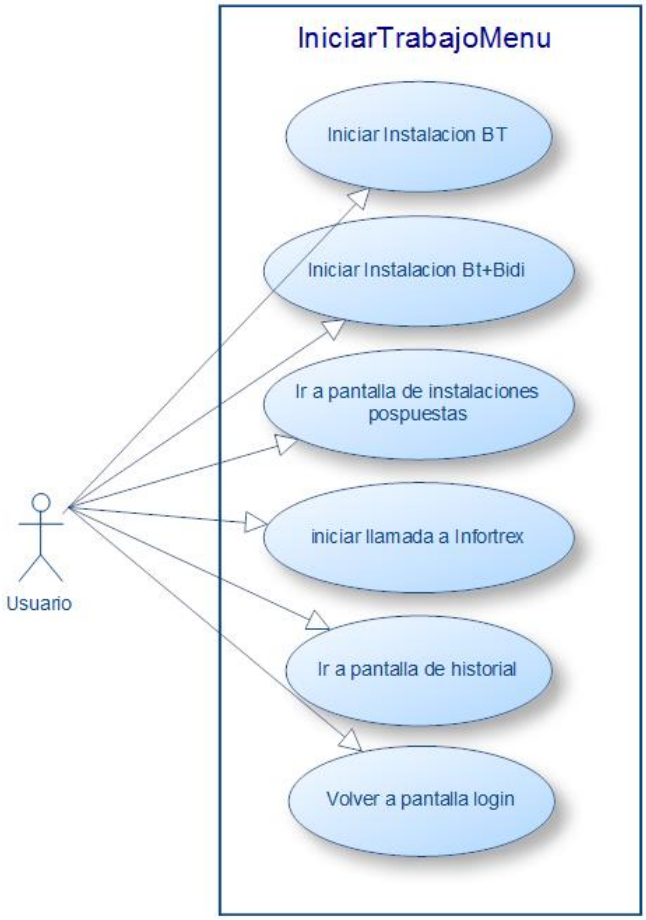



*Figura 3.23. Diagrama de casos de uso de pantalla de menú inicial*

**IniciarTrabajoBT**

En esta actividad se llevará a cabo la orientación del usuario a través de los diferentes pasos que debe realizar antes de registrar un trabajo. En esta actividad se emplea como layout la pantalla de instalación de tipo Bluetooth que se diseñó en el apartado anterior. En esta actividad se debe gestionar el campo de texto definido para mostrar todos los datos capturados, empezando en la primera línea con el nombre del usuario y continuando con los pasos que se vayan completando. Los requisitos básicos de esta actividad corresponden a los que se listan a continuación:

- Al ser creada la actividad por una llamada de otra debe reescribirse la información capturada por el momento en el campo de detalle de la instalación y deben mostrarse exclusivamente los botones que tengan sentido enseñar. Se deben cumplir las siguientes condiciones:

- Si aún no se ha escaneado la hoja de pedido correctamente no se debe mostrar ningún botón a excepción del de escanear hoja de pedido.

- Si no se ha realizado una fotografía no se debe mostrar el botón de revisar fotografía.

- Si no se han obtenido todos los datos de un modo correcto no se deben mostrar los botones de posponer instalación y de enviar instalación.

- Se debe permitir repetir cualquier paso durante la instalación por lo que si por ejemplo, se repite el paso de escanear la hoja de pedido de una forma errónea

se deberán deshabilitar todos los botones para mantener la lógica de la gestión de botones.

- En el caso de que cualquier paso se haya realizado correctamente se debe sustituir la imagen del botón por el que corresponda o en el caso de que se repita cualquier paso erróneamente se debe también sustituir por el que se estipula como tarea pendiente.

- Si se pulsa el botón de escanear hoja de pedido se debe arrancar la aplicación gratuita desarrollada por el grupo Zxing [41] específicamente para desarrolladores, más en concreto la actividad denominada como “scan”, comprobándose si dicha aplicación se encuentra instalada y en caso contrario orientando al usuario a instalarla a través del mercado de aplicaciones de Android. Una vez abierta la aplicación y escaneado el código Bidi de la hoja de pedido, se debe recoger la información obtenida y validada, mediante la comprobación del prefijo establecido “5469656E6461”. A continuación se regresa a la pantalla del asistente de instalación y en caso de haber superado la validación se incorporará la información obtenida al resumen de la actividad. Este funcionamiento debe ser equivalente para el botón encargado de escanear el código Bidi adherido al dispositivo instalado salvo con la diferencia en la validación que en este caso debe presentar el siguiente prefijo “a6:16:3e”.

- Si se opta por pulsar el botón para realizar una fotografía se iniciará la actividad creada para permitir capturar una imagen utilizando la cámara. Pasando como parámetros el resumen de la información capturada, el tipo de actividad que la ha invocado, el nombre y el identificador del usuario con el fin de poder retornar a esta actividad y reflejar la información al completo anteriormente capturado ya que la actividad que se encarga de gestionar la cámara se va a compartir para las dos diferentes actividades que pueden llevar a cabo instalaciones.

- Si se desea llevar a cabo la tarea de enlazarse vía Bluetooth al dispositivo instalado, se deberá pulsar el botón creado para tal efecto. La actividad deberá tratar dicho evento iniciando la actividad que se encargará de gestionar la búsqueda de dispositivos Bluetooth. Pasando como parámetros los mismos que se necesitan en el inicio de la actividad para la captura fotográfica con el fin de poder mantener estable a la aplicación.

- Si se pulsa el botón creado para revisar la fotografía tomada, se deberá también iniciar la ejecución de la actividad creada para gestionar esta funcionalidad.

- En el caso final de desear enviar toda la información obtenida a Infortrex al cumplir con todos los pasos del asistente. Se debe primeramente permitir al usuario introducir en el registro un breve comentario sobre el trabajo, y posteriormente conformar la URL con la que se invocará al servicio que se mantiene a la espera en el servidor de la empresa, adjuntar a la petición la imagen capturada, realizar la comunicación y mantener a la espera de la respuesta mientras se muestra una ventana de progreso al usuario informando de que el proceso se está llevando a cabo. Una vez terminado se deberá dar el aviso de que se ha realizado el registro correctamente o de por qué no ha sido posible el registro completo del trabajo en cuyo caso se debe almacenar la instalación en la memoria y esperar a que posteriormente se registren automáticamente.

- Una vez completada la instalación correctamente o incorrectamente se debe regresar a la pantalla de menú inicial.

- Para el botón de posponer instalación el proceso a seguir es el mismo salvo que no se debe intentar la conexión con el servidor de Infortrex y en su lugar almacenar directamente la información en la memoria de la aplicación.

- En caso de pulsar el botón de regreso del terminal se debe invocar a la actividad de menú inicial, pasando como parámetros los datos de identificador y nombre del usuario como es habitual en el resto de actividades.

En la Fig 3.24. se resume, a modo de diagrama, la funcionalidad básica de la actividad, convirtiéndose en la base para la implementación de la propia actividad en código y ofreciendo la posibilidad de ser mostrada de una manera clara y efectiva a la persona que se encargue de validar el diseño.

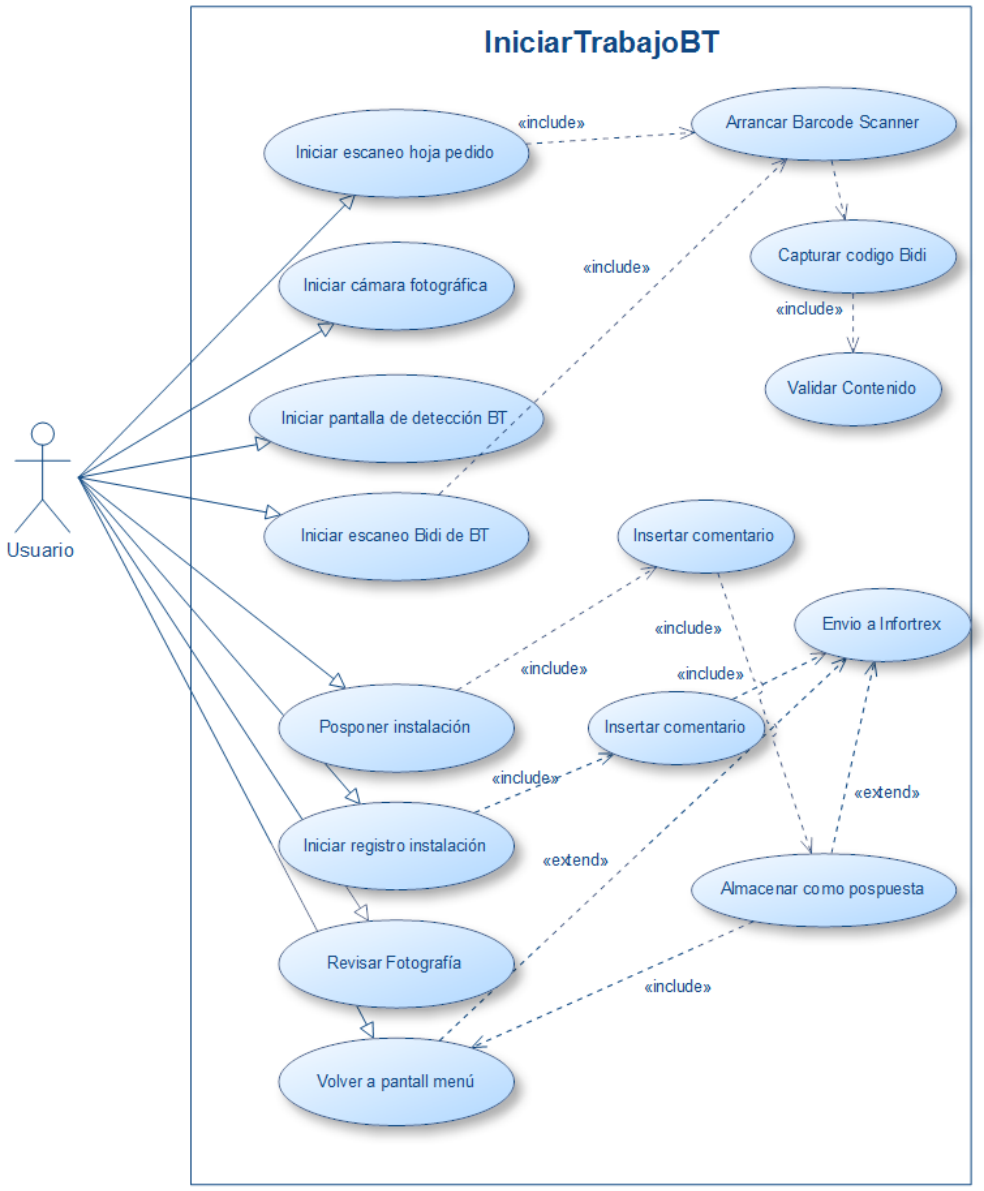



*Figura 3.24. Diagrama de casos de uso de pantalla de instalación sencilla*

## IniciarTrabajoBidi

Esta actividad debe trabajar de un modo equivalente a como lo hace la actividad *IniciarTrabajoBT* ya descrita anteriormente. La diferencia más importante entre ambas actividades corresponde a que la pantalla a mostrar en este caso corresponde a la de tipo instalación Bluetooth acompañado de una pegatina promocional y se debe añadir a la

funcionalidad el botón de escaneo del código Bidi presente en la pegatina. Siendo el modo de trabajar exactamente igual a cómo se escanea cualquier otro código Bidi salvo que la validación en este caso debe comprobar que la información obtenida del escaneo comienza con la siguiente cadena “b1:a6:16:3e”. Todo esto afecta a que durante el envío de la información capturada debe de ir adjunto un valor adicional al del caso anterior.

De igual modo que se realizó para las actividades anteriores, se adjunta el diagrama de casos de uso, Fig 3.25., que es semejante al adjuntado para la actividad *IniciarTrabajoBidi* con la diferencia de que se añade la posibilidad de escaneo del código Bidi existente en la pegatina promocional a colocar por el instalador.

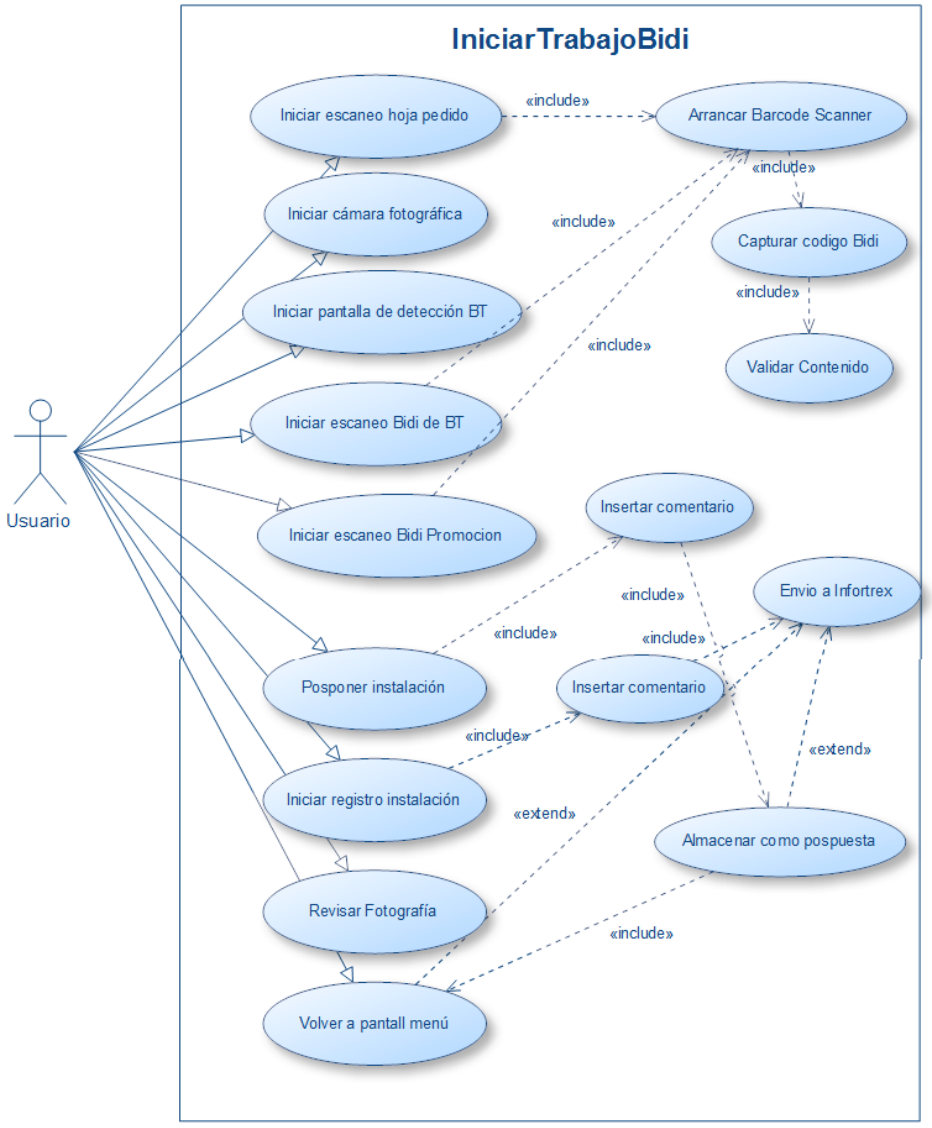



*Figura 3.25. Diagrama de casos de uso de pantalla de instalación completa*

**BluetoothActivity**

Para completar cualquier instalación, como ya se ha comentado, es necesario detectar el dispositivo que se ha instalado en el local en cuestión y de esta forma se determina que el módulo Bluetooth instalado se encuentra operativo. Dentro de las dos posibles pantallas creadas a modo de asistente para registrar el trabajo se encuentra el botón destinado a detectar el módulo Bluetooth, que una vez pulsado debe iniciar esta actividad que a continuación se describe mediante los siguientes puntos:

- Primeramente se debe asignar y hacer visible la pantalla de enlace con el módulo Bluetooth que se ha diseñado y posteriormente hacer la comprobación del estado del Bluetooth del propio terminal del instalador para una vez habilitado permitir el uso del botón definido para detectar un posible listado de módulos.

- A modo de referencia, se debe emplear para manejar todas las posibilidades de gestión del Bluetooth de un terminal, la librería *android.bluetooth*, tanto para comprobar el estado del terminal como para buscar dispositivos y obtener información de ellos.

- Una vez pulsado el botón de búsqueda de dispositivos se informará al usuario de que el proceso se está llevando a cabo y durante el proceso se irán añadiendo a la pantalla todos los Bluetooth que se detecten y que cumplan la siguiente condición, la dirección de cada dispositivo que debe figurar en el listado debe comenzar por la cadena “07:12:05” o por “00:12:06”, siendo una condición que permite detectar exclusivamente los dispositivos de interés para el registro de la instalación.

- En caso de no detectar ningún dispositivo del tipo esperado se debe mostrar un mensaje informando al usuario de este suceso dándole la posibilidad de reintentar el proceso tantas veces desee.

- Una vez se disponga del Bluetooth esperado en la pantalla con pulsar sobre el ítem se captura la información deseada, es decir, su dirección; y se regresa al asistente de la instalación correspondiente gracias a la información que en la llamada a esta actividad se proporcionó continuando pasando por parámetros la información existente en el resumen de datos, el nombre y el identificador de usuario.

- En caso de pulsar el botón de regreso durante la ejecución de esta actividad se debe volver a la actividad del asistente de registro de la instalación correspondiente, conservando toda la información existente hasta el momento.

En la Fig 2.36. puede observarse, a modo de resumen, las diferentes opciones que el usuario debe ser capaz de realizar mediante el uso de la actividad explicada.

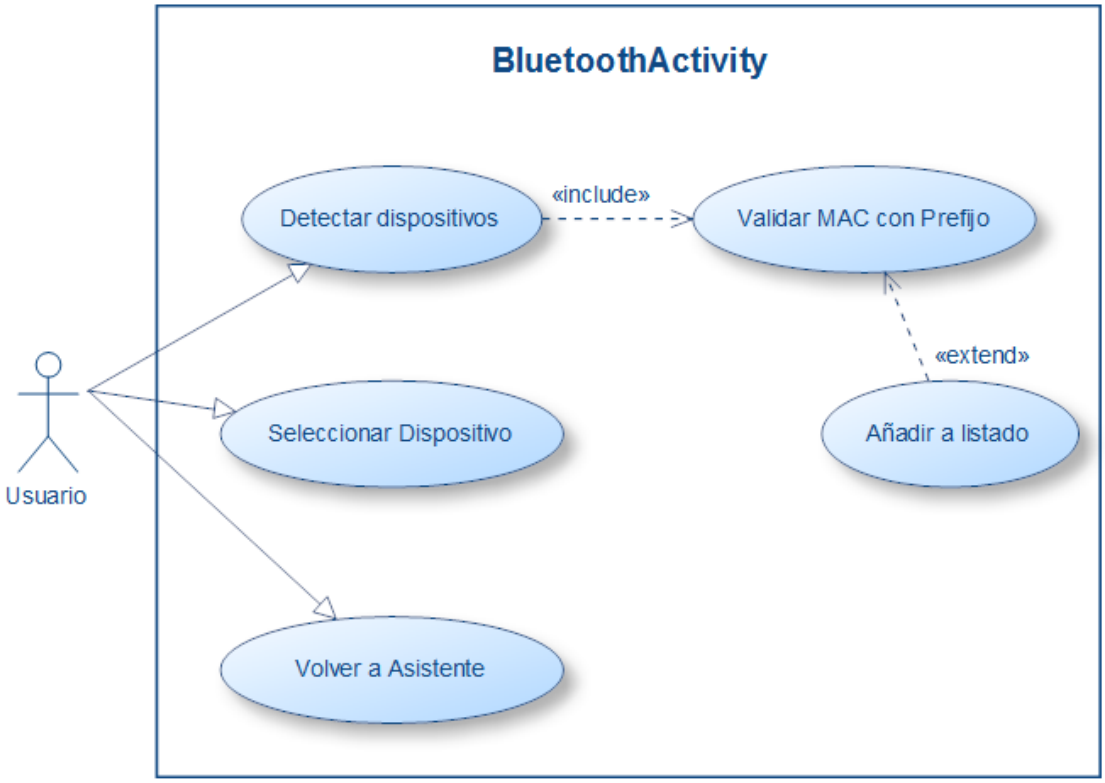



*Figura 3.26. Diagrama de casos de uso de pantalla de enlace con Bluetooth*

**Camara**

La pantalla que se debe mostrar durante el proceso de captura es la ofrecida durante la fase de diseño del apartado anterior. Para la actividad encargada de gestionar esta funcionalidad se ofrecen los siguientes requerimientos:

- Para trabajar con la imagen se opta por redimensionar la imagen al tamaño de la pantalla de forma que se reducirá en gran medida el tamaño de la misma y por lo tanto el envío a los sistemas de Infortrex será más ligero beneficiando tanto a los proveedores que no consumirán de un modo significativo su tarifa de

datos como a la empresa ya que no almacenará imágenes demasiado pesadas dentro de la base de datos, que como ya se ha descrito anteriormente se almacena directamente en un registro de la base de datos.

- Las imágenes se deben mostrar en todo momento en el área del layout especificada para tal efecto.

- En el momento de pulsar sobre el botón destinado a capturar la imagen esta será almacenada en la memoria del terminal y regresará a la pantalla del asistente que corresponda conservando y añadiendo la nueva información obtenida en esta fase realizada, en este caso la ruta a la imagen almacenada.

- Para llevar a cabo el control del hardware de la cámara fotográfica se facilita como referencia la librería de utilidades denominada *android.hardware.camera* que deberá ser importada.

- En caso de pulsar sobre el botón volver del terminal, la aplicación deberá regresar a la actividad que llamo a esta actividad conservando los datos existentes previamente.

El diagrama de casos de uso para esta actividad queda reducido a lo mostrado en la Fig 3.27., siendo la interacción del usuario simplemente el realizar la captura de la imagen.

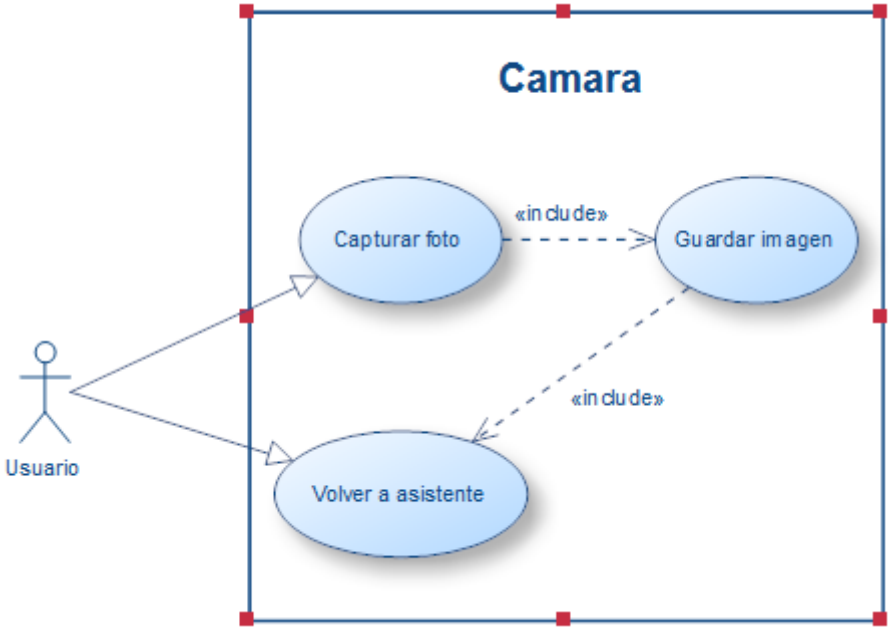



*Figura 3.27. Diagrama de casos de uso de pantalla de captura fotográfica*

**Revisar Foto**

Una vez realizada la captura de una fotografía la funcionalidad de esta actividad debe estar habilitada para el usuario, en ella simplemente se llevará a cabo la visualización de la imagen almacenada en el terminal y cuya ruta se encuentra a disposición de la aplicación gracias al resumen de datos que se mantiene continuamente actualizado.

Se mostrará la imagen a pantalla completa y se decide que únicamente se puede abandonar la actividad pulsando la tecla de regresar del terminal, volviendo a la actividad que fue originaria de la llamada a la actividad de *revisarFoto*.

El diagrama de casos de uso en este caso es realmente como se puede observar en la Fig 3.28., el usuario solamente debe cerrar la previsualización. No existe ninguna otra alternativa para el usuario como ya se ha descrito en este capítulo.

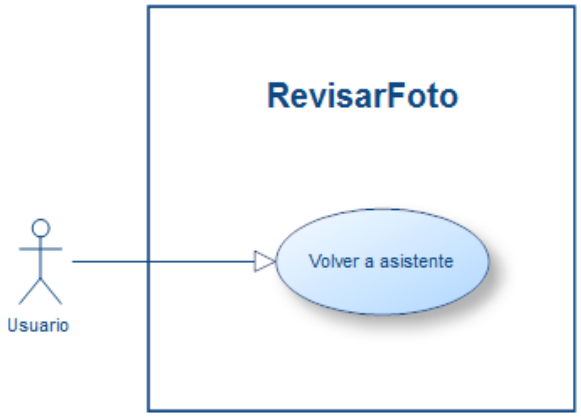



*Figura 3.28. Diagrama de casos de uso de pantalla de previsualización de captura*

**Historial**

Desde la pantalla del menú inicial de la aplicación se puede acceder a la posibilidad de consultar el historial de registros en el servidor de Infortrex del usuario. En el siguiente listado de puntos se ofrecen la base de requisitos funcionales que la empresa requiere para esta actividad.

- Una vez iniciada la actividad se debe asignar a la actividad el layout diseñado para ella, la pantalla de registros realizados. En cuanto se inicie la actividad está se debe comunicar con el servicio REST correspondiente para obtener los datos y poder ofrecérselos al usuario.

- Mientras la comunicación se lleva a cabo se debe mostrar al usuario un mensaje informando del progreso de la obtención de información del sistema.

Para ello únicamente será necesario incluir el identificador de usuario dentro de la URL.

- Se establece que la información que debe figurar de cada ítem sea la siguiente, siendo obligatorio el orden en el que se muestran por proximidad temporal para mejorar la utilidad de la función:

    - Fecha de registro del trabajo
    - Hora de registro del trabajo
    - Información del Bluetooth
    - Información del Bidi adherido al dispositivo
    - Información de la hoja de pedido
    - Información de la pegatina promocional

- En caso de pulsar sobre cualquier ítem de los que figuran en el listado se debe mostrar, en el caso de que exista tal información, el comentario que el instalador introdujo en el momento del registro del trabajo.

- Para volver a la pantalla de menú de la aplicación se deberá pulsar el botón de regreso del terminal.

Por todo esto descrito, la posibles interacciones del usuario quedan reducidas de una forma significativa a la consulta de un registro del listado, como puede observarse en la Fig 3.29..

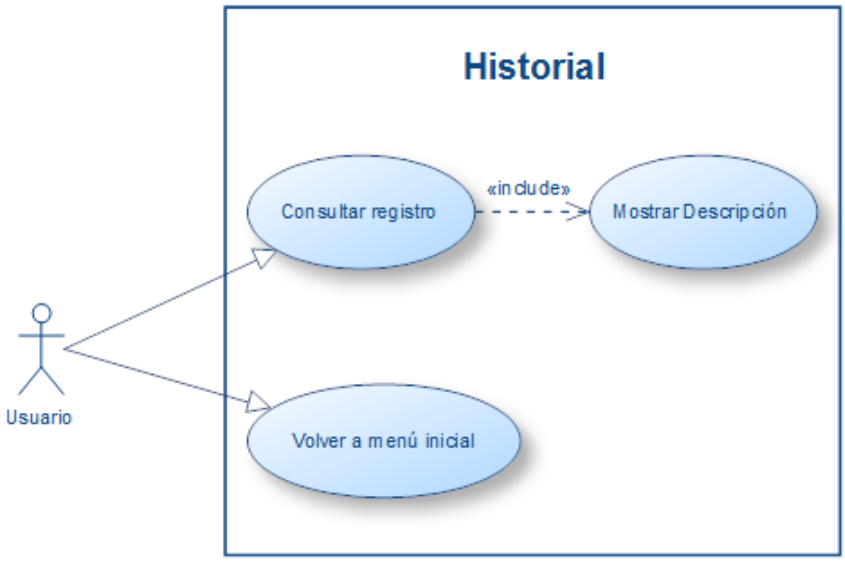



*Figura 3.29. Diagrama de casos de uso de pantalla de histórico*

**Pospuestas**

Esta actividad se diseña realmente con el fin de proporcionar al usuario la opción de observar las instalaciones que el usuario ha llevado a cabo y no se han podido registrar por cualquier asunto que pueda suceder. Las características de funcionamiento básico se reflejan en el siguiente listado:

- Para ello únicamente se asignará la pantalla que se ha diseñado para este efecto, siendo rellenada en el momento del inicio con la información almacenada en el propio terminal. En principio se requiere, según el diseño de la pantalla previo, que figure el comentario que se introdujo en el momento de la finalización de la actividad y la fecha y hora del mismo.

- Mediante el pulsado de cualquiera de los ítems del listado se debe ofrecer al usuario la posibilidad de llevar a cabo un intento de registro directamente o simplemente el borrado de la instalación de la memoria de la aplicación.

- En caso de que el usuario decida realizar un registro manual de una instalación pospuesta el proceso será completamente el mismo que se emplea en el registro automático salvo con la diferencia de que si no es posible el registro en el servidor esta instalación no debe ser almacenada en el terminal al ya estar almacenada.

- En caso de pulsar sobre el botón de regreso del terminal la aplicación debe regresar a la pantalla con el menú de inicio.

El diagrama de casos de uso durante la ejecución de esta actividad se representa en la Fig 3.30., donde se observa claramente que el usuario podrá, como ya se ha descrito anteriormente, consultar una instalación pospuesta y borrarla del terminal o lanzar un registro manual contra el servidor de Infortrex.

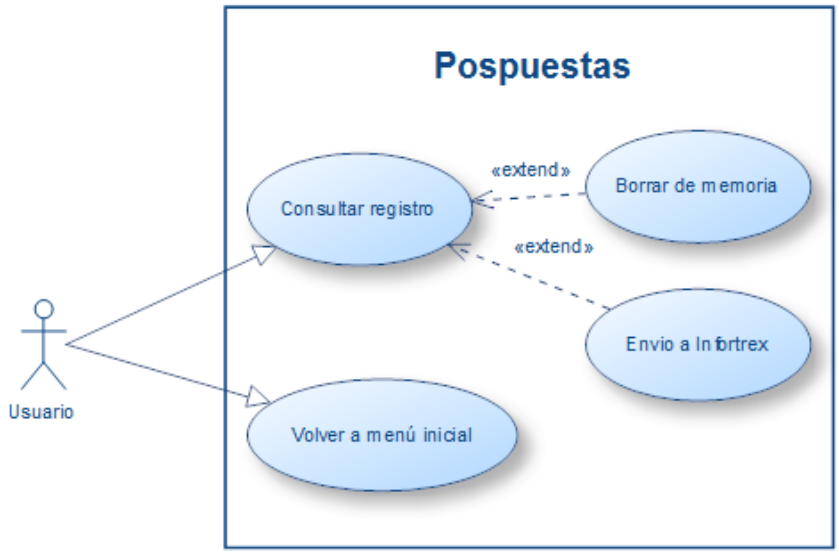



*Figura 3.30. Diagrama de casos de uso de pantalla de instalaciones pospuestas*

## 3.3 Desarrollo de la aplicación

Una vez que se dispone de todas las especificaciones, tanto de funcionalidad de las actividades a implementar como de diseño de las diferentes pantallas de la aplicación que se debe desarrollar, en este apartado se pasa a explicar de qué manera ha sido creado la aplicación con el IDE de Eclipse desde su definición hasta la creación de los servicios REST encargados de interconectar a la base de datos de PostgreSQL de Tryton con las peticiones iniciadas desde la propia aplicación.

### *3.3.1 Creación de la aplicación*

Empleándose todos y cada uno de los componentes, que se han enumerado y descrito anteriormente, se empieza con el desarrollo de la aplicación. Siendo el primer paso la creación del proyecto de aplicación Android en Eclipse.

Una vez creado el proyecto se lleva a cabo el desarrollo de los layouts y activitys planificados.

### *3.3.2 Implementación de layouts*

En esta sección se entra en el detalle de la creación de todos los layouts que fueron diseñados en su momento. Es fundamental que para garantizar el buen resultado de esta

tarea se cumplan con todos los requerimientos previos y con las especificaciones que se establecieron en la fase de diseño; ya que todos estos aspectos han sido validados por el responsable del proyecto y toda la implementación de las actividades está basada consecuentemente en estos bocetos.

Se emplea un fichero XML para cada pantalla, estos fichero son ubicados dentro de la subcarpeta “layout” de la carpeta “res” y a continuación se pasa a describir cada pantalla implementada. Para facilitar el entendimiento de la estructura de vistas de cada pantalla se ha optado por representar cada fichero XML formando un diagrama de árbol con todos los elementos que lo componen y adjuntar una vista del resultado final de la pantalla para explicar la implementación final.

**Pantalla de autenticación**

Para la implementación del layout de logado de la aplicación, tal y como se observa en la Fig 3.31., se ha tomado como contenedor un LinearLayout con orientación vertical, por lo tanto abarcará toda la pantalla este layout y debe configurarse con el valor fill_parent tanto para el ancho como el alto, en el que se colocan los elementos de forma consecutiva. En el layout se establece como fondo una imagen para hacer más atractiva la pantalla. Esta imagen se coloca, al igual que el resto de imágenes que conformen la aplicación, en la estructura de proyecto en la carpeta res dentro de la carpeta drawable-mdpi.

En el interior del layout se establece, basando la implementación en el diseño previo, el logo de Infortrex al cual se le asigna un tamaño en unidades abstractas siendo así dependiente al tamaño de la pantalla. En el resto de componentes de la aplicación siempre se definen medidas de este modo para garantizar en la medida de lo posible una buena apariencia en todos los dispositivos.

A continuación se colocan los dos campos de texto en los que el usuario introducirá los datos de autenticación. De entre todos los atributos de los que se emplean, son especialmente destacables que ambos componentes se configuran con una línea individual y que el contenido del componente de la contraseña se esconderá enmascarado. El contenido inicial con el que se muestran los dos campos se encuentra almacenado en el fichero “strings.xml” dentro de la carpeta values. Esta práctica con los textos se ha generalizado a todas las cadenas de texto existentes en la aplicación con el fin de garantizar la simplicidad a la hora de realizar futuras modificaciones o incluso para poder incorporar la opción de traducción a varios idiomas.

Tras los campos de texto se añaden el botón de confirmación de la intención de logado y una imagen para que el fondo de la pantalla tenga una mayor vistosidad, siendo necesario su ajuste al ancho y al alto de la pantalla para que se encuentren completamente integradas ambas imágenes. En ninguno de estos dos componentes se entrará en mayor detalle al considerarse innecesario para el interés del capítulo.

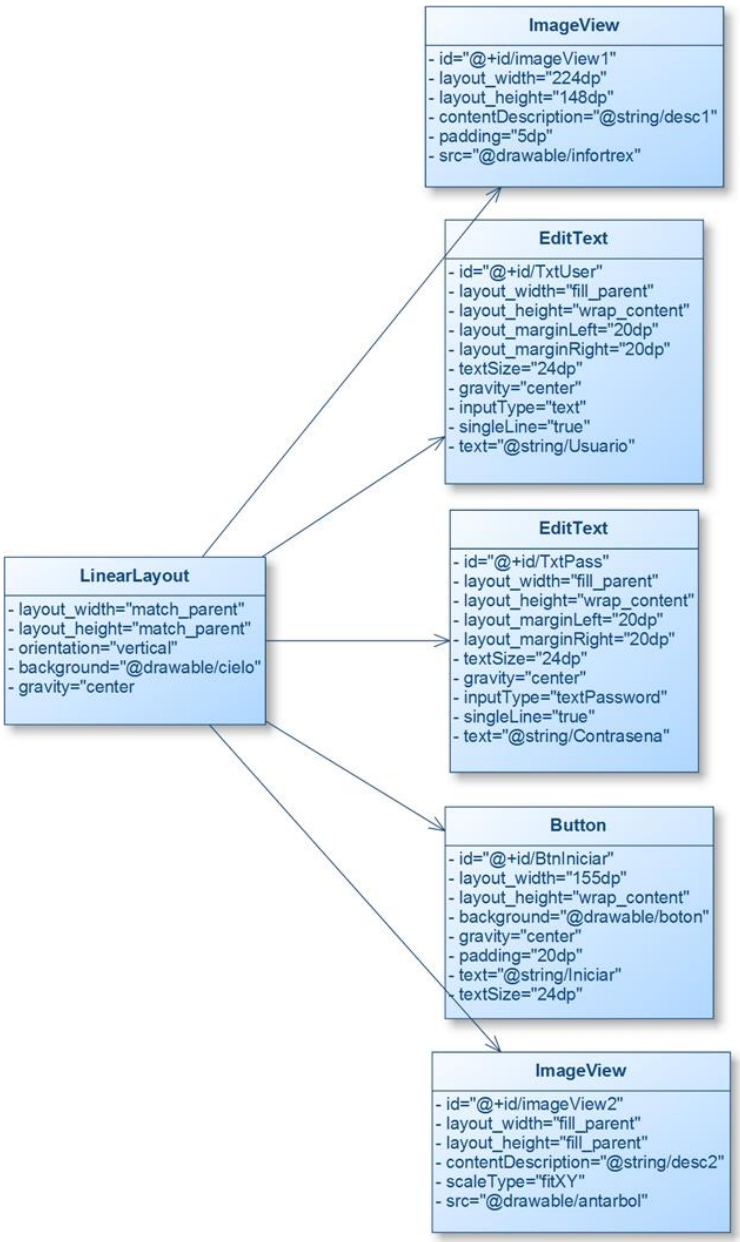



*Figura 3.31. Detalle de componentes del layout de autenticación*

El resultado final de la pantalla se muestra en la Fig 3.32. tal y como se muestra en el editor gráfico de layouts de Eclipse. Un aspecto de la implementación de las pantallas es que en este proyecto únicamente se ha empleado este modo del editor para comprobar el resultado que se iba consiguiendo, siendo la programación en XML la manera más exitosa de conseguir el resultado esperado ya que el editor gráfico no resulta del todo intuitivo y eficaz.


*Figura 3.32. Layout de pantalla de autenticación*

**Pantalla con menú inicial**

A continuación se describe, Fig 3.33., la generación de la pantalla del menú inicial que, como ya se ha explicado en la fase de requisitos, toma importancia tras realizar un correcto proceso de autentificación. En esta pantalla se ha estructurado de forma similar a como se ha llevado a cabo la pantalla anterior todos los componentes se encuentran en el interior de un LinearLayout con orientación vertical que ocupa la pantalla completa.

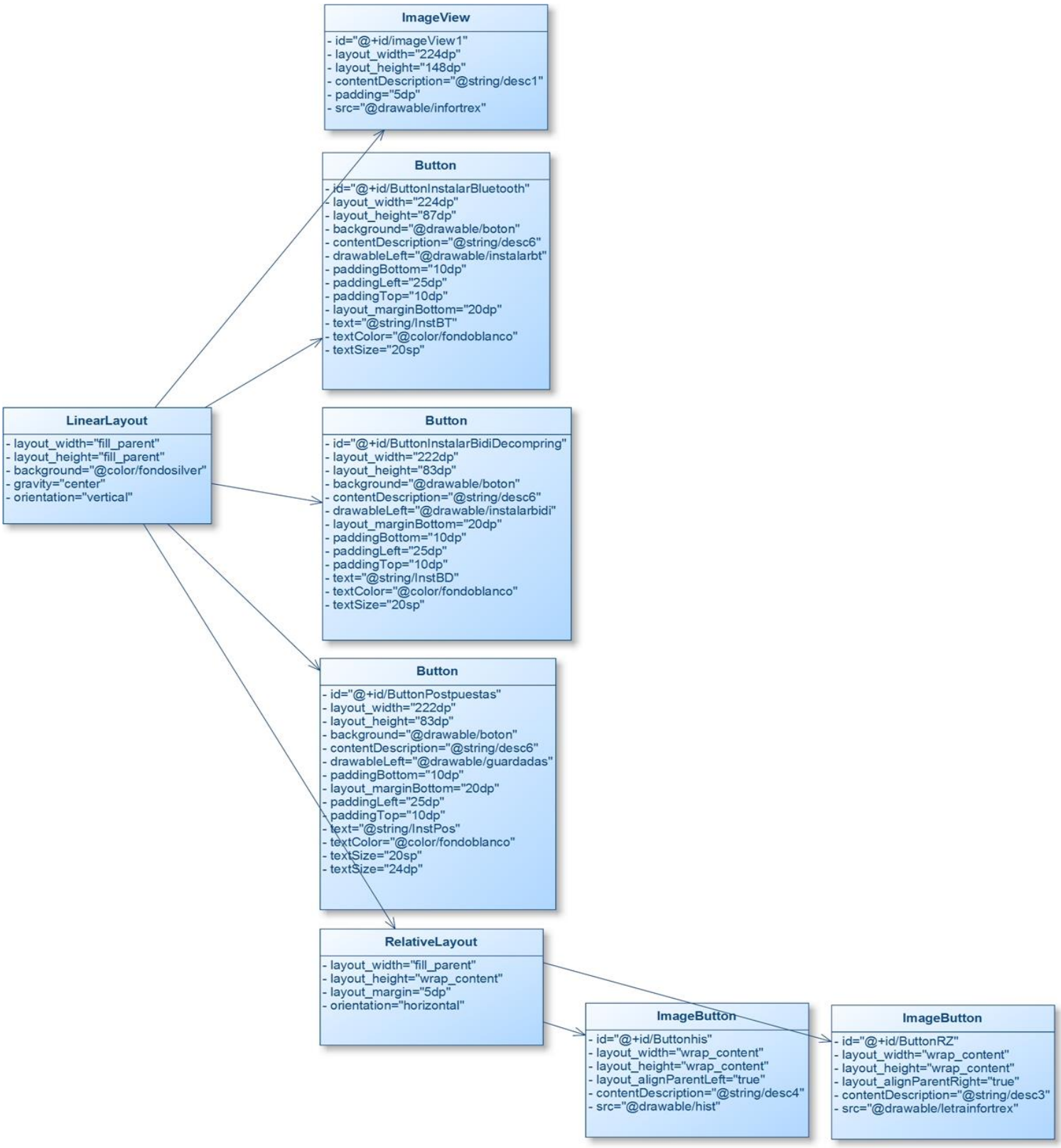



*Figura 3.33. Detalle de componentes del layout del menú principal*

Para presentar el resultado final de esta pantalla se ofrece, en la Fig 3.34., una captura de pantalla del layout implementado. En ella se puede observar como los botones han sido diseñados para que vengan incluidos los iconos correspondientes en el componente gracias a la creación de nuevos elementos Drawable.


*Figura 3.34. Layout de pantalla del menú principal*

**Pantalla de instalación tipo Bluetooth**

En la implementación de la pantalla que actúa como asistente de la instalación de un dispositivo Bluetooth se puede encontrar la pantalla más compleja de todas las que dispone la aplicación, no por ello difícil de ser entendida. Este layout se encuentra basado en un ScrollView ya que esta actividad es propensa a ocupar una longitud de pantalla considerable en algún caso. Para poder acceder a todo el contenido se emplea este tipo de componente. En la Fig 3.35. se puede observar en detalle cada uno de los elementos que componen el layout y la estructura del fichero XML correspondiente.

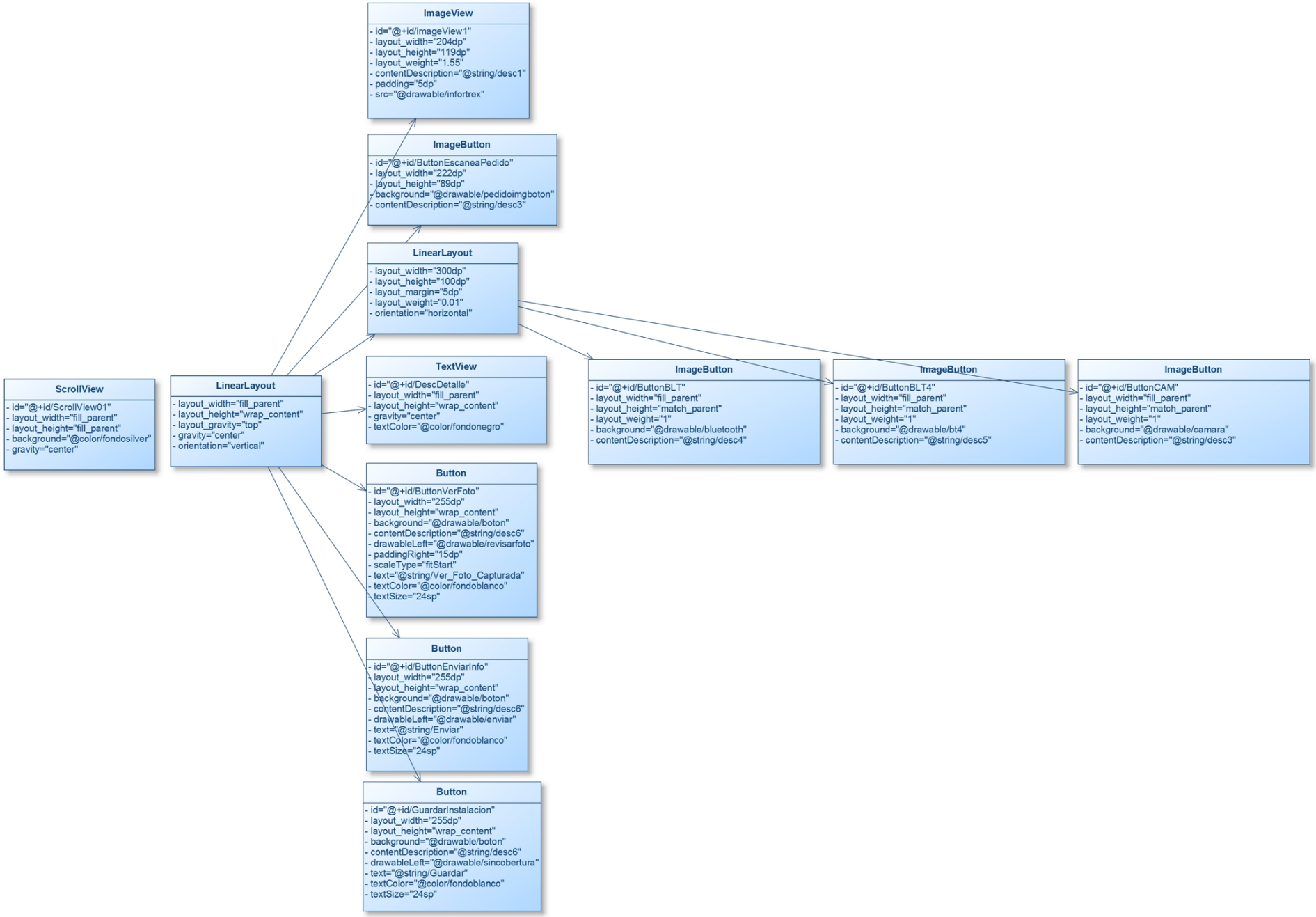



*Figura 3.35. Detalle de componentes del layout del menú instalación sencillo*

Para observar el resultado obtenido con la implementación previamente detallada se adjunta la Fig 3.36., en la cual se pueden observar todos los componentes necesarios acorde a los requisitos de funcionalidad. Es importante destacar que en esta figura no es posible observar el componente TextView que debe figurar entre el LinearLayout horizontal compuesto de tres botones y el botón que permite revisar la fotografía ya que en el editor gráfico no es posible visualizar este componente al no disponer de texto en su interior hasta el momento.


*Figura 3.36. Layout de pantalla del menú de instalación sencillo*

## Pantalla de instalación tipo Bluetooth junto a pegatina promocional

La pantalla que se emplea como asistente para el registro de instalaciones donde es necesario, además de colocar el dispositivo Bluetooth tradicional, ubicar una pegatina promocional; se emplea el mismo modelo que para el caso de la instalación sencilla salvo

por la diferencia que, en este caso, se ha añadido un nuevo elemento de tipo Button para permitir el escaneo del código Bidi de la pegatina a colocar por el usuario. Como se puede observar en la Fig 3.37. el diagrama es prácticamente el mismo que en el caso anteriormente explicado.

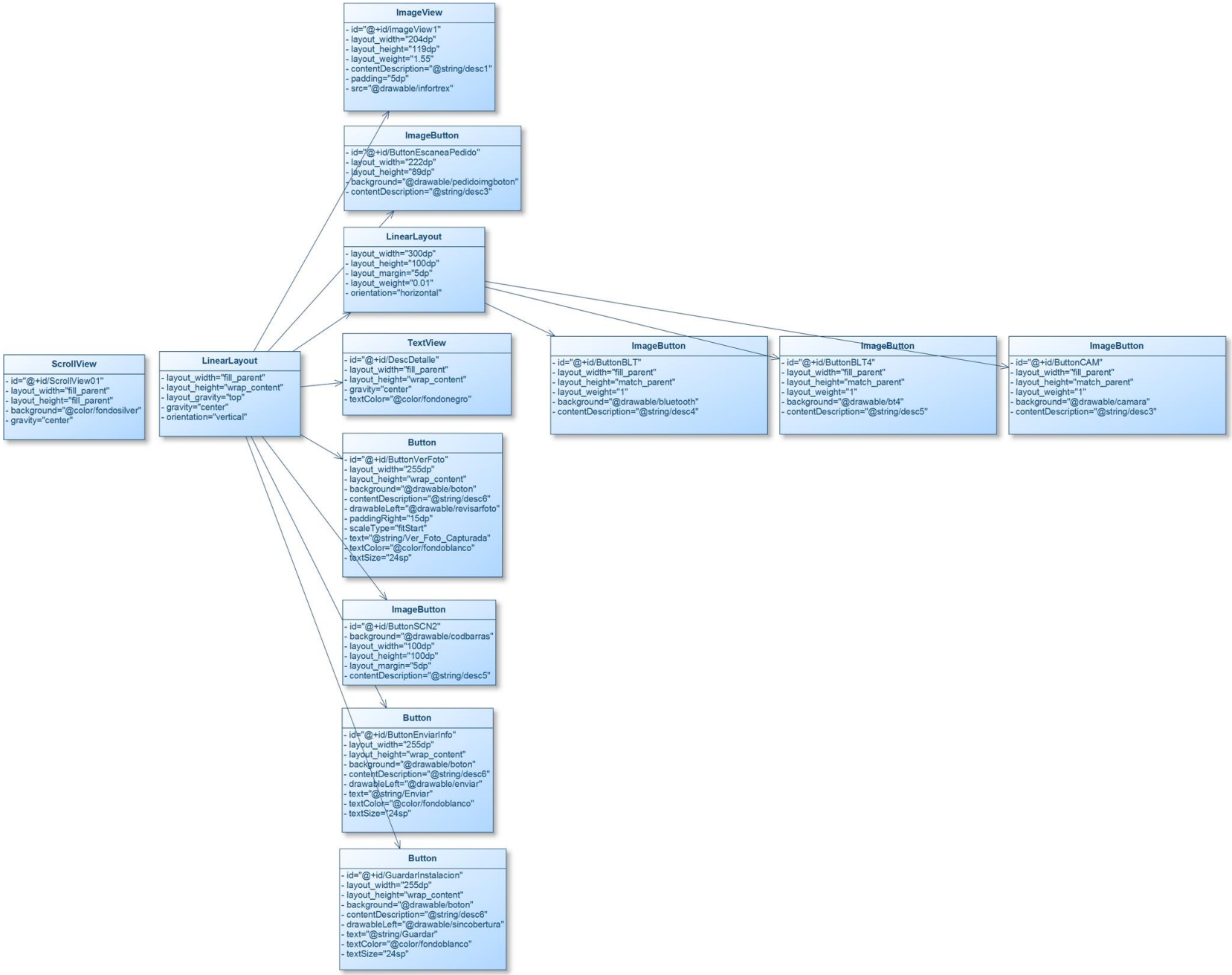



*Figura 3.37. Detalle de componentes del layout del menú instalación completo*

A continuación se muestra, en la Fig 3.38., el resultado que se ha obtenido con la implementación descrita en la anterior figura. De igual modo que sucedía en la pantalla del otro asistente tampoco se puede observar el TextView existente en su interior.


*Figura 3.38. Layout de pantalla del menú de instalación completo*

**Pantalla de enlace con módulo Bluetooth**

La pantalla en la que se permite detectar todos los dispositivos de Infortrex y seleccionar el instalado, se ha implementado del modo que se observa en el diagrama de la Fig 3.39..

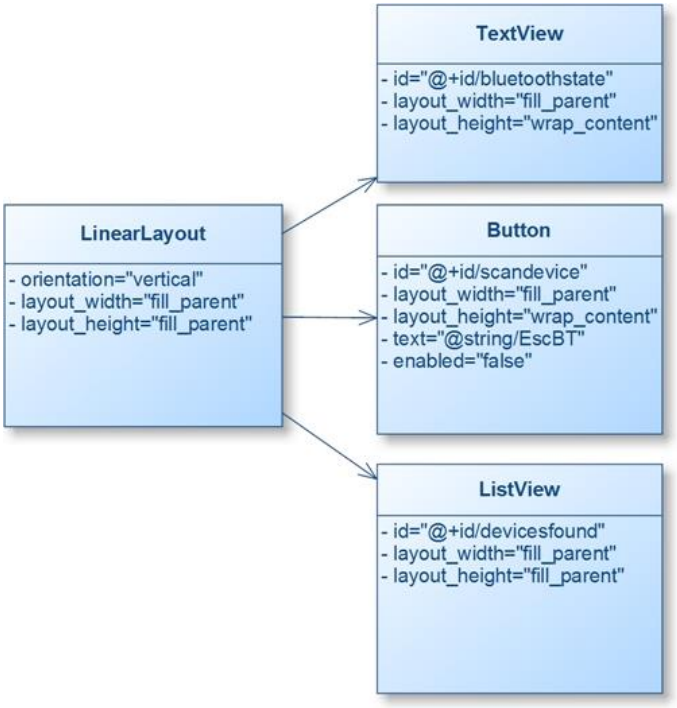



*Figura 3.38. Detalle de componentes del layout de detección de dispositivos*

La apariencia de esta pantalla en el terminal que figura en el editor gráfico para este layout implementado se muestra en la Fig 3.40..

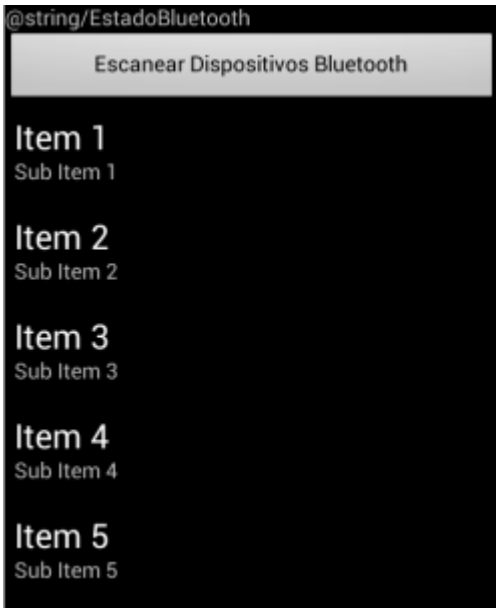



*Figura 3.40. Layout de pantalla de detección de dispositivos*

**Pantalla de captura fotográfica**

Para cumplir con los requisitos de apariencia establecidos en la fase de diseño, la pantalla encargada de cumplir con la funcionalidad de “cámara de fotos”, Fig 3.41., únicamente se requiere de un campo para mostrar la imagen recogida por el terminal de manera instantánea y un botón para llevar a cabo el almacenamiento de la misma.

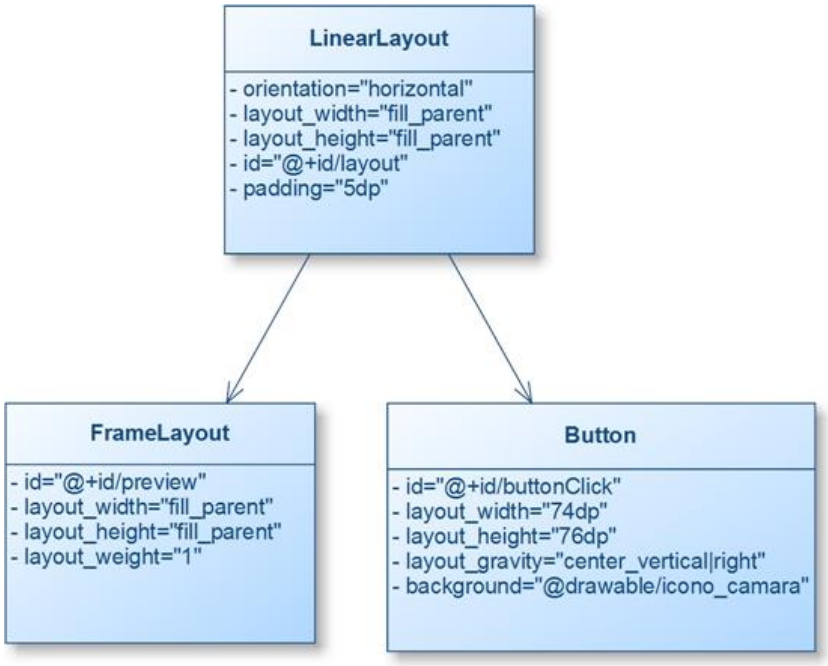



*Figura 3.41. Detalle de componentes del layout de cámara fotográfica*

Como puede observarse en la Fig 2.42, la pantalla no parece estar compuesta por un Layout horizontal sino una vertical. La razón de ello es que en pruebas anteriores con cámaras fotográficas se pudieron observar dificultades en la gestión de algunas propiedades de las imágenes y por ello se opta por girar el layout completo dando una apariencia de ser un LinearLayout con orientación vertical aunque como ya se ha explicado no lo es en realidad.

*Figura 3.42. Layout de pantalla de cámara fotográfica*

**Pantalla de visualización de fotografía**

Para la pantalla encargada de visualizar la fotografía almacenada, únicamente se requieren de los elementos que se observan en la Fig 3.43.. En ella únicamente se observa un LinearLayout que en principio no tiene importancia el parámetro de orientación ni ningún otro salvo los que indican que debe rellenar el tamaño de la pantalla al completo la fotografía a mostrar y que se adhiere al contenido del elemento de tipo FrameLayout. En este caso no se adjunta imagen del aspecto de la pantalla ya que depende completamente de la imagen a mostrar no existiendo ningún componente adicional en la misma.

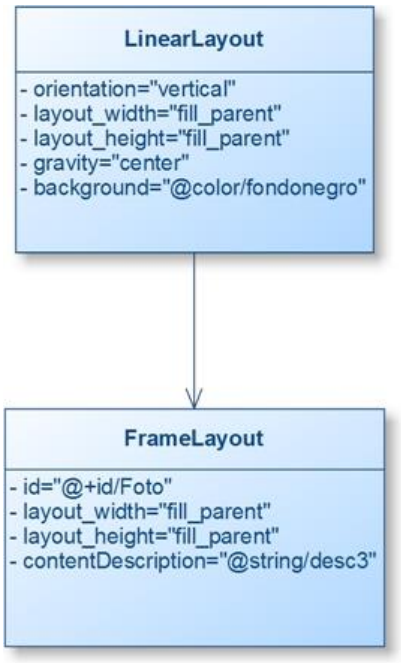



*Figura 3.43. Detalle de componentes del layout de previsualización*

**Pantallas de resumen**

Para la implementación de las pantallas que facilitan el resumen al usuario tanto de las instalaciones ya registradas en los servidores de Infortrex; como de las instalaciones que aún están pendientes de enviar, ambos se van a basar en el mismo layout. Puede verse la estructura de este layout en la Fig 3.44.. Como aspecto importante a destacar está el elemento raíz, consistente en un LinearLayout con orientación vertical en el que se integrará un elemento ListView.

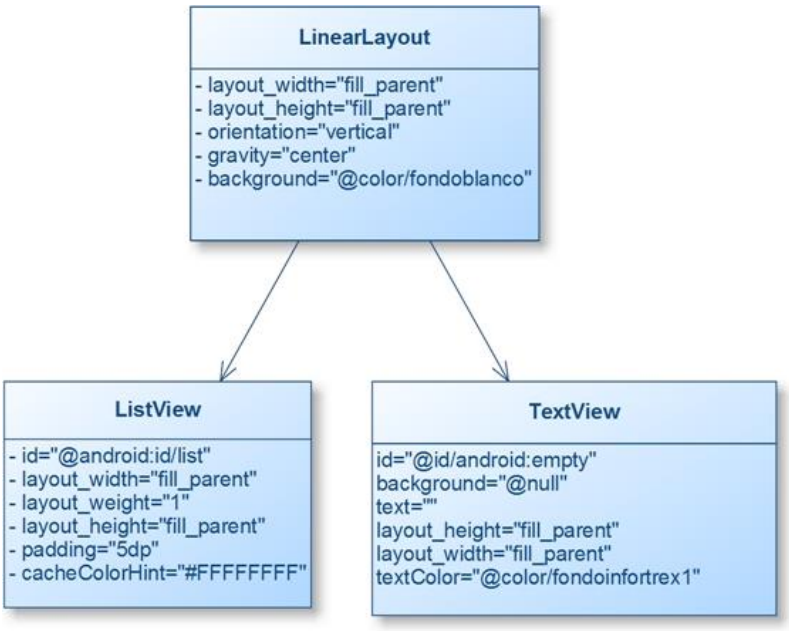



*Figura 3.44. Detalle de componentes de layouts de resumen*

La diferencia entre ambas pantallas de resumen radica en el elemento que compone este ListView, Fig 3.45 y Fig 3.47.

Primeramente se analiza el desarrollo abarcado para la pantalla de resumen de las instalaciones registradas. En este caso es necesario que el usuario sea capaz de observar varios datos. Según el diseño presentado previamente, se definió que cada componente contendría dos partes importantes: la sección de la izquierda en la que se mostrarían fecha y hora y la sección de la derecha donde estarían presentes todos los datos relacionados a la captura llevada a cabo por el usuario.

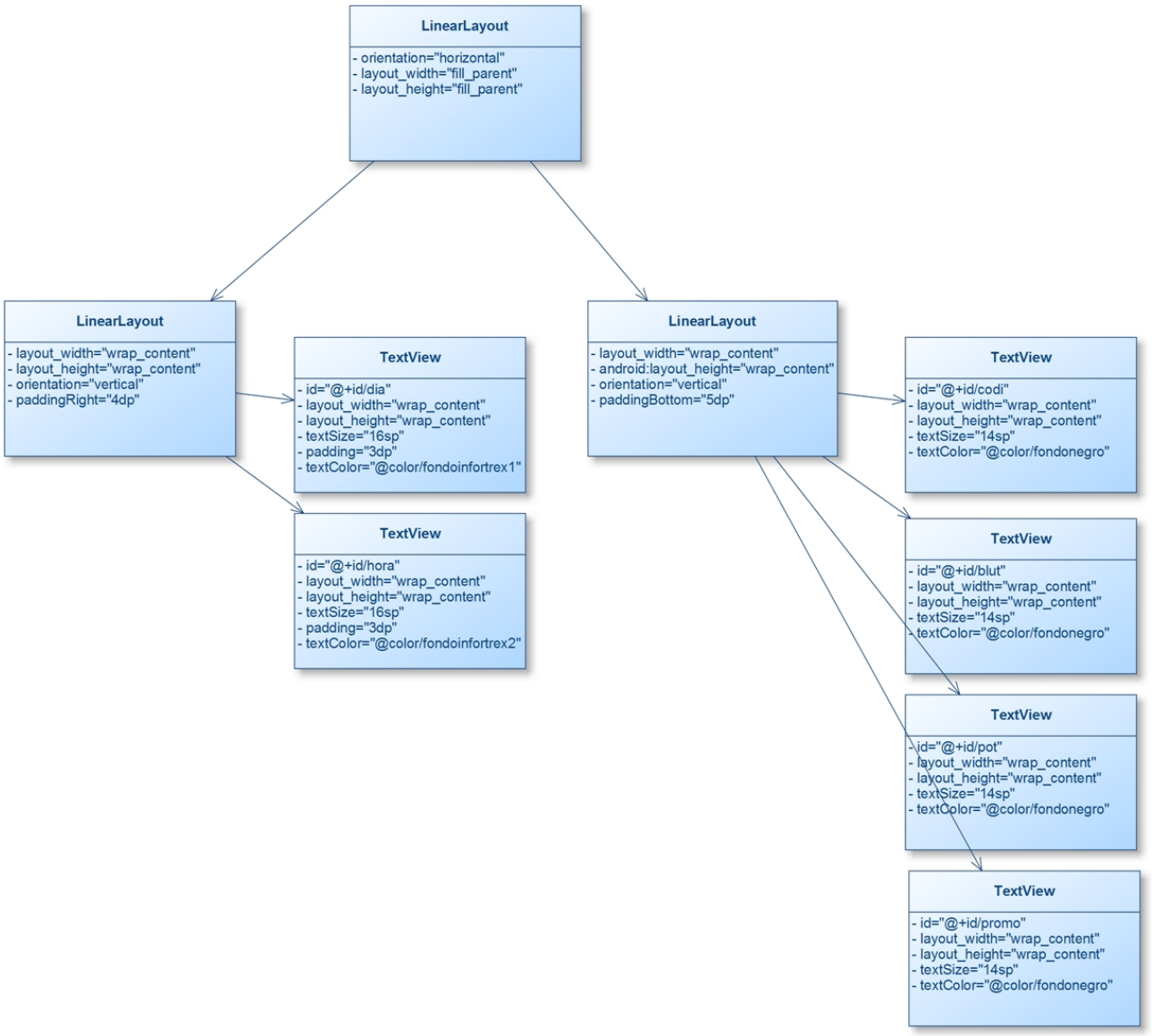



*Figura 3.45. Detalle del componente ListView del layout de resumen de registros*

La estética de la pantalla se ve reflejada en la Fig 3.46., esta imagen representa el formato de un listado de esta vista en la que se han registrado tres elementos cualesquiera que sean. En cada elemento se puede observar la información que contendrá y el color de texto, coherente con los colores característicos de la empresa.


*Figura 3.46. Layout de pantalla de resumen de registros*

Y para la pantalla encargada de presentar el listado de instalaciones pospuestas, la implementación de cada elemento únicamente requiere la configuración de un LinearLayout con orientación vertical que se compone de dos TextViews donde, según la fase de diseño, se ocuparán con la fecha de realización de la instalación y de la descripción introducida por el usuario.

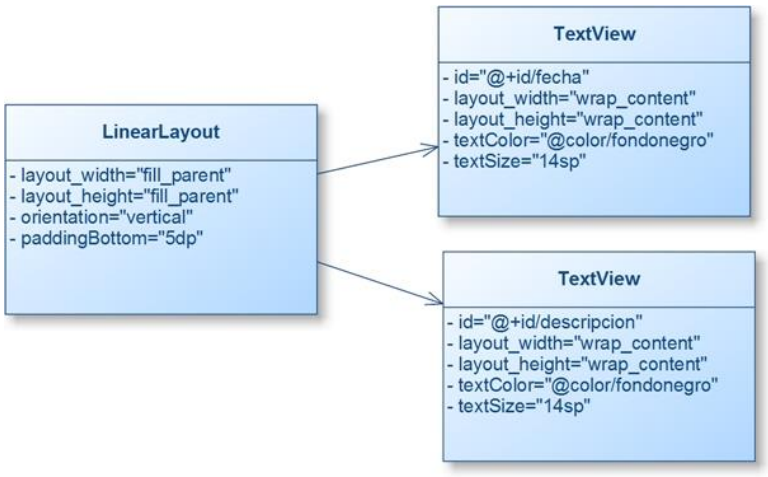



*Figura 3.47. Detalle del componente ListView del layout de resumen de instalaciones pospuestas*

La pantalla resultante, que se captura en la Fig 3.48., presenta un aspecto sencillo en el que tan solo se podrán observar la fecha y la descripción tal. Al igual que en la pantalla anterior también se ha establecido, a modo de ejemplo visual, un listado compuesto de tres elementos.


*Figura 3.48. Layout de pantalla de resumen de instalaciones pospuestas*

Uno de los aspectos importantes en el desarrollo de aplicaciones en Android reside en la independencia de la implementación de layouts y de actividades siempre que se disponga de una base común en la que centrarse. Como se ha podido observar en esta sección, las pantallas han podido ser creadas sin necesidad de implementar el código Java necesario para la gestión de pantallas, cuestión que se tratará en el siguiente apartado.

### *3.3.3 Implementación de actividades*

En esta sección se abordará la implementación de las actividades encargadas de gestionar la lógica de los layouts creados en el apartado anterior. La explicación del desarrollo se abordará analizando de una manera esquemática toda su funcionalidad, entrando en detalle en los aspectos de mayor interés para cada actividad creada.

Se emplea un fichero Java para cada pantalla en la mayoría de los casos, estos ficheros son ubicados dentro de la carpeta “src”2 y dentro del paquete *com.outsidebuilder.instaladores,* creado especialmente para este proyecto. Para facilitar el entendimiento de la lógica de los métodos más importantes de cada clase se presenta un diagrama de flujo con el detalle de su funcionamiento y una explicación textual en el caso de que se considere apropiado.

**Pantalla de autenticación**

La clase encargada de gestionar el proceso de autenticación, denominada “InstaladoresActivity” como toda actividad Android debe heredar de la clase Activity, representa el inicio de ejecución de la aplicación y por ello se establece en el fichero de configuración principal de la aplicación “AndroidManifest.xml” del siguiente modo:

```
<activity
      android:name=".InstaladoresActivity"
      android:label="@string/app_name"
      android:icon="@drawable/iconoinfortrex"
      android:screenOrientation="portrait">
      <intent-filter>
          <action android:name="android.intent.action.MAIN"/>

          <category android:name="android.intent.category.LAUNCHER"/>
      </intent-filter>
</activity>
```

En la anterior sección de código se referencia a la actividad principal que el lanzador de aplicación se debe ejecutar, también se identifica el icono de la aplicación y la orientación de la pantalla es fijada a la posición vertical. A diferencia del resto de actividades que se implementan en estos casos la definición en el AndroidManifest exclusivamente se limita a la siguiente declaración:

```
<activity
    android:name=".ActividadX"
    android:screenOrientation="portrait">
</activity>
```

La estructura de la clase, la cual se puede observar en la Fig 3.49., consiste principalmente en la sobrescritura de algunos de los métodos de su clase padre Activity que requieren ser modificados, implementar el método “connect” y la creación de una clase anidada para gestionar las peticiones con los servidores de Infortrex. A continuación se listan la utilidad de cada uno de ellos haciendo especial hincapié en el método connect().

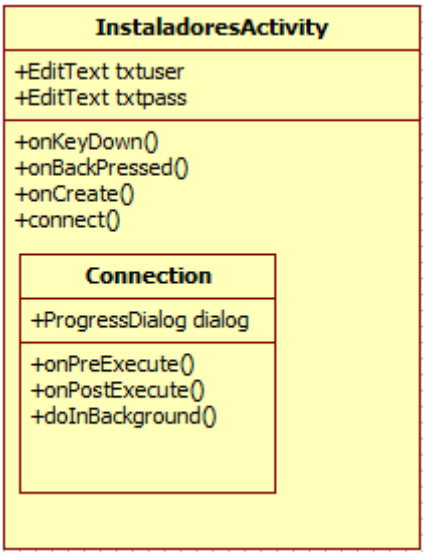



*Figura 3.49. Estructura de clase InstaladoresActivity*

*public boolean onKeyDown(int keyCode, KeyEvent event)*: En este método se evalúa cualquier pulsación de una tecla física del terminal, tratando dicho evento de forma que se finalice la aplicación (finish()) en el caso de que el parámetro *keyCode* conlleve el valor reservado para identificar la tecla de “volver”. En cualquier otro caso el comportamiento será el que ya se realizaba en la clase Activity.

*public void onBackPressed()*: Debido a que el control de los eventos de pulsación de esta tecla ya se controla en el anterior método, la sobrescritura de este método conlleva la eliminación del control que pudiera llevar el método de la clase padre.

*public void onCreate(Bundle savedInstanceState):* Este método representa por defecto en Android el primer método que se ejecuta, de cada actividad, al ser invocada; por lo que se empleará para configurar la actividad y establecer valores almacenados anteriormente en el terminal. En este caso se emplea principalmente para identificar el botón de logado y asignarle la función de hacer login así como de recoger datos de un logado anterior que hubiera sido satisfactorio. El diagrama de flujo del método onCreate() se detalle en la Fig 3.50..

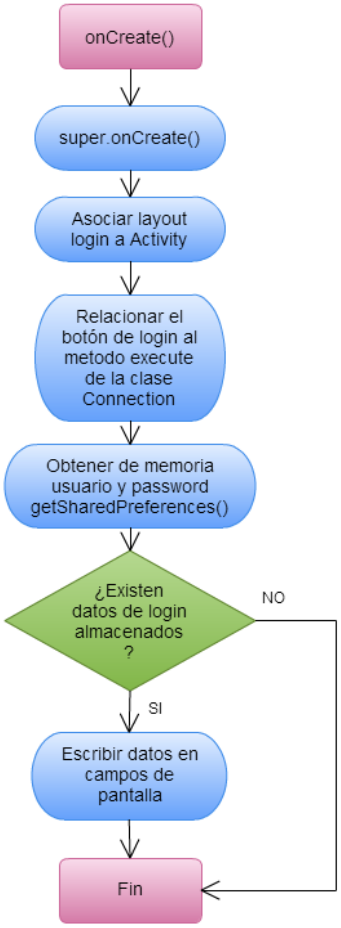



*Figura 3.50. Diagrama de flujo de método onCreate()*

*private String connect()*: Este método implementa el lanzamiento de la petición de logado al servidor de Infortrex, en el que se alojan los servicios REST que más adelante se desarrollan, y en caso de que el proceso se haya llevado a cabo correctamente se invocará la siguiente actividad. A continuación se adjunta la Fig 3.51. que respresenta el diagrama de flujo de este método. En ella se puede observar la conformación de la petición HTTP[30] [42] cuyo formato de URL respeta los requisitos explicados en apartados anteriores y que se explicará algo más en profundidad en el siguiente capítulo. Las respuestas son recibidas con formato JSON, el cuál emplea un formato muy ligero de representación de objetos Java. En esta memoria del proyecto no se entra a explicar la estructura de un JSON ya que se escapa a los objetivos de la realización del proyecto pero en la bibliografía se puede acceder a recursos que facilitan la comprensión de tal modelo.

[30] Hypertext Transfer Protocol

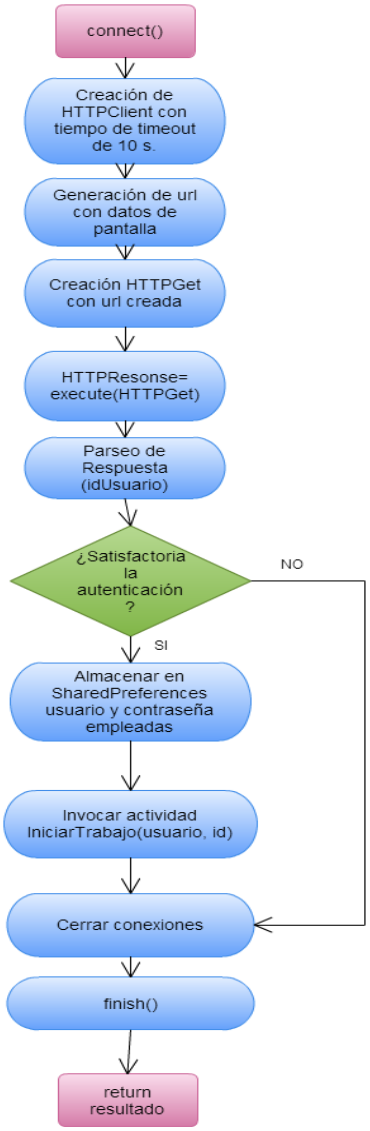



*Figura 3.51. Diagrama de flujo de método connect()*

Como aspecto importante de una de las etapas del diagrama de flujo se observa que en la invocación de la siguiente actividad se incluyen como parámetros el identificador de usuario que se ha obtenido como respuesta, el nombre de usuario para lanzar futuras peticiones al servidor entre otras utilidades y un valor que determina que la actividad de origen es el login para a continuación lanzar un mensaje de bienvenida al usuario.

Al ser necesario el mostrar al usuario una ventana de progreso mientras el proceso de logado se lleva a cabo, ha sido necesario implementar una clase anidada en la clase “InstaladoresActivity” que lanza el método connect() anteriormente explicado. Esta clase se denomina “Connection” y hereda de la clase AsyncTask, clase que se emplea en Android para realizar funciones en segundo plano. El funcionamiento real de esta clase se resume en la Fig 3.52..

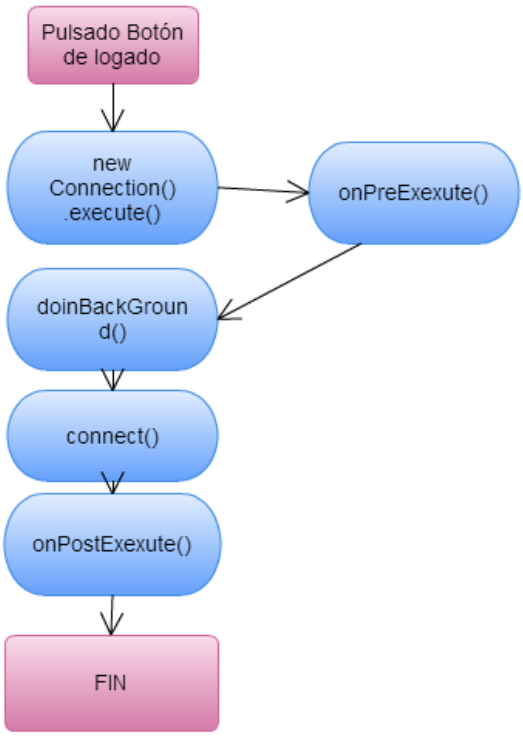



*Figura 3.52. Diagrama de flujo de proceso de login*

Todos estos métodos son una sobrescritura de los métodos de igual nombre de la clase padre y a continuación se detalla su función:

*protected void onPreExecute()*: Este método realiza el lanzamiento de una ventana de información al usuario mediante la siguiente sentencia que se adjunta a modo de referencia. Está sección de código es especialmente importante ya que en otras actividades también se emplea para informar al usuario.

```
ProgressDialog.show(DecompringInstaladoresActivity.this,"Info","Iniciando
Sesion",true, false);
```

Analizando la sentencia se puede observar entre sus parámetros: la actividad que lanza la ventana informativa, el texto del título de la ventana, el texto en el contenido, que el tiempo es indeterminado y que no es cancelable ya que el final vendrá dado por la respuesta a la petición lanzada.

*protected void onPostExecute(Object o)*: este método recibe si el proceso de logado ha sido satisfactorio gracias a la respuesta del método doInBackGround. En caso de no ser satisfactorio se emite un mensaje al usuario indicándole de esta eventualidad ocurrida.

*Protected Object doInBackground (Object… arg0):* Este método es el encargado de lanzar el método connect() de la clase InstaladoresActivity mientras hace viable que se muestre el mensaje de progreso del proceso de autenticación al usuario.

Es importante destacar que en esta actividad se necesita activar, en los permisos de la aplicación, la utilización de la conexión a Internet mediante la siguiente sentencia en el AndroidManifest. Como ya se ha indicado anteriormente este archivo debe registrar todos los permisos que la aplicación requiere para ser instalada.

```
<uses-permission android:name="android.permission.INTERNET" />
```

Para el funcionamiento correcto de las demás actividades además de este permiso se debe informar del uso de la cámara de fotos, del Bluetooth, de lectura y escritura de almacenamiento externo y de llamadas telefónicas mediante sentencias similares.

## Pantalla con menú inicial

La actividad encargada de gestionar el control de la pantalla de menú de la aplicación es IniciarTrabajoMenu, Fig 3.53., al igual que la actividad encargada del login esta actividad también hereda de la clase Activity y por lo tanto deberán ser sobrescritos algunos de sus métodos y creados nuevos métodos encargados de proporcionar la funcionalidad esperada.

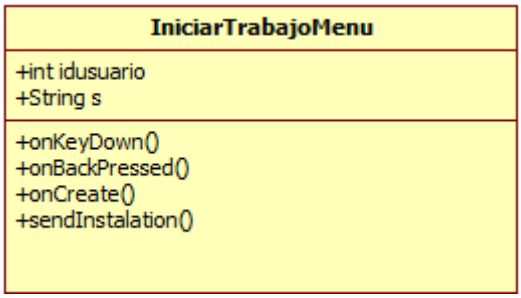



*Figura 3.53. Estructura de clase IniciarTrabajoMenu*

*onKeyDown() y onBackPressed()*: Los métodos son empleados en este caso para finalizar la actividad e invocar la actividad raíz de la aplicación. Esto significa que en caso de pulsar el botón volver del terminal, durante la presencia del menú inicial de la aplicación, se volverá a iniciar la ejecución del método onCreate de la actividad responsable del login de la aplicación.

*public void onCreate(Bundle savedInstanceStat)*: en este método se controla el inicio de la ejecución de la actividad. Este método es invocado exclusivamente en cualquiera de los siguientes casos y trata cada uno de ellos del siguiente modo:

- Tras realizarse un login correcto: La aplicación muestra el siguiente mensaje de Bienvenida, "*Usuario, Bienvenido a la aplicación para instaladores*". Donde la cadena "Usuario" es sustituida por el nombre del usuario.

- Tras intentar enviar una orden de registro de instalación. La aplicación debe comprobar si la ejecución de la orden de registro ha sufrido algún error durante todo el proceso gracias a un parámetro que la actividad anterior pasa como parámetro en la ejecución de la actividad del menú de inicio con el fin de informar al usuario del error en el caso de que lo hubiera habido.

- Tras posponer un registro de instalación: En este caso no realiza ninguna acción especial.

- Tras pulsar la tecla volver del terminal en cualquiera de los dos asistentes y en la consulta del historial de registros del usuario. En este caso no se realiza ninguna acción especial.

Una vez realizadas todas estas comprobaciones explicadas, el método registra las acciones a realizar para los eventos de pulsado de los diferentes botones existentes en la pantalla. Los casos existentes son los siguientes:

- Botón de llamada telefónica: En el caso de pulsar este botón la aplicación inicia una llamada telefónica al número de teléfono de las oficinas de Infortrex. La implementación es sencilla, simplemente se lanza una activity en donde el

Intent lleva como parámetros la constante de iniciar una llamada de teléfono denominada *ACTION_CALL* y el número de teléfono a marcar. El código que a continuación se adjunta va insertado en el método onClick del evento tratado.

```
Intent intent = new Intent(Intent.ACTION_CALL, Uri.parse(url));
startActivity(intent);
```

- Botón de instalación tipo Bluetooth: En este caso la acción a realizar es la invocación de la actividad encargada de controlar al asistente de instalaciones de esta tipología. Pasando como parámetros en la invocación de la actividad el identificador de usuario, el nombre del usuario y un flag indicando que la invocación proviene de este menú que se está explicando.

- Botón de instalación tipo Bluetooth y pegatina promocional: En este caso la acción a realizar es idéntica en las formas al evento anterior salvo que el asistente que se invoca es el encargado de este tipo de instalaciones.

- Botones de consulta de historial y botón de registro de instalaciones pospuestas: En estos casos únicamente se lanza la ejecución de su actividad sin necesidad de indicar la procedencia de la llamada.

A continuación se comprueba si existen instalaciones pendientes almacenadas en la memoria de la aplicación, habilitando el botón encargado de facilitar el registro de estas instalaciones pospuestas y adicionalmente se crea un hilo de ejecución independiente que se encarga de intentar registrar las instalaciones pospuestas de forma automática, de forma que se ejecuten los diferentes intentos cada cinco minutos, el núcleo del método "run" del hilo llama al método llamado "sendInstalation" que a continuación se explica.

*public boolean sendInstalation()*: Este método se encarga de lanzar las peticiones de registro para el hilo encargado de gestionar el registro de instalaciones automáticas. En el caso de que se haya llevado a cabo satisfactoriamente todas las instalaciones de forma automática se informa al usuario mediante una notificación en su terminal de que ya no dispone de instalaciones pendientes y se deshabilita el botón de registrar instalaciones pospuestas de la pantalla.

El lanzamiento de una notificación al usuario se realiza mediante la clase NotificationManager que permite mediante la ejecución de su método notify() informar al usuario sobre cualquier evento en la barra de notificaciones del terminal.

El proceso de envío de órdenes de registro al servidor se realiza del mismo modo al que más tarde se explicará en detalle en cada uno de los asistentes disponibles. Por este motivo no se entrará en detalle sobre este asunto en estos momentos.

### Pantalla de instalación tipo Bluetooth

La actividad encargada de controlar el asistente que guía al usuario en el proceso de registro de instalaciones se denomina “IniciarTrabajoBT”, como todas las actividades esta también hereda de la clase Activity. Como aspecto a destacar es que al igual que la actividad encargada del login esta nueva actividad también dispone de una clase anidada encargada de ejecutar en un segundo plano la orden de registro mientras la pantalla muestra el progreso del envío. La forma de gestionar esta clase es idéntica a como se realizó anteriormente, por este motivo no es especialmente interesante el entrar en un mayor detalle de nuevo. A modo de resumen el contenido de esta actividad, se muestran en la Fig 3.54., los atributos y método que conforman la clase.

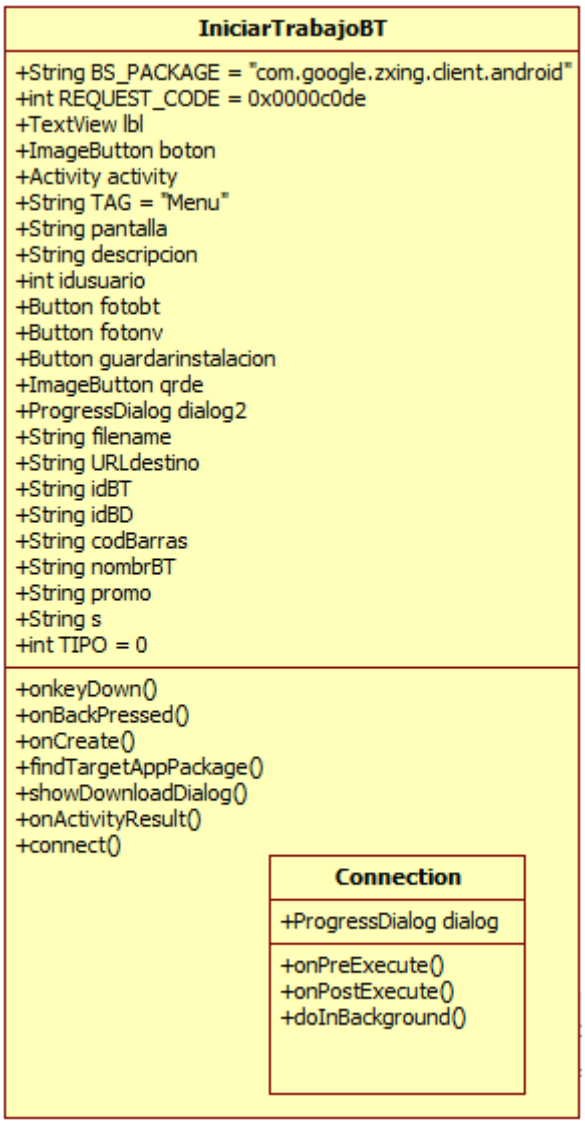



*Figura 3.54. Estructura de clase IniciarTrabajoBT*

A continuación se detalla la implementación de los métodos presentes en la actividad.

*onKeyDown() y onBackPressed()*: Los métodos son empleados en este caso para finalizar la actividad y regresar a la actividad encargad de controlar el menú de la aplicación. Esto significa que en caso de pulsar el botón “volver” del terminal durante la

presencia del asistente de instalación tipo Bluetooth de la aplicación se volverá a iniciar la ejecución del método onCreate de la actividad responsable del menú de la aplicación.

*public void onCreate(Bundle savedInstanceStat)*: en este método se controla el inicio de la ejecución de la actividad. Este método es invocado exclusivamente en cualquiera de los siguientes casos que a continuación se comentan:

- Mediante el menú de inicio: Gracias al flag pasado por parámetro a la invocación de esta actividad se identifica que la procedencia de la ejecución es el menú de inicio.

- Tras capturar información guiada por esta actividad

A modo de explicación de este método se puede observar su funcionamiento en la Fig 3.55..

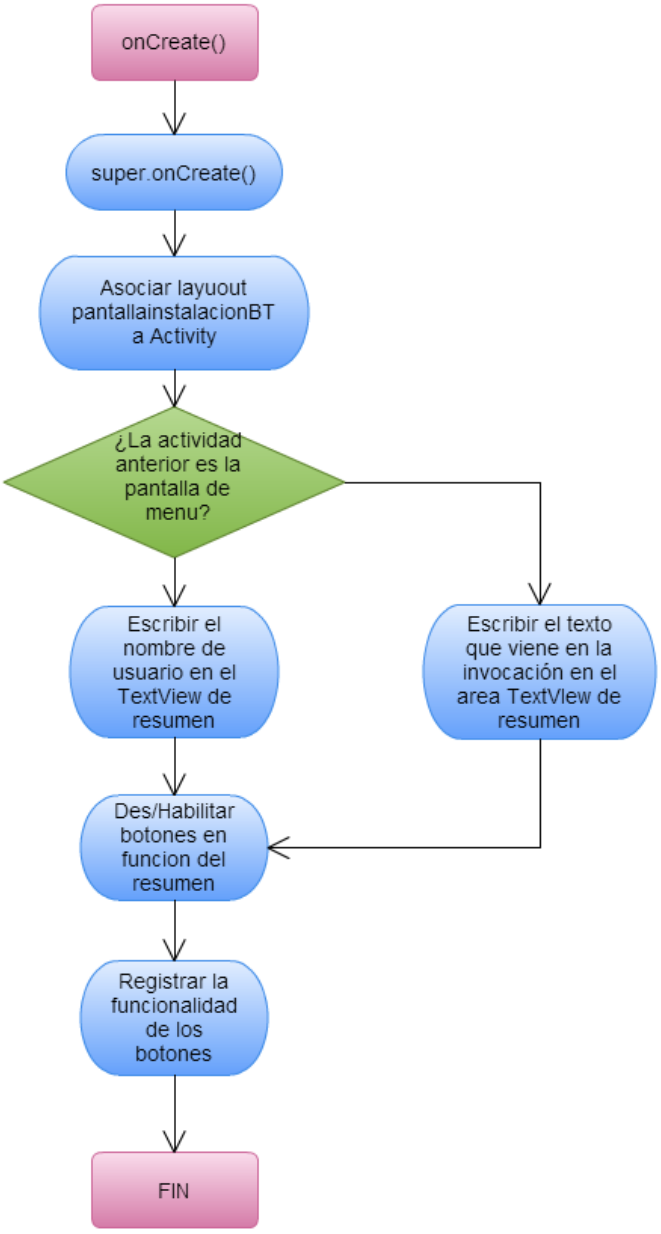



*Figura 3.55. Diagrama de flujo del método onCreate*

En el registro de funcionalidad de los diferentes botones nos encontramos con los siguientes casos:

Botones referentes a escaneo de códigos Bidi: En función del botón pulsado, entre los que es necesario leer un código Bidi, se fija el valor a un atributo de clase que nos permite identificar que parte de la captura de datos se va a llevar a cabo, conociendo de esta

forma el tipo de validación a realizar y el formato en el que se va a incluir la información capturada en el campo de texto en el que se muestra el resumen del asistente así como los botones que son necesarios habilitar o deshabilitar en cada caso.

Es importante señalar que para la lectura de todos los códigos Bidi va a ser necesario disponer, como aplicación instalada en el terminal, de la aplicación “BarCodeScanner” presente en el mercado de aplicaciones Android. Por ello en el momento de pulsar cualquiera de estos botones es necesario consultar si se dispone de esta aplicación instalada y en el caso de que no se disponga de la misma se guiará al usuario para descargar esta aplicación. El código que implementa estos aspectos resulta especialmente interesante de mostrar ya que se necesitará ejecutar durante todo el proceso de registro cada vez que se intente leer un código.

```
            Intent intentScan = new
Intent("com.google.zxing.client.android.SCAN");
            intentScan.putExtra("PROMPT_MESSAGE", "Escanee el codigo que figura
en la hoja de pedido");
            String targetAppPackage = findTargetAppPackage(intentScan);
            if (targetAppPackage == null) {
                  showDownloadDialog();
            } else startActivityForResult(intentScan, REQUEST_CODE);
```

En este código se pueden observar principalmente tres invocaciones importantes que, como se ha podido observar en el resumen de los métodos y atributos de la clase, se explicarán en detalle más adelante dos de ellos. Estos métodos son targetAppPackage() y showDownloadDialog mientras que la llamada a startActivityForResult() ejecuta la actividad de la aplicación BarCodeScanner que permite escanear los códigos Bidi y posteriormente ofrecernos la información capturada mediante el método onActivityResult(), para identificar que el resultado obtenido del escaneo corresponde a la petición de iniciar la actividad principal de la aplicación Barcode Scanner se introduce el parámetro REQUEST_CODE que posteriormente en la respuesta se comprobará.

Botón de envío de instalación: Este botón se implementa para que realice los primeros pasos en lo que conforma la función principal de la aplicación, consistente en el envío de toda la información capturada, incluyendo la fotografía. Primeramente se parsea toda la información existente en el texto de resumen del asistente de la instalación para que de este modo se pueda conformar la URL con todos los datos a registrar, a continuación se hace una limpieza de todos los datos capturados ya que no va a ser necesario volver a disponer de ellos. Posteriormente se lanza una ventana de dialogo al usuario para que introduzca opcionalmente un texto descriptivo sobre la instalación que acaba de llevar a cabo y una vez finalizado este paso se ejecuta el envío de igual forma que se realizó en su momento la petición de autenticación contra el servidor, a través de la clase Connection, la ventana de progreso correspondiente y el método connect() que más adelante se explica.

Botón de guardado de instalación: La funcionalidad de este botón actúa de forma similar a como lo hace el botón de envío salvo con la diferencia de que no ataca a la clase Connection sino que en su lugar se almacenan todos los datos que se requieren enviar posteriormente, como la URL de envío y la ruta a la fotografía captura, de la misma forma que anteriormente se explicó para el guardado en memoria de los datos de logado, y se actualizan las variables, también existentes en memoria, que permiten a la aplicación conocer si existen instalaciones sin registrar en memoria que emplea la actividad encargada del menú.

Botones de revisión de fotografía, detección de modulo Bluetooth y captura de fotografía : En estos casos únicamente se lanza la ejecución de su actividad correspondiente pasando como parámetro el identificador y el nombre del usuario; y un flag que permite a la actividad invocada conocer quien ha sido la originante de la llamada.

*private String findTargetAppPackage(Intent intent)*: como ya se ha podido ver en el código implementado para el método onClick() de los botones que requieren del soporte

de la aplicación BarCodeScanner, este método es invocado para determinar si dicha aplicación se encuentra instalada en el terminal del usuario. En cuanto a la implementación de este método no se ha considerado su explicación en detalle ya que se entraría en un muy bajo nivel y esto no representa el objetivo de esta memoria.

private AlertDialog showDownloadDialog(): Este método actúa en el caso de que en la invocación al método findTargetAppPackage() se haya obtenido una respuesta negativa, es decir que no exista la aplicación BarCodeScanner instalada en el terminal. En cuyo caso se le ofrece al usuario la opción de descargar la aplicación guiándole hasta el lugar adecuado dentro del mercado de Android. La invocación de la actividad que permite hacer esto se ha implementado de la siguiente manera, siendo especialmente interesante la manera de acceder a contenido del mercado de aplicaciones:

```
Uri uri = Uri.parse("market://details?id=com.google.zxing.client.android”);
Intent intent = new Intent(Intent.ACTION_VIEW, uri);
activity.startActivity(intent);
```

*public void onActivityResult(int requestCode, int resultCode, Intent intent)*: la implementación de este método consta del procesamiento de la respuesta obtenida tras el escaneo de cualquiera de los códigos Bidi necesarios para el registro de una instalación. A continuación se muestra un diagrama, Fig 3.56. en el que se muestra un esquema simplificado del código de este método.

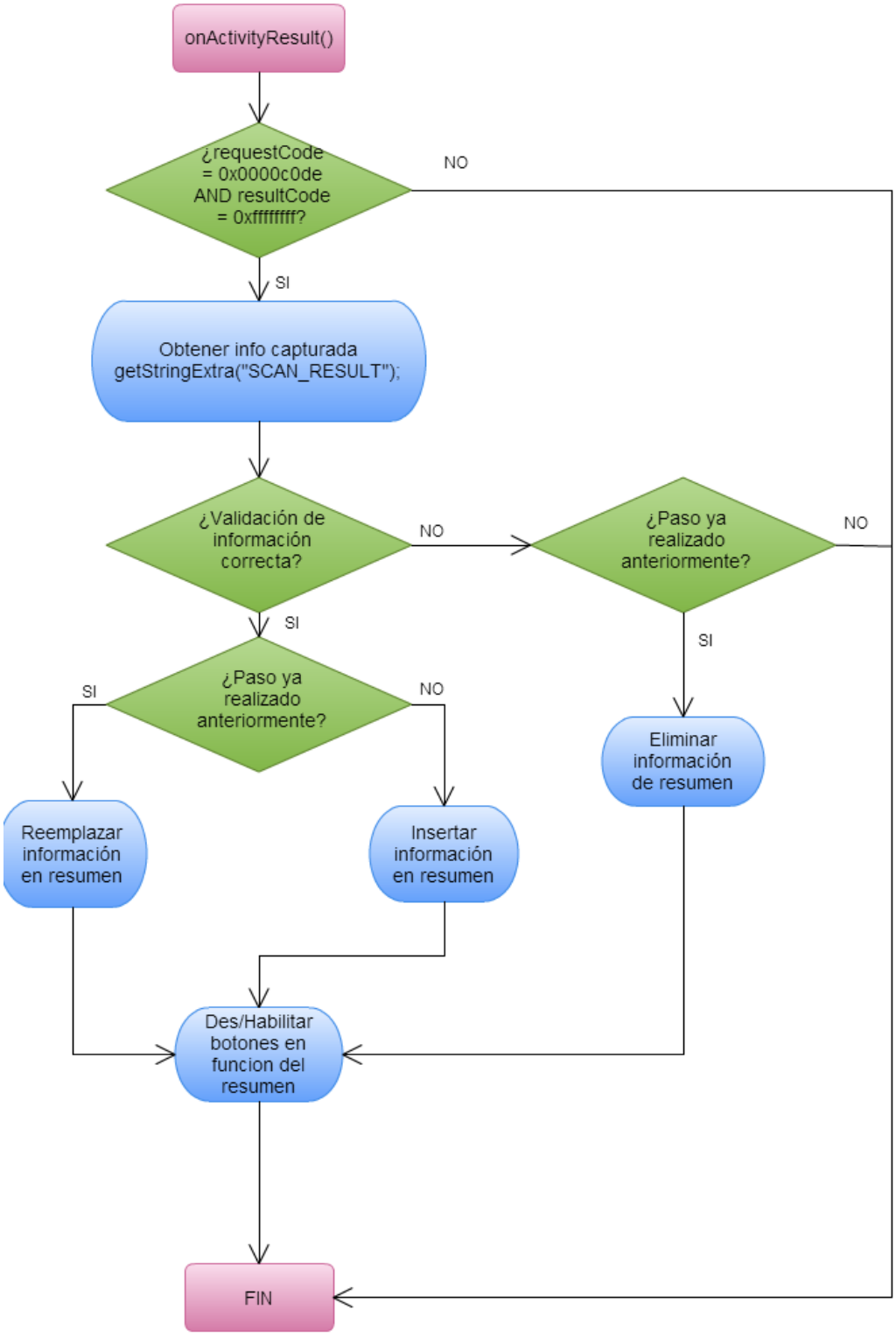



*Figura 3.56. Diagrama de flujo del método onActivityResult*

Como aspecto a aclarar del diagrama de flujo, dos de los parámetros con los que se invoca al método son los códigos que hacen referencia al identificador de petición, puesto para todas las peticiones al valor “0x0000c0de” por defecto y al flag con el resultado de la respuesta, donde un valor de “0xffffffff” significa que el proceso de escaneo ha sido correcto. Por esta razón en el método que se está describiendo la primera comprobación se encarga de estos dos valores.

*private String connect()*: Este método implementa el lanzamiento de la petición de registro al servidor de Infortrex. El contenido de este método se basa en el método ya explicado en la actividad inicial de logado, con las diferencias de que en este caso la petición se construye sobre un objeto *HttpPost* en lugar de un *HttpGet* ya que va a ser necesario incluir en el cuerpo de la petición el archivo con la imagen obtenida con la cámara fotográfica.

En el caso de que el proceso de registro no se pueda realizar por alguna razón de un modo correcto, la instalación se almacena en la memoria del terminal y se gestiona de igual modo que si hubiera sido pospuesta de un modo directo por el usuario. De cualquier modo se invoca nuevamente a la actividad encarga de gestionar el menú principal de la aplicación.

## Pantalla de instalación tipo Bluetooth junto a pegatina promocional

La implementación de esta actividad se basa en la actividad anteriormente comentada para la pantalla del asistente de instalaciones de tipo Bluetooth. La clase se compone de los mismos atributos y los mismos métodos que la clase anterior excepto la diferencia de que se añade un botón adicional que actúa bajo el soporte de la aplicación “BarcodeScanner”.

Es importante destacar que todas las llamadas a nuevas actividades desde esta pantalla deben contener un identificador que permite identificar que la actividad origen es uno u otro de los asistentes.

## Pantalla de enlace con módulo Bluetooth

La actividad encargada de controlar la pantalla que permite detectar el dispositivo Bluetooth instalado y obtener la información que se requiere de este dispositivo se denomina “AndroidBluetooth”, como todas las actividades esta también hereda de la clase Activity. En la Fig 3.57. se pueden observar tanto los atributos como los métodos implementados para cumplir con la funcionalidad establecida y que posteriormente se explican con una mayor profundidad.

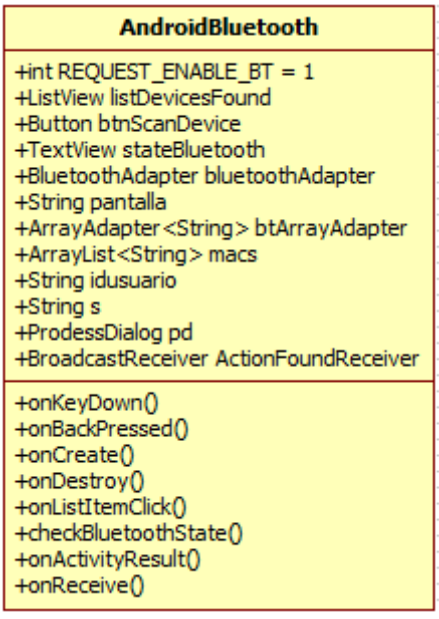



*Figura 3.57. Estructura de clase AndroidBluetooth*

Es importante dejar constancia en estos momentos de que el botón llamado “btnScanDevice” tiene definido como comportamiento, al ser pulsado, ejecutar el método “startDiscovery” de la clase BluetoothAdapter el cual arranca la detección de dispositivos Bluetooth en las proximidades del usuario.

A continuación se detalla la implementación de los métodos presentes en la actividad.

*onKeyDown() y onBackPressed()*: Los métodos son empleados en este caso para finalizar la actividad y regresar a la actividad encargada de controlar el asistente que corresponda en cada caso. Esto significa que en caso de pulsar el botón volver del terminal durante la presencia del asistente de instalación tipo Bluetooth de la aplicación se volverá a reanudar la ejecución de la actividad responsable del menú del asistente.

*public void onCreate(Bundle savedInstanceStat)*: en este método se controla el inicio de la ejecución de la actividad. Este método es invocado exclusivamente cuando se pulsa el botón de enlazado de Bluetooth en cualquiera de los asistentes. Las acciones realizadas en el interior del método se resumen en la Fig 3.58. a modo de diagrama de flujo.

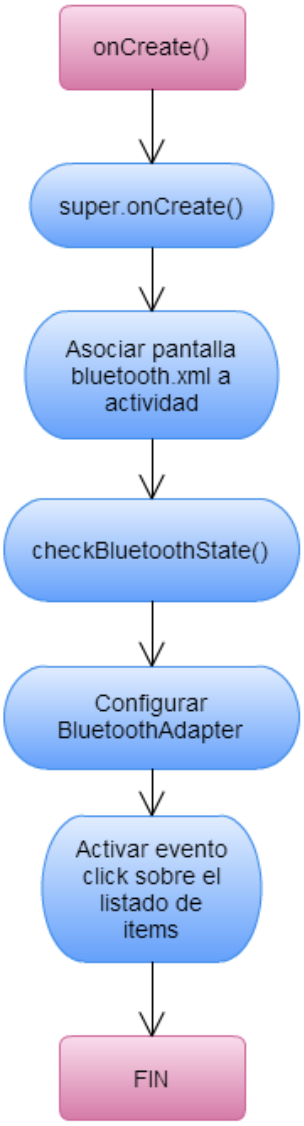



*Figura 3.58. Diagrama de flujo del método onCreate*

El núcleo de la configuración del adaptador de Bluetooth se realiza según puede observarse en las siguientes líneas de código:

```
   bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();

   //Captura del ListView de la vista de la pantalla
   listDevicesFound = (ListView)findViewById(R.id.devicesfound);

   //Creacion de un ArrayAdapter para listado de ítems
   btArrayAdapter = new ArrayAdapter<String>(AndroidBluetooth.this,
android.R.layout.simple_list_ítem_1);
```

```
//Asignacion del Array a la vista
listDevicesFound.setAdapter(btArrayAdapter);

//creacion del evento de click
btnScanDevice.setOnClickListener(btnScanDeviceOnClickListener);

//Filtrado de tipo de acciones a tratar
IntentFilter filter = new IntentFilter();
filter.addAction(BluetoothDevice.ACTION_FOUND);
filter.addAction(BluetoothAdapter.ACTION_DISCOVERY_FINISHED);

//Activacion de la gestion de eventos de Bluetooth
this.registerReceiver(ActionFoundReceiver, filter);
```

Como aspecto interesante, conveniente a ser explicado, de la anterior porción de código es el filtrado de las posibles acciones a recibir, siendo únicamente necesarias de controlar las que informan sobre la detección de módulos Bluetooth y la que indica que el proceso de detección ha finalizado.

*protected void onDestroy()*: este método únicamente invoca al método onDestroy de la clase padre y finaliza las acciones del Bluetooth.

*protected void onListÍtemClick (View v, int pos, long id):* este método se encarga de gestionar el pulsado de cualquier ítem de la lista de dispositivos Bluetooth detectados. En la implementación únicamente se detecta cual es el asistente al que regresar tras la pulsación y adicionalmente se complementa el resumen de la instalación con los datos capturados durante este paso.

*private void checkBluetoothState():* siempre se ejecuta al inicio de la actividad, es decir durante la ejecución del método *onCreate* y en él se establece el estado del Bluetooth del terminal del usuario con el fin de habilitar el botón de inicio del proceso de detección. Para ello se emplea el método *isEnabled* de la clase BluetoothAdapter y en el caso de que no

se encuentre habilitado el Bluetooth se inicia una actividad del sistema para facilitar al usuario la activación de esta característica en el terminal.

*private void onReceive(Context context, Intent intent):* En el interior de este método se gestiona la recepción de acciones. Al haber sido filtradas las posibles acciones o eventos recibidos, únicamente se deben controlar los siguientes casos:

- detección de dispositivo Bluetooth, en cuyo caso se deberá añadir la información que proporciona el dispositivo a un ítem del listado a mostrar.

- finalización del proceso de búsqueda, en cuyo caso se deberá ver si el listado contiene algún dispositivo Bluetooth detectado y en caso negativo se debe informar al usuario de que no ha sido posible localizar ningún elemento.

**Pantalla de captura fotográfica**

La actividad encargada de gestionar la tarea de capturar una imagen donde se pueda observar con claridad la instalación realizada se denomina “CameraActivity”, como todas las actividades esta también hereda de la clase Activity. Esta clase se apoya en otra clase adicional, que implementa la interfaz SurfaceHolder.CallBack y que hereda de la clase SurfaceView, para la representación de la imagen que la cámara va capturando en tiempo real. En la Fig 3.59. se pueden observar tanto los atributos como los métodos implementados para las dos clases que se han comentado.

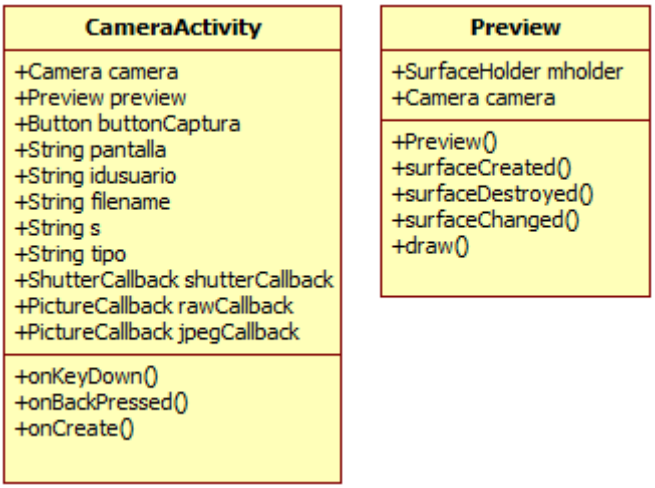



*Figura 3.59. Estructura de clases CameraActivity y Preview*

A continuación se describe en detalle la implementación de la funcionalidad de la pantalla mediante un análisis de las clases, que como ya se ha comentado, se ven involucradas.

*Clase CameraActivity*

*onKeyDown() y onBackPressed()*: Los métodos son empleados para anular la opción de almacenar una fotografía y de este modo regresar al asistente del registro de la instalación. La implementación de estos métodos es igual a las anteriormente descritas en el resto de actividades.

*public void onCreate(Bundle savedInstanceStat)*: la implementación de este método no tiene ningún aspecto excesivamente complejo, en él lo que se configura son los dos elementos de la vista que se ha asignado y que anteriormente ha sido explicada y cuyo código ya ha sido comentado. El layout se encuentra compuesto por un FrameLayout cuyo interior estará asignado a un objeto de la clase Preview de forma que sea la clase la encargada de refrescar su contenido y un botón de captura de imagen cuyo evento

relacionado con su pulsado invoca al método *takePicture* de la clase Camera de la librería android.hardware.

En el momento en el que se pulse el botón se opta por almacenar la imagen en formato “JPEG” por lo que automáticamente se invoca al método *onPictureTaken* del atributo *jpegCallback* y cuya funcionalidad ha sido implementada para que la imagen se almacene en una carpeta específica para la aplicación dentro de la siguiente ruta, “/sdcard/infortrex” y con un nombre de fichero aleatorio. Una vez se ha efectuado el almacenamiento se actualiza el componente del asistente en el que se muestra el resumen de los datos capturados y se invoca la actividad responsable del asistente que corresponda para continuar con el registro de la instalación.

*Clase Preview*

Para esta clase se ha empleado el código compartido en el repositorio cw-advandroid [43] alojado en la plataforma GitHub [44] bajo el proyecto de CameraPreview donde se comparte esta implementación como ejemplo de utilización.

A modo de explicación esta clase se encarga de gestionar un objeto “Camera”, responsable de establecer parámetros como el tamaño de la foto o registrar el evento de *callBack* empleado en la clase CamaraActivity ya descrito anteriormente; o de controlar el renderizado y posicionamiento de elementos en la pantalla mediante el atributo de la clase SurfaceHolder. Se considera que una descripción más detallada del funcionamiento de esta clase queda fuera del objetivo de esta memoria ya que, como se ha indicado anteriormente, este código se ha abierto a un repositorio público y es accesible fácilmente a cualquier interesado.

**Pantalla de visualización de fotografía**

La actividad encargada de mostrar una previsualización de la fotografía, ya almacenada previamente, se ha implementado bajo el nombre de “RevisarFotoActivity”, la estructura de esta clase puede observarse en la Fig 3.60.

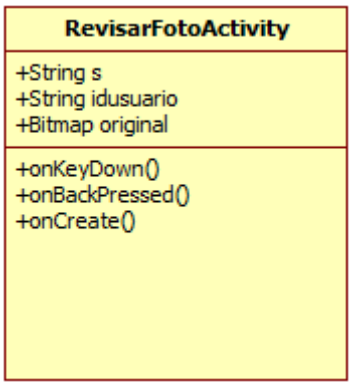



*Figura 3.56. Estructura de clase RevisarFotoActivity*

En esta actividad únicamente es preciso mostrar la fotografía cuya ruta viene almacenada en el resumen de datos capturados de la instalación del asistente correspondiente y para mostrar esta imagen dentro del componente ImageView asignado para tal comportamiento se hace uso de un atributo de tipo Bitmap.

A continuación se explica el funcionamiento de la implementación de sus tres métodos:

*onKeyDown() y onBackPressed()*: Los métodos son empleados para regresar al asistente del registro de la instalación. La implementación de estos métodos es igual a las

anteriormente descritas en el resto de actividades añadiendo la liberación de recursos del atributo de la clase Bitmap.

*public void onCreate(Bundle savedInstanceStat)*: la implementación del método *onCreate* para esta clase representa el más sencillo de toda la aplicación. En él se parsea el texto de resumen del asistente que es pasado como parámetro en Intent que invoca a la actividad de forma que se obtenga la ruta del fichero que se debe mostrar. Una vez obtenida esta información se ejecuta el método *decodeFile* de la clase BitmapFactory que decodifica una ruta dada en un objeto Bitmap y posteriormente se asigna al ImageView, del layout de la pantalla, el objeto Bitmap mediante el método setImageBitmap.

**Pantalla de historial de registros**

La actividad encargada de gestionar la pantalla con el listado de registros realizados por el usuario se denomina “HistorialActivity” y en este caso al tratarse de un listado de ítem, la clase no hereda directamente de la clase Activity sino de la clase ListActivity. La estructura de la clase se puede observar en la Fig 3.61., adjuntada a continuación.

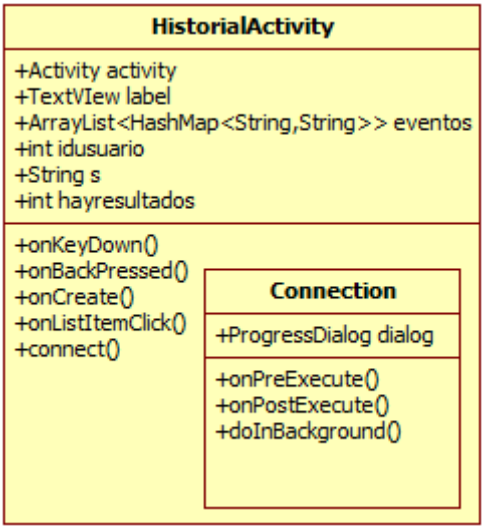



*Figura 3.61. Estructura de clase HistorialActivity*

Para implementar la opción de consulta del historial de registros es necesario realizar una consulta al servidor de Infortrex, de modo que se obtenga toda la información relevante de cada una de las instalaciones. Por ello se va a necesitar mostrar un ProgressDialog para informar al usuario de que la petición se encuentra en proceso de forma que se ha implementado la actividad según el resto de actividades de la aplicación de estas características, lo que influye en la creación de una clase Connection anidada encargada de ejecutar la consulta en un hilo en un segundo plano.

A continuación se detallan algunos aspectos importantes de la implementación de los métodos que difieren del funcionamiento habitual a lo ya explicado en anteriores actividades

:

*onKeyDown() y onBackPressed()*: Los métodos son empleados para regresar al menú inicial de la aplicación ya que este función de consulta se encuentra dentro de este menú. Su implementación exceptuando el destino de su ejecución no difiere en el resto de implementaciones que ya se han explicado.

*protected void onListÍtemClick (ListView l, View v, int position, long id)*: La función que cumple este método es la de mostrar un mensaje al usuario indicándole la descripción de la instalación que se introdujo en el momento de hacer el registro del ítem pulsado. Ya que en el momento del almacenaje siempre se va a almacenar un comentario, en caso de que el usuario no hubiera insertado la descripción, figurará el texto por defecto “No se ha introducido descripción”. Para mostrar correctamente el comentario introducido se conserva el orden en el que aparecen los registros en la pantalla y gracias al parámetro position del método se presenta el mensaje correspondiente.

*public void onCreate(Bundle savedInstanceStat):* Para esta actividad el método ejecutado al ser invocada la actividad, *onCreate*, únicamente almacena dentro de los atributos la información ofrecida en la intención pasada por parámetro , en este caso un identificador de usuario y el texto resumen del asistente de instalación; y ejecuta el método *execute* de la clase Connection de forma que la actividad trabaje del modo que se ha realizado en el resto de actividades similares.

*Clase Connection*

Esta clase, como ya se ha comentado en anteriores actividades se encarga de gestionar la ejecución de la petición de información contra el servidor. En ella se han implementado los siguientes métodos con la siguiente funcionalidad:

*protected void onPreExecute():* este método únicamente lanza la ventana de información al usuario indicándole que la petición de consulta de historial se encuentra en proceso.

*protected String doInBackground(Object... arg0)*: método encargado de invocar el método connect() de la actividad.

*protected void onPostExecute(Object sb*): responsable de ordenar la información del servidor a traves del JSON recibido en la respuesta y ubicarla correctamente dentro del componente correspondiente del Layout establecido. Para ello se introducen los datos de cada registro en un HashMap<*String, String*> tras obtener dichos datos del parseo del siguiente modo:

```
Eventos = new ArrayList<HashMap<String, String>>();
```

```
   HashMap<String,String> datosEvento=new HashMap<String, String>();

   datosEvento.put("fecha", evento[0]);…datosEvento.put("promo", "Bidi
Promocion:\n"+evento[7]);

   Eventos.add(datosEvento);
```

Posteriormente se incorpora el ArrayList Eventos en el que se encuentra toda la información a mostrar en las columnas que corresponden dentro del ListView.

```
   String[] from=new String[] {"fecha","hora","codigo","bt", "bt4", "promo"};

   int[] to=new int[]{R.id.dia,R.id.hora,R.id.codi,R.id.blut, R.id.pot,
R.id.promo};

   SimpleAdapter ListadoAdapter=new SimpleAdapter(HistorialActivity.this,
Eventos, R.layout.row, from, to);

   setListAdapter(ListadoAdapter);
```

*private String connect()*: Este método implementa el lanzamiento de la petición de consulta al servidor de Infortrex y la posterior recepción de la respuesta. El contenido de este método se basa en el método ya explicado en la actividad inicial de logado.

## Pantallas de registros pospuestos

La implementación de esta actividad, denominada “PospuestasActivity”, se basa en dos actividades ya desarrolladas y que ya han sido explicadas en detalle. A modo de resumen, el contenido de esta actividad se muestra en el listado de atributos y métodos que pueden observarse en la Fig 3.61..

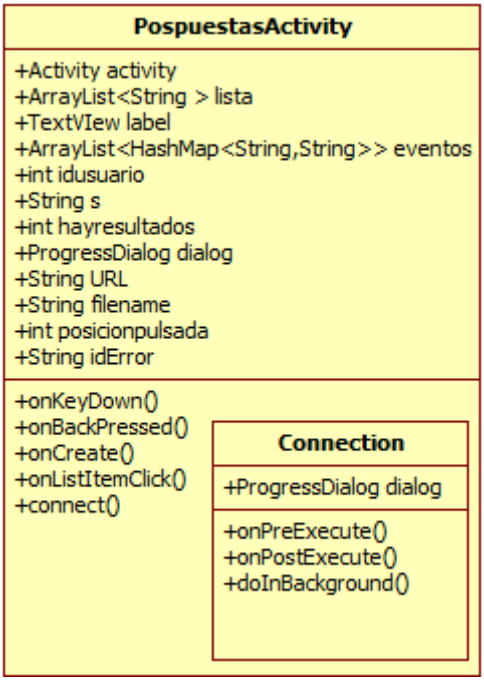



*Figura 3.61. Estructura de clase PospuestasActivity*

Como se observa, el contenido de la clase es equivalente al que se ha comentado para la actividad encargada de mostrar el historial de registros del usuario. Los cambios existentes en la actividad PospuestasActivity son de funcionalidad pero no de estructura. A continuación se resume en que parte de la clase se ha implementado cada operación.

Carga de listado de instalaciones pospuestas: Al no ser necesario solicitar información al servidor de Infortrex como en el caso de la pantalla de historial, en esta actividad la manipulación de los datos almacenados se realizará en el método *onCreate* mientras se muestra un mensaje informativo para avisar al usuario de que el procesamiento se está llevando a cabo. En el listado de datos que se forme se van a asignar al evento de pulsado sobre cualquier ítem las opciones de enviar instalación manualmente y la de eliminar de la memoria. En el caso de que la opción escogida sea la de enviar la instalación al servidor se ejecuta el método *execute* de la clase Connection, anidada dentro de la actividad, del modo que ya se ha explicado en la pantalla de cualquiera de los asistentes.

Envío de petición de registro: Como es necesario mostrar una ventana de progreso mientras se procesa la comunicación con el servidor, se tratará el código del mismo modo que ya se realizó en cualquier caso anterior mediante un hilo secundario. Lo métodos *onPreExecute* y *onPostExecute* se emplean para lanzar y cerrar la ventana de progreso típica.

Al tratarse de código empleado en actividades ya explicadas, se ha considerado que no resulta especialmente interesante seguir haciendo énfasis en este aspecto.

### *3.3.4 Back-end: Servicios Web REST*

La integración de la aplicación Android desarrollada con el módulo ERP, ambos ya explicados, se realiza mediante la actuación de peticiones a una serie de servicios REST. Durante este apartado se detallará cual es la lógica establecida y la funcionalidad de cada uno de estos servicios, previamente se realiza una breve introducción a este tipo de comunicación.

**REST: Representational State Transfer**

La Transferencia de Estado Representacional o REST es una técnica de arquitectura software para sistemas hipermedia distribuidos como la Web. El término se originó en el año 2000, en una tesis doctoral sobre la web escrita por Roy Fielding [45], uno de los principales autores de la especificación del protocolo HTTP y ha pasado a ser ampliamente utilizado por la comunidad de desarrollo.

Si bien el término REST se refería originalmente a un conjunto de principios de arquitectura, en la actualidad se usa en el sentido más amplio para describir cualquier interfaz web simple que utiliza XML y HTTP, sin las abstracciones adicionales de los protocolos basados en patrones de intercambio de mensajes como el protocolo de servicios web SOAP. Los sistemas que siguen los principios REST se llaman con frecuencia RESTful.

REST afirma que la web ha disfrutado de escalabilidad como resultado de una serie de diseños fundamentales clave:

- Un protocolo cliente/servidor sin estado: cada mensaje HTTP contiene toda la información necesaria para comprender la petición. Como resultado, ni el cliente ni el servidor necesitan recordar ningún estado de las comunicaciones entre mensajes. Sin embargo, en la práctica, muchas aplicaciones basadas en HTTP utilizan cookies y otros mecanismos para mantener el estado de la sesión (algunas de estas prácticas, como la reescritura de URLs, no son permitidas por REST)

- Un conjunto de operaciones bien definidas que se aplican a todos los recursos de información: HTTP en sí define un conjunto pequeño de operaciones, las más importantes son POST, GET, PUT y DELETE.

- Una sintaxis universal para identificar los recursos. En un sistema REST, cada recurso es direccionable únicamente a través de su URI.

Para este proyecto se ha establecido que la comunicación va a estar ocultada tras un Servlet [46] escrito en Java por temas de seguridad, de forma que no va a ser posible comunicarse con los servicios directamente sino que estos serán invocados a través del

Servlet. El sistema estará compuesto por un Servlet denominado “DispatcherInstaladores” y dos servicios REST adicionales encargados de funcionalidades independientes, estos servicios son llamados “instaladorAccion” y “loginInstaladores” que serán encargados de la comunicación con la base de datos del módulo Tryton. La lógica del sistema se muestra a continuación en la Fig 3.62..

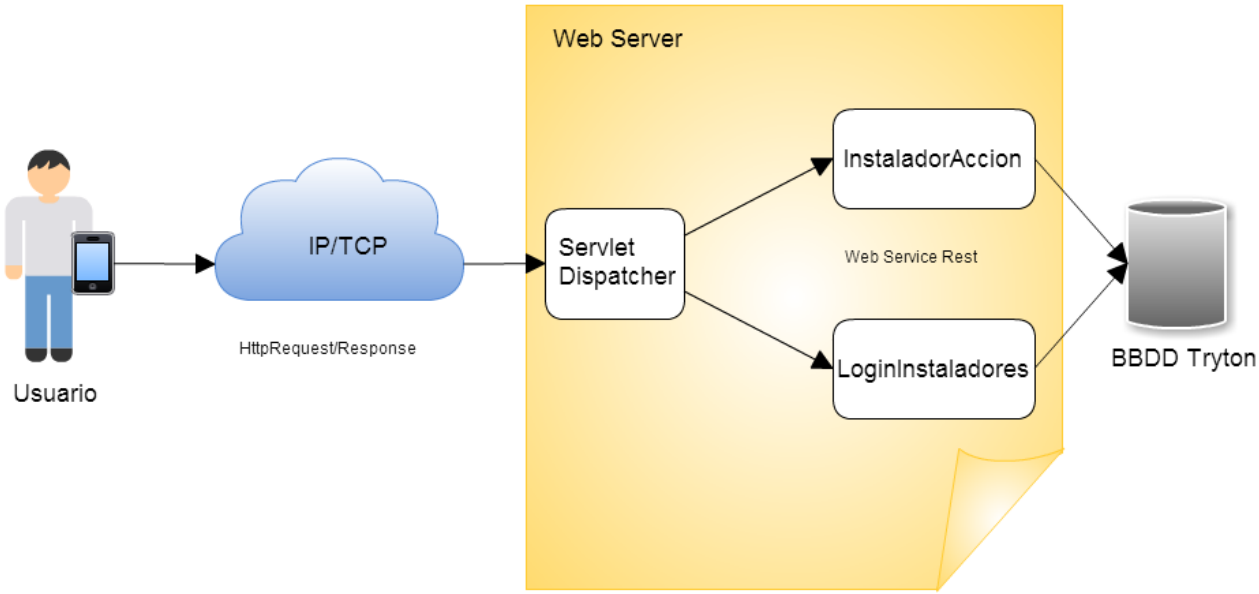



*Figura 3.62. Esquema de peticiones a Servicios REST*

Primeramente se describe el comportamiento del Servlet DispatcherInstaladores y posteriormente de cada uno de los servicios.

**DispatcherInstaladores**

Esta clase Java hereda de la clase HttpServlet de la librería *javax.servlet.http* [47] y dispone de los siguientes métodos de entrada para las diferentes peticiones realizadas por la aplicación Android:

- *public void doGet (HttpServletRequest request, HttpServletResponse response):* método invocado durante el proceso de autenticación de usuario de la aplicación Android y para la obtención del historial de instalaciones registradas en la base de datos.

  Su funcionamiento es el siguiente:

  - recupera el parámetro *params* incorporado en la petición HTTP con la siguiente línea de código,

```
request.getParameter("param");
```

  - construye la URL para realizar la petición al servicio REST correspondiente con el siguiente patrón,

```
ip:puerto/servicioREST?param1=value1&param2=value2…
```

  - establece la comunicación con el servicio gracias a la url generada mediante la siguiente porción de código,

```
serverAddress = new URL(url);
connection = (HttpURLConnection) serverAddress.openConnection();
connection.setRequestMethod("GET");
connection.setDoOutput(true);
connection.setReadTimeout(10000);
connection.connect();
```

- posteriormente se recibe la respuesta del servicio en formato JSON y esta se adjunta en la respuesta hacia la aplicación Android.

```
rd=new BufferedReader(new InputStreamReader(connection.getInputStream()));
out = response.getWriter();
sb = new StringBuilder();
while ((line = rd.readLine()) != null) {
    sb.append(line + '\n');
}
out.print(sb);
out.flush();
```

- como última acción desconecta todas las conexiones establecidas durante la comunicación.

- *public void doPost (HttpServletRequest request, HttpServletResponse response):* método invocado durante el proceso de alta de nuevos registros. Su funcionamiento es similar al del método *doGet* salvo porque en este caso se analiza la petición para localizar en su interior la imagen que la aplicación adjunta para realizar el registro. El análisis de la petición se realiza del siguiente modo:

  - Se comprueba que el objeto HTTRequest sea "multipart" indicando que contiene una imagen en su interior.

```
boolean isMultipart = ServletFileUpload.isMultipartContent(request);
```

- Se obtiene el nombre del fichero, se modifica este nombre por uno creado aleatoriamente y se guarda la imagen en el servidor, más exactamente en el interior de una carpeta que contendrá todas las imágenes de los usuarios.

- Para este método es importante destacar que la invocación del servicio REST correspondiente se realiza a través de su método POST del siguiente modo,

```
connection.setRequestMethod("POST");
```

**LoginInstaladores**

En los servicios REST se emplea el paquete *javax.ws.rs* [48] para crear las interfaces de alto nivel. Esto quiere decir que se indica dentro de la clase Java:

- el nombre lógico del servicio con el que se llama a los recursos,

```
@Path("/LoginInstaladores")
```

- el tipo de datos que devuelve el servicio, siendo un objeto de la clase LoginInstaladores transformado a formato JSON,

```
@Produces("application/json")
```

- y el método que debe ser invocado en cada caso según el mensaje HTTP,

```
@GET ó @POST
```

En el interior de la clase se implementa el método con la siguiente definición, de forma que se capturan los parámetros necesarios para realizar la consulta a base de datos adecuada:

```
   public OLoginInstaladoresGet LoginInstaladoresGet(
   @QueryParam("usuario")   String   usuario,   @QueryParam("password")   String
password){…
```

La funcionalidad de este método se resume en el diagrama de flujo de la Fig 3.63., Siendo el objetivo a cumplir que la aplicación Android sea capaz de validar el usuario y el password introducidos por el usuario en la pantalla de autenticación.

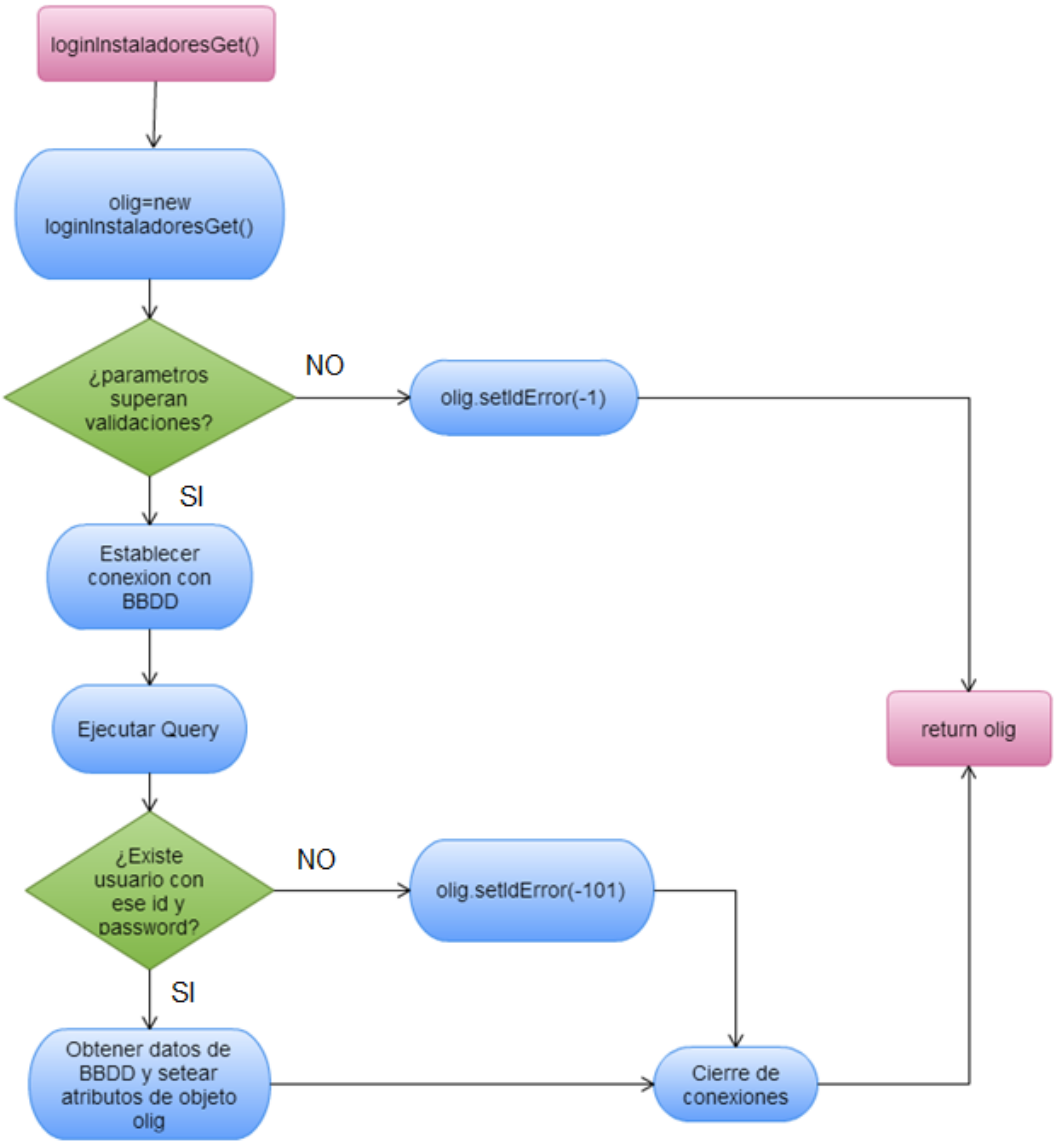



*Figura 3.63. Diagrama de flujo de proceso de autenticación*

Como aspecto importante a ser destacado del diagrama de flujo anterior es la presencia de un identificador de error que la aplicación Android analiza para dar información al usuario sobre la razón del fallo a la hora de acceder al asistente.

**InstaladoresAccion**

En el interior de la clase se implementan los métodos *instaladorAccionGet* y *instaladorAccionPost*, de forma que en cada método se capturan los parámetros necesarios para realizar la consulta o la modificación en la base de datos, al igual que se realizó para el servicio anteriormente explicado.

Primeramente se explica el funcionamiento del método GET del servicio y se adjunta la definición del método y la recepción de sus parámetros:

```
   public OinstaladorAccionGet instaladorAccionGet( @QueryParam("idUsuario") int
idUsuario){…
```

La funcionalidad del método GET se resume en el diagrama de flujo de la Fig 3.64., Siendo el objetivo a cumplir que la aplicación Android sea capaz de recuperar todos los registros que ha realizado a lo largo de todo su historial en el que caso de que existan.

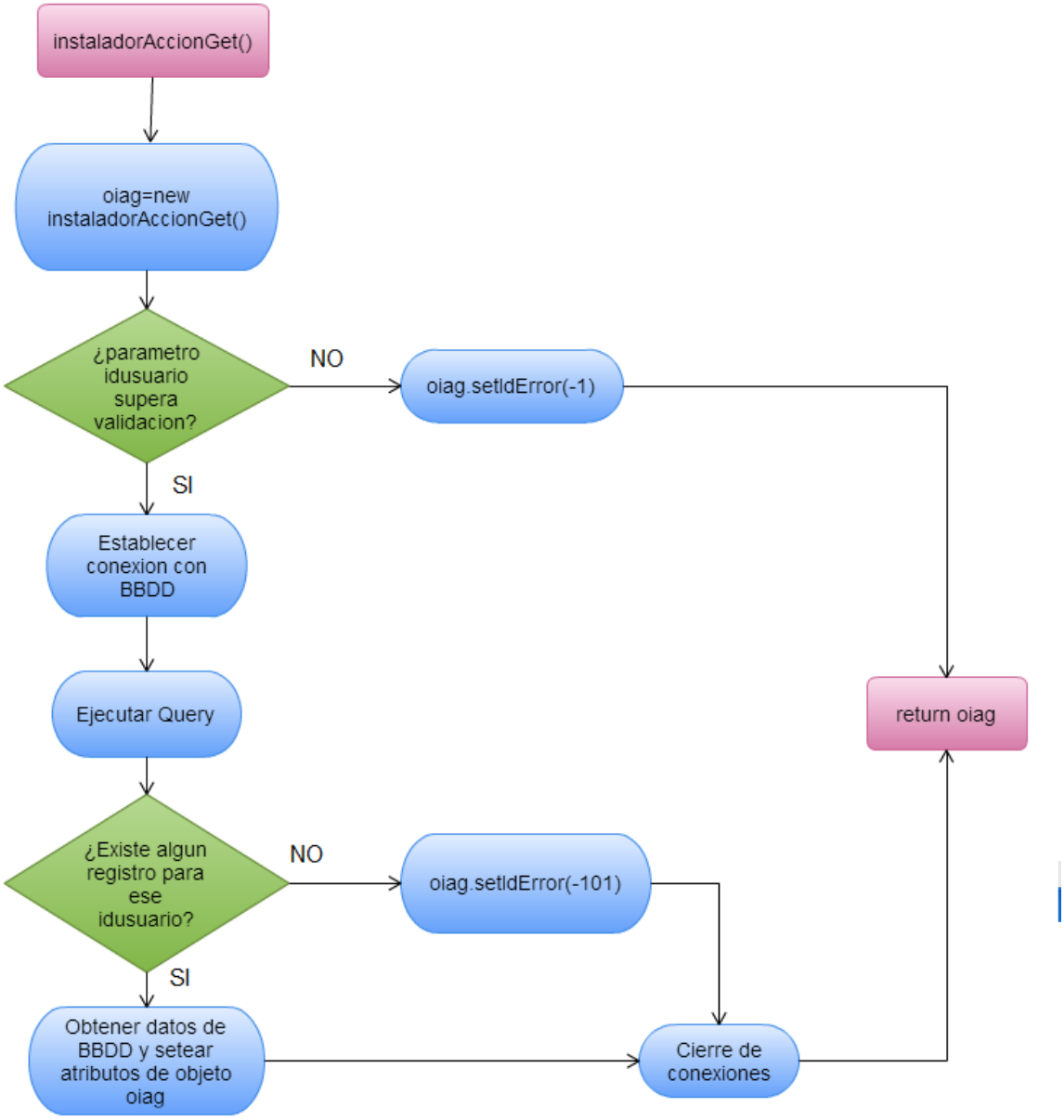



*Figura 3.64. Diagrama de flujo de una petición de consulta de histórico*

A continuación se explica el funcionamiento del método PUT del servicio incluyendo la cabecera del método:

```
public OInstaladorAccionPutPost instaladorAccionPut(
    @QueryParam("idUsuario") int idUsuario,
```

```
        @QueryParam("descripcion") String descripcion,
        @QueryParam("idBd") int idBD,
        @QueryParam("idBt") int idBT,
        @QueryParam("promocion") String promocion,
        @QueryParam("url") String url,
        @QueryParam("bluetooth") String bluetooth,
        @QueryParam("bluetooth4") String bluetooth4){…
```

La funcionalidad del método PUT se resume en el diagrama de flujo de la 3.65., Siendo el objetivo a cumplir que la aplicación Android sea capaz de registrar una instalación que el usuario acaba de realizar siempre que tal pedido se encuentre dado de alta en el sistema y la posterior notificación al cliente sobre la finalización de dicha tarea.

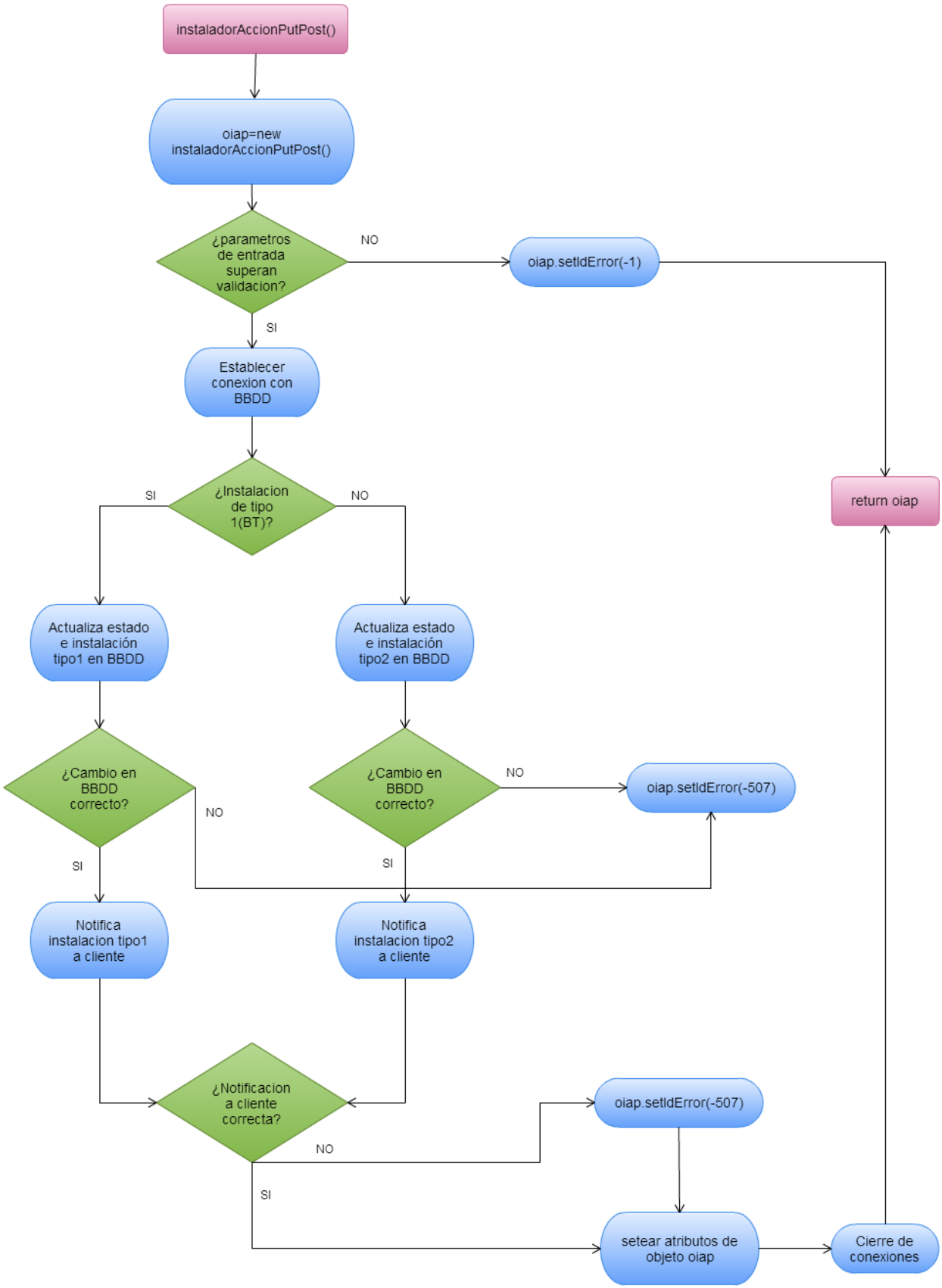



*Figura 3.65. Diagrama de flujo de una petición de registro de instalación*

Como se puede observar en el diagrama, existe un punto en el que el servicio notifica al cliente que ha solicitado la tarea de instalación, esta notificación se realiza invocando, del mismo modo que se ha realizado internamente, el método POST de un servicio REST alojado en los servidores del cliente. El sistema completo debe permitir almacenar registros a pesar de que no haya sido notificado el cliente, ya que posteriormente desde la empresa se analizan los registros que hayan sufrido de este tipo de errores para informar al cliente sobre tal incidencia.

# 4. Sistema completo

Para este capítulo ya se dispone de toda la información sobre cada una de las secciones del sistema. Haciendo referencia a estas secciones, en el Capítulo 2 se explicó el diseño y la implementación del módulo ERP para que la empresa fuese capaz de gestionar de un modo eficiente todas las tareas que solicita el cliente, es decir, la creación del módulo junto a todas las vistas, todos los modelos y la asociación entre estos elementos así como al cron desarrollado en Python encargado de la inserción automática de pedidos; en el Capítulo 3 se detalló el proceso de creación de la aplicación móvil en Android, hablando tanto de la fase de diseño como de implementación de la propia aplicación para proveedores y de la lógica de servicios REST encargada de la interacción con la base de datos del módulo Tryton.

A lo largo de esta sección se trata de unificar estos dos capítulos y formar una idea global del funcionamiento completo del sistema y la forma en la que ha sido implementado en su totalidad. Primeramente se llevará a cabo una breve descripción de la arquitectura empleada en el entorno de desarrollo con el que se ha trabajado, posteriormente se llevará a cabo una esquematización de todo es sistema y al final del capítulo se describirá el funcionamiento de todo el sistema basando esta explicación en un análisis completo de la gestión de una instalación de un dispositivo.

## 4.1 Arquitectura

Este apartado pretende clarificar los bloques de los que se compone el sistema completo. Se enumerarán los diferentes elementos en los que se divide el sistema y se explicará la manera en la que se trabaja con ellos:

El entorno de desarrollo Tryton ha sido alojado en un ordenador de la empresa reservado para esta función. El entorno Tryton abarca desde el alojamiento del servidor y la base de datos PostgreSQL hasta la evidente necesidad de un cliente GTK[31] para interactuar con el entorno. En principio se debería haber alojado tanto el servidor Tryton como la base de datos en un servidor para proporcionar un mayor rendimiento y disponer de cierta seguridad, pero dadas las circunstancias no se requirió tales medidas para realizar el desarrollo y de esta manera la implementación resultó bastante más accesible y cómoda.

El Servlet que controla la invocación de los diferentes servicios Web REST por parte de la aplicación Android así como de estos servicios se encuentran alojados en un servidor Apache, habilitado para esta cuestión, con visibilidad desde el exterior de la red de la empresa. La razón de esta visibilidad es debido a que la aplicación debe ser capaz de conectar con los servicios y de igual modo el sistema del cliente también debe tener visibilidad desde este servidor Apache ya que los servicios requieren de esta conectividad para poder notificar de la realización de las tareas encargadas.

Adicionalmente la base de datos debe estar accesible para los servicios ya que el acceso a la información que alojan sus tablas resulta esencial para el correcto funcionamiento del sistema. De igual forma el cron desarrollado para leer la cuenta de correo en la que se reciben los correos electrónicos del cliente que solicita las diferentes tareas también requiere que disponga de visibilidad de la base de datos.

La implantación de este sistema en un entorno real ha sido realizada completamente por la persona responsable dentro del departamento de sistemas, por lo tanto este aspecto no se considera que requiera de una explicación más detallada ya que queda fuera del objetivo de este proyecto.

[31] Graphical User Interface Toolkit

## 4.2 Esquema completo

En esta sección se describe y explica la Fig 4.1., la cual muestra un esquema en el que se puede observar el sistema completo implementado durante la ejecución del proyecto.

El sistema se encuentra compuesto por 3 ramas definidas claramente y que a continuación se explican brevemente:

- *Rama Cliente*: Se encuentra compuesta por el elemento capacitado para realizar el alta de pedidos, concretamente el cron implementado en Python explicado en la sección 2.4.4, Generación de pedidos; y por la notificación de la finalización del trabajo al cliente explicado en la sección 3.3.4.

  El envío de un correo electrónico a la cuenta acordada por parte del cliente supone el primer paso para la elaboración de un trabajo aunque no es obligatorio ya que se permite la creación de pedidos de forma manual desde el cliente GTK de Tryton.

  La notificación al cliente de que el trabajo ha sido finalizado constituye el último paso en la ejecución de la tarea. Se recuerda que uno de los objetivos de este proyecto consistía justamente en la reducción de los tiempos de notificación. Como ya se explicó anteriormente este punto se encuentra implementado en la lógica de los servicios REST.

- *Rama Administración*: Se encuentra compuesta por el entorno completo que proporciona Tryton, concretamente el servidor y el cliente que trabajan apoyándose en el módulo desarrollado en este proyecto, explicado en el Capítulo 2.

  Desde esta rama se capacita principalmente a Infortrex para la gestión de todos los trabajos encargados por el cliente e igualmente ofrece la base para el correcto funcionamiento de la aplicación Android. Esta rama constituye el núcleo lógico de todo el sistema implementado.

- *Rama Proveedor*: Se encuentra compuesta principalmente por los dos servicios REST y su controlador, sirviendo como base a la aplicación Android que permite el acceso a toda la red de proveedores de Infortrex, descrito en detalle en el Capítulo 3.

Como elemento común, de todas estas ramas enumeradas anteriormente, se encuentra la base de datos PostgreSQL, resultando un elemento crítico del sistema ya que si no se garantiza su disponibilidad total ninguna rama es capaz de realizar su función correctamente.

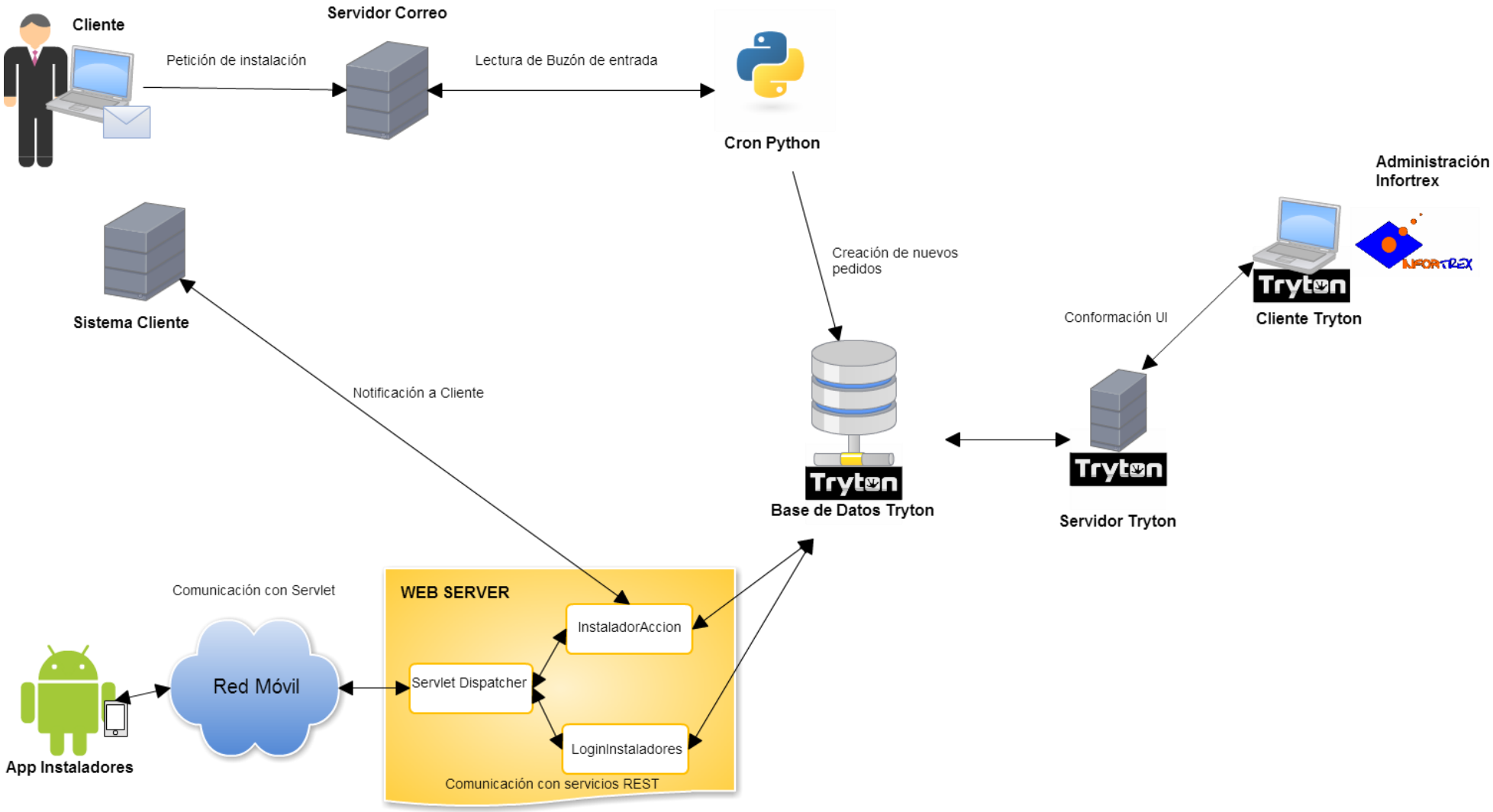



*Figura 4.1. Esquema del sistema completo*

## 4.3 Descripción del sistema completo

Una vez que ya se ha presentado la visión completa del sistema, a continuación se pretende complementar esta idea con una serie de diagramas de secuencia. De forma que se ofrece una visión completa de cómo funcionan estas ramas de una forma coordinada empleando para ello el ejemplo de la realización de una tarea modelo.

Este diagrama se dividirá a su vez en cuatro diagramas secuenciales para facilitar la compresión y la visibilidad. A continuación se explica detenidamente según un orden temporal cada diagrama conformando la secuencia de pasos que comúnmente suceden para la realización de un trabajo.

Diagrama1: Alta automática de pedido.

En la Fig 4.2. se presenta el diagrama de secuencia de lo que corresponde al primer paso en la gestión de un trabajo encargado por el cliente.

Siguiendo el diagrama de secuencia se puede observar con detenimiento todo el proceso automático llevado a cabo por el sistema desde el envío de un correo electrónico por parte del cliente a la cuenta de correo predefinida y según el formato acordado; hasta la creación del nuevo pedido en la base de datos.

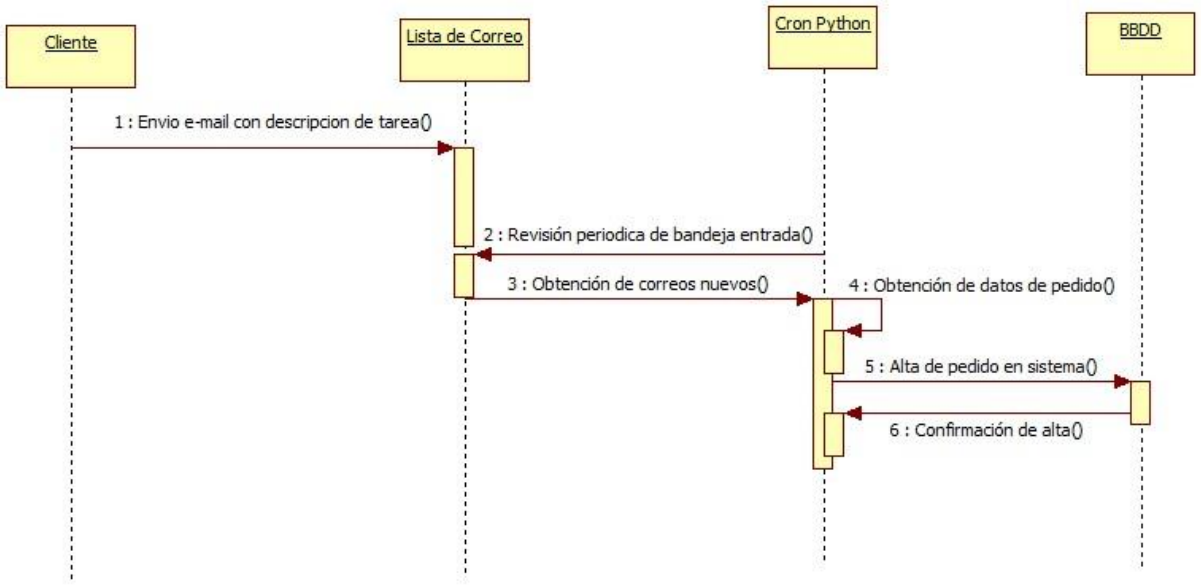



*Figura 4.2. Diagrama de secuencia de la creación de tareas a realizar*

Diagrama 2: Gestión administrativa del pedido.

En la Fig 4.3. se presenta el diagrama de secuencia de lo que corresponde a la gestión por parte de la empresa del pedido creado en el sistema anteriormente.

Observando el diagrama se observa detalladamente la comunicación entre el servidor y el cliente Tryton en función de las interacciones del usuario de la empresa. En este paso, y como anteriormente se ha comentado en el Capítulo 2, la empresa debe ponerse en contacto con el responsable del local en el que se va a llevar a cabo la instalación para acordar un horario y las condiciones de la tarea. Una vez realizadas estas gestiones el usuario asigna esta instalación a una empresa de las que se encuentren dadas de alta en el módulo Tryton para que la aplicación sea capaz de reconocer de dicha instalación la tiene pendiente de realizar.

Una vez que el usuario ya ha realizado todos los cambios necesarios a través del cliente Tryton ya se habrá completado toda la información necesaria para la elaboración de la hoja de pedido y su envió al proveedor seleccionado.

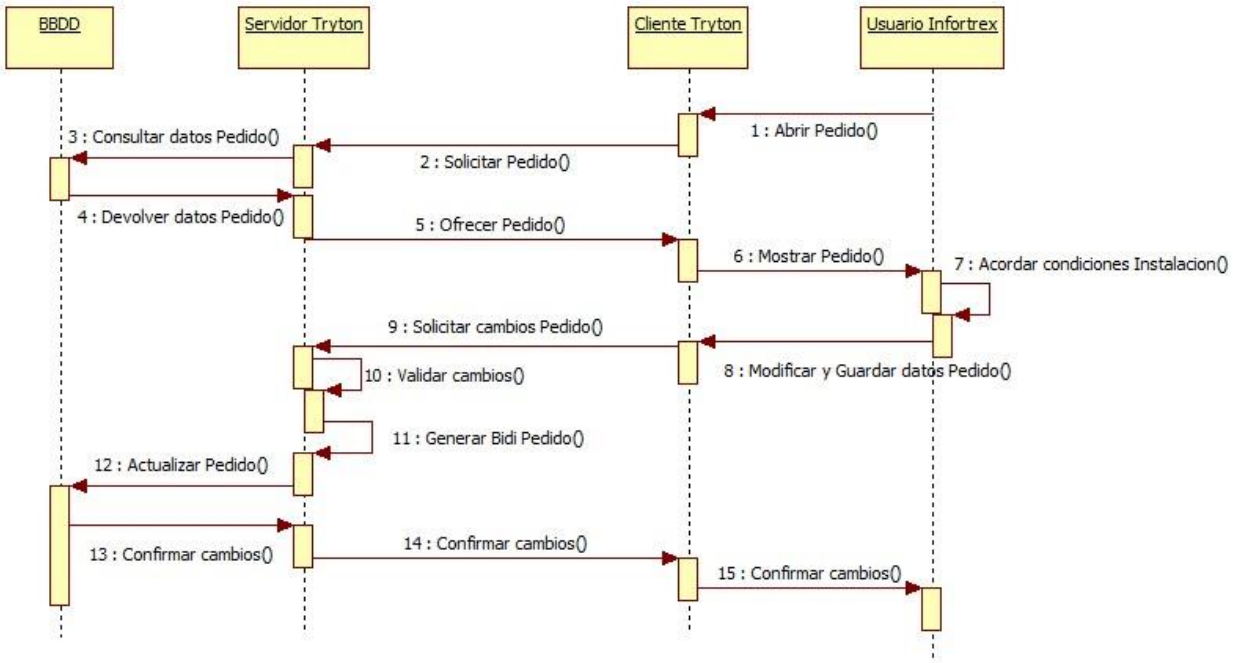



*Figura 4.3. Diagrama de secuencia de preparación de tarea de instalación*

Diagrama 3: Notificación de tarea a proveedor.

En la Fig 4.4. se muestra el diagrama de secuencia de lo que corresponde a la creación de la hoja de pedido y su posterior envío al cliente.

Es importante reflejar en estos momentos que el proceso de notificación al cliente se realiza mediante el envío de un correo electrónico adjuntando la hoja de pedido al proveedor cuya dirección figura su vista de formulario del módulo Tryton. La empresa en

no consideró necesario la automatización de este pero es posiblemente una mejora futura que desarrollar.

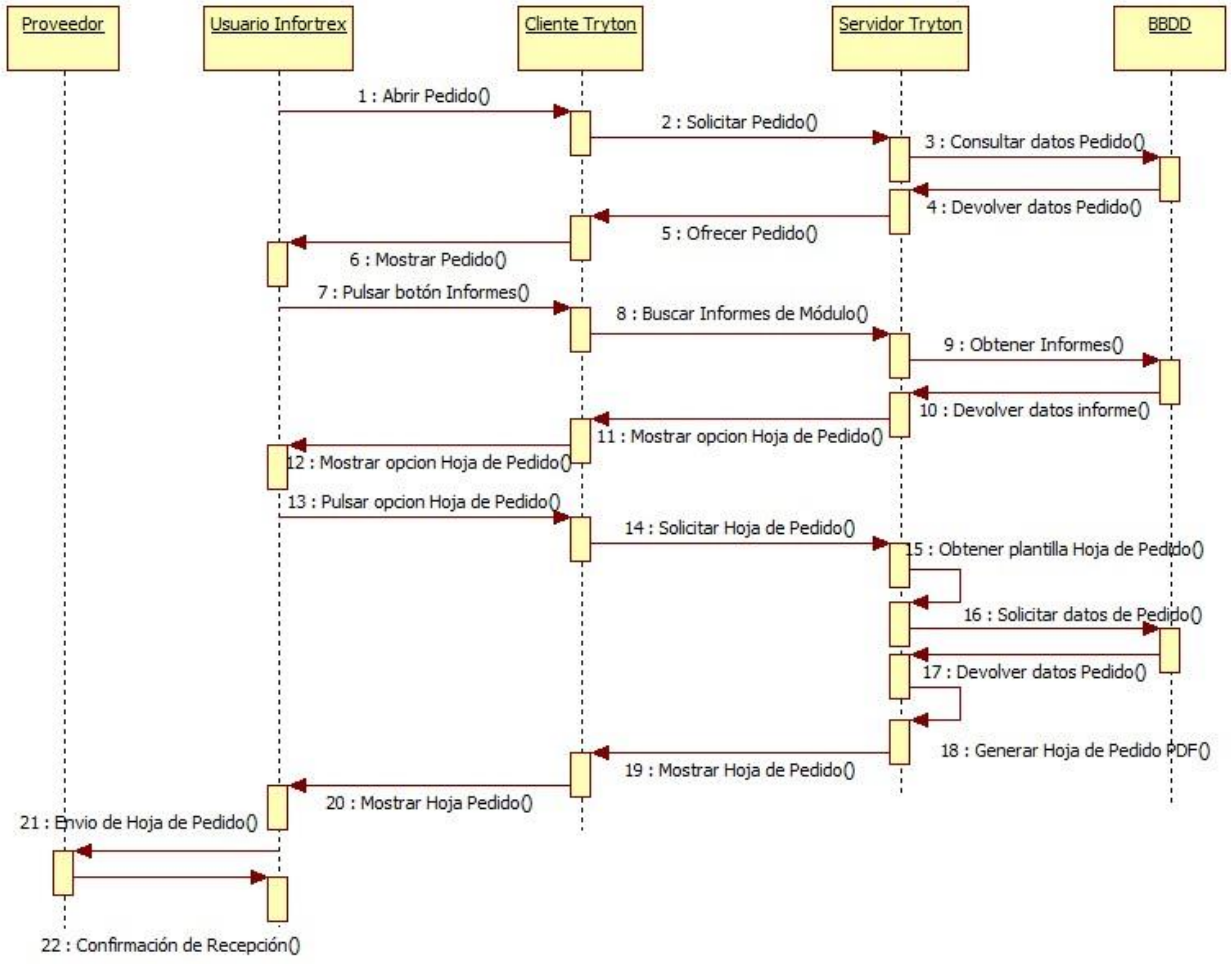



*Figura 4.4. Diagrama de secuencia de notificación de tarea a instalador*

Diagrama 4: Realización de la tarea y notificación a cliente.

A continuación, en la Fig 4.5., se muestra el diagrama de secuencia de lo que corresponde al registro del trabajo realizado por el proveedor mediante la aplicación Android.

En el diagrama se representa, en cuanto a las interacciones del usuario con la aplicación, los dos pasos obligatoriamente necesarios para realizar una instalación: la autenticación de usuario y el envío del registro; obviándose el proceso de captura de datos ya que no aporta información útil a esta secuencia.

Otro aspecto interesante en el diagrama corresponde al momento de notificación al cliente. Este paso, como se puede observar, se realiza una vez se ha registrado la información en la base de datos de forma que únicamente se informa al cliente que la instalación ha sido finalizada si a Infortrex le consta de que ha sido realizada correctamente. En el caso de que la comunicación con el cliente se encuentre interrumpida en ese momento, la aplicación dará por valido el registro pero en el módulo Tryton aparece este registro como no notificado a cliente.

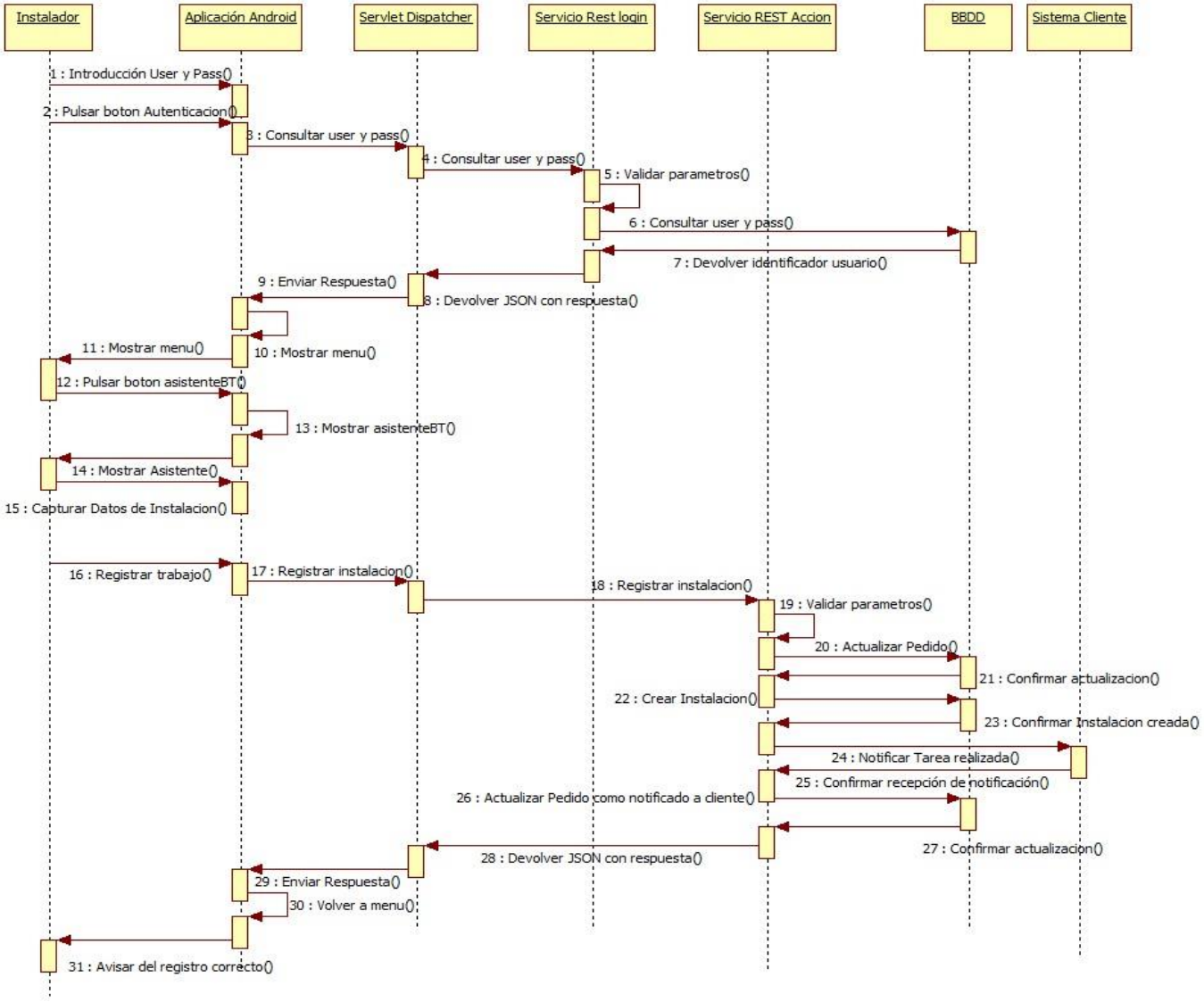



*Figura 4.5. Diagrama de secuencia de registro de instalación completada*

# 5. Conclusiones

A lo largo del documento se ha tratado de plasmar los elementos fundamentales del trabajo realizado, pero es importante conocer el alcance real de este proyecto tanto para la empresa como a nivel personal.

Este proyecto se ha basado en el diseño e implementación de un sistema que permita a la empresa Infortrex realizar de un modo eficiente y eficaz, la realización y gestión de una serie de trabajos encargados para un cliente. Las necesidades de la empresa, hacen que se haya requerido la implementación integrada de un módulo del ERP Tryton a medida y una aplicación para la plataforma Android que se proporcionará a su red de proveedores para llevar a cabo el registro de estas tareas.

Tras la finalización del proyecto, como uno de sus resultados más importantes se encuentra el haber implantado en la empresa las bases para incorporar una herramienta de gestión de recursos, de forma que en un futuro pueda verse integrado con otros procesos que se consideren oportunos.

La plataforma introducida resulta fácilmente ampliable y mantenible, por lo que la empresa dispone de todas las herramientas para introducir mejoras, que evidentemente existen dentro del sistema desarrollado, de forma que se obtengan aún mejores resultados en cuanto a productividad de los trabajadores, de forma que el trabajo que realizan sus empleados se pueda centrar en tareas más prioritarias y específicas.

Por otro lado se ha tratado, en todo momento, de facilitar el trabajo de los empleados de la empresa y de los proveedores que harán uso del sistema, ya que en todo momento se ha contado con su validación y su opinión en cuanto a usabilidad. Adicionalmente se ha elaborado una documentación que facilita el uso de la herramienta de gestión y de la aplicación, obteniéndose un feedback realmente bueno en este aspecto. En cuanto a los tiempos de respuesta hacia el cliente, la implantación de este sistema ha provocado una mejora considerable de este indicador, siendo especialmente buenos en el entorno de prueba por lo que una vez pase el sistema a producción, a corto plazo, mejorará la satisfacción del cliente.

He intentado aportar todos mis conocimientos adquiridos en el ámbito académico para realizar este proyecto aunque mis nociones sobre los temas que han intervenido en la elaboración del proyecto no eran demasiadas, por lo que la tarea de formación en el mundo de la gestión empresarial de recursos, en el desarrollo de aplicaciones Android y en programación en el lenguaje Python ha resultado fundamental y una de las tareas que más recursos ha consumido de todo el proyecto.

De forma global, el principal problema que me he encontrado a la hora de realizar el proyecto, ha sido mi corta experiencia en el desarrollo de proyectos de cierta envergadura. La organización y la planificación han formado parte de un reto personal. La planificación inicial dista bastante del tiempo real que me ha llevado la realización del proyecto, esto es debido al optimismo a la hora de pensar en cuánto tiempo me llevaría realizar cada una de las partes, sin pensar en la documentación y conocimientos que debía reunir previamente y la que tenía que redactar posteriormente.

Considero que este proyecto me ha aportado una perspectiva y un conocimiento concretos no sólo sobre la implantación de un paquete de gestión empresarial sino todo lo que conlleva el proyecto de implantación en cuento al estudio de necesidades, análisis de requisitos, diseño, estudio de los paquetes disponibles en el mercado, coste económicos y temporales.

Así mismo, considero que también he aprendido a trabajar y a entender a la figura del cliente y del proveedor, y que realizando el proyecto he conseguido alcanzar una perspectiva más real sobre el tipo de trabajo que desempeñaré en un futuro cercano.

Sin duda este proyecto ha sido muy completo y ha cubierto ampliamente los objetivos personales sobre la realización de mi proyecto fin de carrera ya que profesionalmente me ha abierto varias puertas en empresas. Durante toda la carrera no se ha llegado a tratar muchos aspectos que en este proyecto he aprendido y que tras su realización considero que podrían resultar interesantes de ofrecer a nivel académico y por supuesto para futuros proyectos universitarios.

# 6. Bibliografía y Enlaces de Interés


[1] I. S. Ojembarrena, «Infortrex,» www.infortrex.es.

[2] DirectiveSoft, «EL ERP de software libre como innovación en tiempos de crisis.,» 6 Noviembre 2009. http://prsync.es/directivesoft/el-erp-de-software-libre-como-innovacin-en-tiempos-de-crisis-2344/.

[3] «Sistema de planificación de recursos empresariales,» abril 2013. http://es.wikipedia.org/wiki/Sistema_de_planificaci%C3%B3n_de_recursos_empresariales.

[4] Kiubik, «El blog de kiubik,» Marzo 2013. http://kiubik.com/el-ecommerce-y-la-movilidad-marcaran-el-futuro/.

[5] http://es.wikipedia.org/wiki/C%C3%B3digo_abierto.

[6] http://www.bluetooth.com.

[7] http://www.android.com/.

[8] https://www.openerp.com/es.

[9] http://www.tryton.org/.

[10] http://developer.android.com/guide/components/activities.html.

[11] http://developer.android.com/guide/topics/ui/declaring-layout.html.

[12] http://es.wikipedia.org/wiki/Front-end_y_back-end.

[13] http://es.wikipedia.org/wiki/Representational_State_Transfer.

[14] A. S. Arya, «How to improve business operational efficiency?,» Octubre 2011. http://www.smeadvisor.com/2011/10/how-to-improve-business-operational-efficiency/.

[15] J. d. Brouwer, «Five reasons why an SME needs a state-of-the-art ERP solution » http://blog.delawareconsulting.be/DelawareBlog/2013/04/five-reasons-why-an-sme-needs-a-state-of-the-art-erp-solution/.

[16] J. T. Alicia Cano, «Estimación del esfuerzo de implantación en sistemas ERP,» *Revista de Procesos y*

*Métricas de las Tecnologías de la Información,* vol. 2, nº 1, pp. 3-16, 2005.

[17] «Ventajas y desventajas de usar software libre en las empresas,» 2 Mayo 2012. http://www.solucionesim.net/blog/2012/05/ventajas-y-desventajas-de-usar-software-libre-en-las-empresas/.

[18] http://www.postgresql.org.es/.

[19] http://www.mysql.com/.

[20] http://www.sqlite.org/.

[21] http://www.python.org/.

[22] «SHA - Secure Hash Algorithm,» https://www.ccn-cert.cni.es/publico/serieCCN-STIC401/es/s/sha.htm.

[23] «XML Tutorial,» http://www.w3schools.com/xml/.

[24] «Código QR,» http://es.wikipedia.org/wiki/C%C3%B3digo_QR.

[25] «Bluetooth low energy,» http://en.wikipedia.org/wiki/Bluetooth_low_energy.

[26] I. S. Asociations, «Standard Group MAC Addresses: A Tutorial Guide,» http://standards.ieee.org/develop/regauth/tut/macgrp.pdf.

[27] «Open Office,» http://www.openoffice.org/.

[28] «A templating library able to output odt and pdf files,» https://pypi.python.org/pypi/relatorio.

[29] J. Motamarri, «Compare RESTFul vs SOAP Web Services,» Agosto 2013. http://java.dzone.com/articles/j2ee-compare-restful-vs-soap.

[30] «Encodes QR-codes,» https://pypi.python.org/pypi/qrencode.

[31] «Psycopg,» http://initd.org/psycopg/.

[32] «Apple,» http://www.apple.com/es/ios/.

[33] «Open Handset Alliance,» http://www.openhandsetalliance.com/.

[34] «The Industry's Foundation for High Performance Graphics,» http://www.opengl.org/.

[35] T. Elgamal, «The Secure Sockets Layer Protocol (SSL),» *Danvers IETF Meeting,* 1995.

[36] «FreeType,» http://www.freetype.org/.

[37] http://www.linux.org/.

[38] «Pro Guard,» http://proguard.sourceforge.net/.

[39] «Introducing JSON,» http://www.json.org/.

[40] T. a. s. foundation, «HTTP Components,» http://hc.apache.org/.

[41] Z. Team, «BarCode Scanner,» Barcode Scanner.

[42] N. W. Group, «Hypertext Transfer Protocol (HTTP/1.1),» 1999.

[43] «The Busy Coder's Guide To Advanced Android Development,» https://github.com/commonsguy/cw-advandroid.

[44] «GitHub,» Available: https://github.com/.

[45] R. T. Fielding, «Architectural Styles and the Design of Network-based Software Architectures,» *UNIVERSITY OF CALIFORNIA, IRVINE,* 2000.

[46] J. I. R. A. I. Javier García de Jalón, «Aprenda Servlets de Java como si estuviera en primero,» http://www.tecnun.es/asignaturas/Informat1/AyudaInf/aprendainf/javaservlets/servlets.pdf.

[47] T. A. S. Foundation, «Servlet API Documentation,» http://tomcat.apache.org/tomcat-5.5-doc/servletapi/.

[48] «Package javax.ws.rs,» [En línea]. http://docs.oracle.com/javaee/6/api/javax/ws/rs/package-summary.html.

[49] «Encodes QR-codes,» https://pypi.python.org/pypi/qrencode.

[50] «The Industry's Foundation for High Performance Graphics,» http://www.opengl.org/.

[51] T. Elgamal, «The Secure Sockets Layer Protocol (SSL),» *Danvers IETF Meeting,* 1995.

# 7. Presupuesto y tiempo dedicado

Se han analizado dos escenarios distintos para el cálculo del presupuesto asociado a este proyecto: por un lado, la tarea realizada hasta ahora, es decir, la fase de desarrollo; por el otro, al ser ésta una aplicación ideada para ser utilizada en un entorno real, se ha considerado interesante estimar cuál sería el presupuesto necesario para su puesta en producción.

## 7.1. Escenario de desarrollo

El presupuesto asociado a este escenario se desglosa en los costes que a continuación se detallan en las siguientes secciones.

### *7.1.1. Coste de personal*

Para llevar a cabo este cálculo es necesario realizar un análisis de las distintas fases en las que se ha dividido el proyecto y consecuentemente contabilizar la duración de cada una de ellas con horas reales invertidas.

| Fase | Duración(días) | Coste diario (€) | Coste Total (€) |
|---|---|---|---|
| *Formación en entorno ERP* | 9 | 122,5 | 1.102,50 |
| *Análisis de requisitos ERP* | 1 | 122,5 | 122,50 |
| *Diseño de modelos ERP* | 1,5 | 122,5 | 183,75 |
| *Diseño de vistas ERP* | 1 | 122,5 | 122,50 |
| *Implementación de modelos ERP* | 2,8 | 122,5 | 343,00 |
| *Implementación de vistas ERP* | 2,3 | 122,5 | 281,75 |
| *Análisis de la generación automática de pedidos (cron Python)* | 0,3 | 122,5 | 36,75 |
| *Implementación de la generación automática de pedidos (cron Python)* | 2 | 122,5 | 245,00 |
| *Ejecución batería de pruebas sobre módulo ERP* | 3 | 122,5 | 367,50 |
| *Documentación del módulo ERP* | 10 | 122,5 | 1.225,00 |
| *Formación en desarrollo de aplicaciones Android* | 9 | 122,5 | 1.102,50 |
| *Análisis de requisitos aplicación Android* | 1 | 122,5 | 122,50 |
| *Diseño de layouts Android* | 1,5 | 122,5 | 183,75 |
| *Diseño de actividades Android y servicios Web* | 2,5 | 122,5 | 306,25 |
| *Implementación de layouts Android* | 2,5 | 122,5 | 306,25 |
| *Implementación de actividades Android* | 10,5 | 122,5 | 1.286,25 |
| *Implementación de Servicios Web* | 3 | 122,5 | 367,50 |
| *Documentación de la aplicación Android* | 10 | 122,5 | 1.225,00 |
| *Ejecución batería de pruebas sobre aplicación Android* | 3 | 122,5 | 367,50 |
| *Redacción de memoria y revisiones* | 44 | 122,5 | 5.390,00 |
| *Suma de tareas* | 120 | | 14.687,75 |
| *Modificador Estudiante [coste de tareas*0,3]* | | -30% | -4.406,33 |
| **TOTAL** | | | **10.281,43** |

*Figura 7.1. Desglose de coste del desarrollo proyecto*

Los datos considerados han sido 22 días laborables por cada mes y jornada de 8 horas diarias. Todos los roles necesarios para el desarrollo del proyecto (diseño, programación en distintos lenguajes, etc.) fueron asumidos por el autor. El salario considerado ha sido el establecido en las plantillas de la universidad para un Ingeniero, esto es, 2.694,39 euros por hombre y mes.

Aunque la estimación se ha hecho distinguiendo claramente las fases por simplicidad, en realidad se fueron introduciendo cambios en el diseño también en la fase de implementación al comprobar de forma práctica ciertas mejoras o límites.

Al total obtenido hay que realizarle un ajuste, teniendo en cuenta que el ingeniero realizador del proyecto no está aún en posesión del título, por lo que se le puede aplicar un factor del 70% en el cálculo. Con esto, el coste de personal asciende a 10.281,43 euros.

### 7.1.2. Coste de material

En lo que a material se refiere se han necesitado los siguientes elementos presenten en la Fig 7.2., para la obtención de cada coste se ha estimado que el periodo de amortización de los materiales son de 3 años.

| Material | Cantidad | Coste Unitario Anual (€) | Coste Total (€) |
|---|---|---|---|
| *Servidor Desarrollo* | 0,2 | 700 | 140 |
| *Cabina discos* | 0,05 | 1.500 | 75 |
| *Puesto de trabajo ERP* | 0,5 | 150 | 75 |
| *Puesto de trabajo Android* | 0,5 | 150 | 75 |
| *Terminal Android* | 0,5 | 100 | 50 |
| **Total** | | | **415** |

*Figura 7.2. Desglose de costes por material*

Otro aspecto interesante es que no han existido costes de software ya que todas las tecnologías empleadas han sido con software gratuito.

### 7.1.3. Costes indirectos

Se calculan como el 25% del coste total y engloban todos aquellos gastos como electricidad, conexión a Internet, firewall, mantenimiento de oficina, etc.

| Concepto | Coste (€) |
|---|---|
| *Coste personal* | 10.281,43 |
| *Coste material* | 415 |
| **Costes indirectos** [0,25*(costePersonal+costeMaterial)] | **2.674,11** |

*Figura 7.3. Desglose de costes indirectos*

### 7.1.4. Total

El presupuesto total para el desarrollo del sistema es de 13.370,54.

| Concepto | Coste (€) |
|---|---|
| *Coste personal* | 10.281,43 |
| *Coste material* | 415 |
| *Costes indirectos* | 2.674,11 |
| **Total** | **13.370,54** |

*Figura 7.4. Coste total*

## 7.2. Escenario de producción

A continuación se muestra la estimación del coste de la puesta en marcha del sistema en un entorno de producción durante un periodo de tiempo de un año. El presupuesto asociado a este escenario se desglosa en las siguientes secciones.

### 7.2.1. Coste de personal

En referencia al personal necesario para llevar a cabo la implantación del sistema en un entorno de producción, el posterior mantenimiento y la administración de los servidores se ha considerado que un único ingeniero podría llevar a cabo esta tarea.

| Fase | Duración(días) | Coste diario (€) | Coste Total (€) |
|---|---|---|---|
| *Puesta en marcha Producción* | 2 | 122,5 | 245,00 |
| *Mantenimiento de ERP* | 4 | 122,5 | 490,00 |
| *Mantenimiento Android* | 4 | 122,5 | 490,00 |
| *Mantenimiento servidores* | 10,5 | 122,5 | 1.286,25 |
| **TOTAL** | 20,5 | | 2.511,25 |

*Figura 7.5. Estimación de coste puesta en producción*

Para el cálculo se considera una dedicación, de esta persona responsable, de entorno a unas 148 horas anuales, es decir, un 7% de las 2112 horas anuales. Por otra parte hay que considerar el tiempo dedicado a la puesta a punto para su funcionamiento, cuya duración viene estimada en dos jornadas laborales.

### *7.2.2. Coste de material*

Al igual que en el apartado anterior, sólo es necesario tener en cuenta los gastos de hardware.

En primer lugar es necesaria una separación física de la base de datos y del servidor de aplicaciones debido a los diferentes requerimientos computacionales entre ellos y la presencia de esta práctica preventiva dentro de la empresa. No se espera a largo plazo que el consumo de recursos por parte de este sistema vaya a significar un consumo importante por lo que en principio basta con alojar el sistema en un servidor estable. El coste de un servidor de estas características se estima que sea de aproximadamente unos 2400 euros, amortizados en un periodo de 3 años.

| Material | Cantidad | Coste Unitario Anual (€) | Coste Total (€) |
|---|---|---|---|
| *Servidor BBDD* | 0,4 | 800 | 320 |
| *Servidor Aplicaciones* | 0,4 | 800 | 320 |
| *Cabina discos* | 0,1 | 1500 | 150 |
| *Puesto de trabajo Mantenimiento ERP* | 0,015 | 120 | 1,8 |
| *Puesto de trabajo Mantenimiento Android* | 0,015 | 120 | 1,8 |
| **Total** | | | **793,6** |

*Figura 7.6. Desglose de coste de material*

### *7.2.3. Costes indirectos*

Se calculan como el 25% del coste total y engloban todos aquellos gastos como electricidad, conexión a Internet, firewall, mantenimiento de oficina, etc.

| Concepto | Coste (€) |
|---|---|
| *Coste personal* | 2.511,25 |
| *Coste material* | 793,6 |
| **Costes indirectos** [0,25*(costePersonal+costeMaterial)] | **826,21** |

*Figura 7.7. Desglose de costes indirectos*

### *7.2.4. Total*

El presupuesto total para el escenario de producción seria de 4.131,06 euros.

| Concepto | Coste (€) |
|---|---|
| *Coste personal* | 2.511,25 |
| *Coste material* | 793,6 |
| *Costes indirectos* | 826,21 |
| **Total** | **4.131,06** |

*Figura 7.8. Coste total del escenario de producción*

## 7.3. Tiempo dedicado

Por último y para finalizar la presente memoria se adjunta el diagrama de Gantt, donde se observa la distribución de tiempos de tareas y sus dependencias; y el WBS[32], donde se observa el árbol de tareas de este proyecto.

[32] Work Breakdown Structure

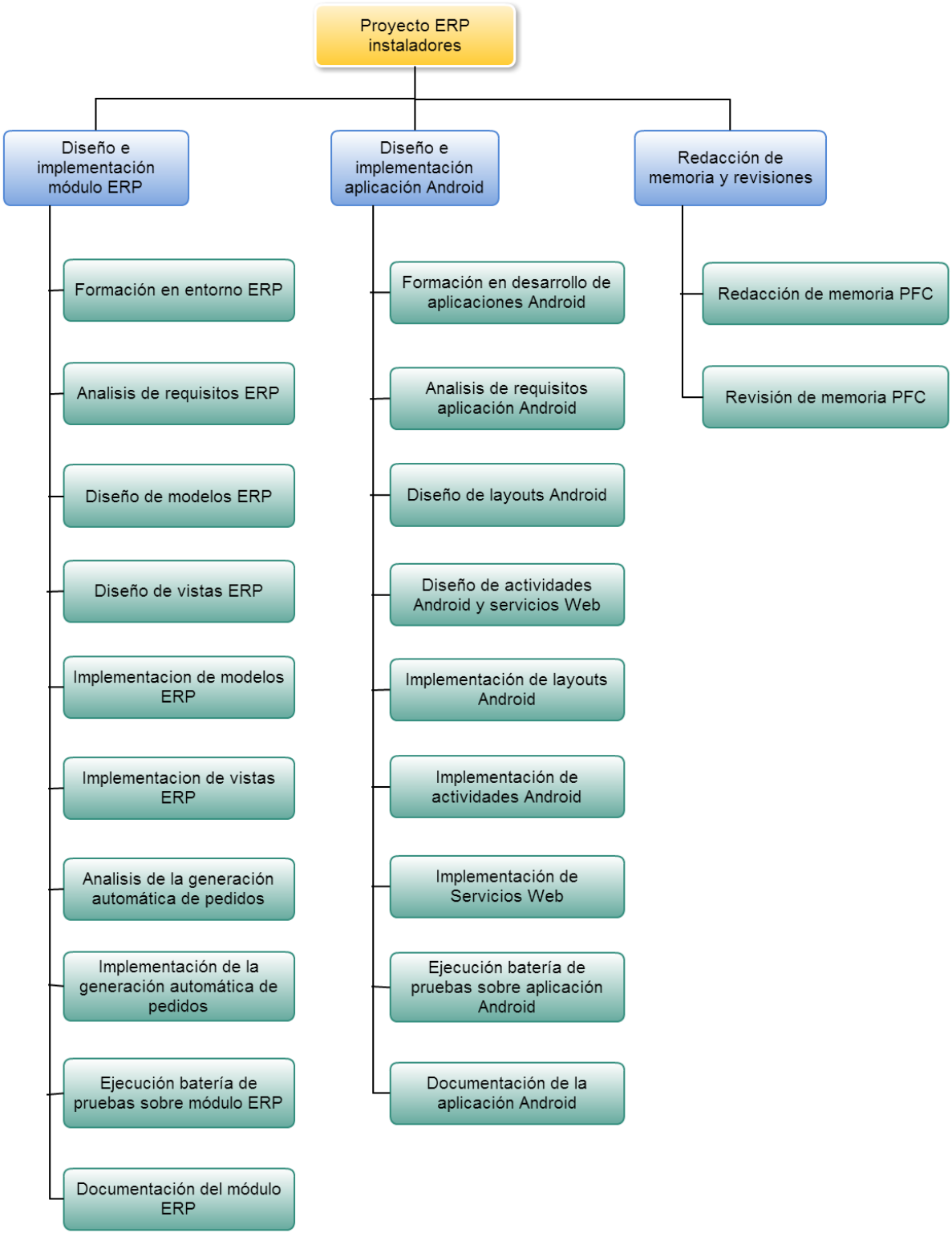
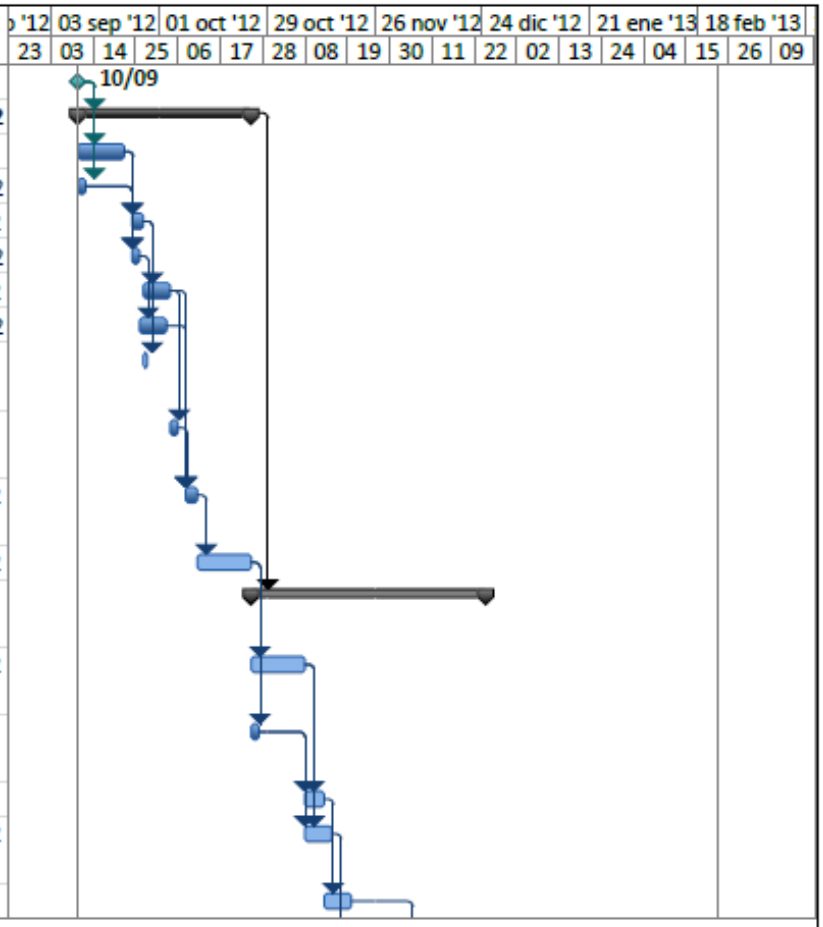



*Figura 7.9. WBS del proyecto*

| Id | Nombre de tarea | Duración | Comienzo | Fin |
|---|---|---|---|---|
| 1 | 1 Inicio Proyecto | 0 días | lun 10/09/12 | lun 10/09/12 |
| 2 | **2 Diseño e implementación módulo ERP** | **33 días** | **lun 10/09/12** | **mié 24/10/12** |
| 3 | 2.1 Formación en entorno ERP | 10 días | lun 10/09/12 | vie 21/09/12 |
| 4 | 2.2 Analisis de requisitos ERP | 2 días | lun 10/09/12 | mar 11/09/12 |
| 5 | 2.3 Diseño de modelos ERP | 3 días | lun 24/09/12 | mié 26/09/12 |
| 6 | 2.4 Diseño de vistas ERP | 2 días | lun 24/09/12 | mar 25/09/12 |
| 7 | 2.5 Implementacion de modelos ERP | 5 días | jue 27/09/12 | mié 03/10/12 |
| 8 | 2.6 Implementacion de vistas ERP | 5 días | mié 26/09/12 | mar 02/10/12 |
| 9 | 2.7 Analisis de la generación automática de pedidos (cron Python) | 1 día | jue 27/09/12 | jue 27/09/12 |
| 10 | 2.8 Implementación de la generación automática de pedidos (cron Python) | 2 días | jue 04/10/12 | vie 05/10/12 |
| 11 | 2.9 Ejecución batería de pruebas sobre módulo ERP | 3 días | lun 08/10/12 | mié 10/10/12 |
| 12 | 2.10 Documentación del módulo ERP | 10 días | jue 11/10/12 | mié 24/10/12 |
| 13 | **3 Diseño e implementación aplicación Android** | **43 días** | **jue 25/10/12** | **lun 24/12/12** |
| 14 | 3.1 Formación en desarrollo de aplicaciones Android | 10 días | jue 25/10/12 | mié 07/11/12 |
| 15 | 3.2 Analisis de requisitos aplicación Android | 2 días | jue 25/10/12 | vie 26/10/12 |
| 16 | 3.3 Diseño de layouts Android | 3 días | jue 08/11/12 | lun 12/11/12 |
| 17 | 3.4 Diseño de actividades Android y servicios Web | 5 días | jue 08/11/12 | mié 14/11/12 |
| 18 | 3.5 Implementación de layouts Android | 5 días | mar 13/11/12 | lun 19/11/12 |




Proyecto: Proyecto1
Fecha: vie 12/04/13

| | | | | | |
|---|---|---|---|---|---|
| Tarea | Hito externo | Informe de resumen manual |
| División | Tarea inactiva | Resumen manual |
| Hito | Hito inactivo | Sólo el comienzo |
| Resumen | Resumen inactivo | Sólo fin |
| Resumen del proyecto | Tarea manual | Fecha límite |
| Tareas externas | Sólo duración | Progreso |

Página 1

*Figura 7.10. Diagrama Gantt del proyecto*

# 8. Anexos

## I. Manual del módulo Tryton

Este Anexo es redactado con la finalidad de servir de ayuda para la utilización del módulo Tryton creado durante este proyecto. Se describirán los pasos que permiten trabajar con el módulo, crear, editar y borrar registros de las diferentes entidades; gestionar todas las instalaciones y la red de proveedores de un modo básico.

Al arrancar el cliente Tryton, se abre la ventana de login, Fig I.1.. En ella se deberá introducir la IP y el puerto correspondiente al servidor de Tryton. Posteriormente se debe introducir el nombre de la base de datos que se esté utilizando ya que es habitual que se disponga de varias instancias en el mismo servidor Tryton, y por último el usuario y el password de la persona que desea entrar en el sistema.

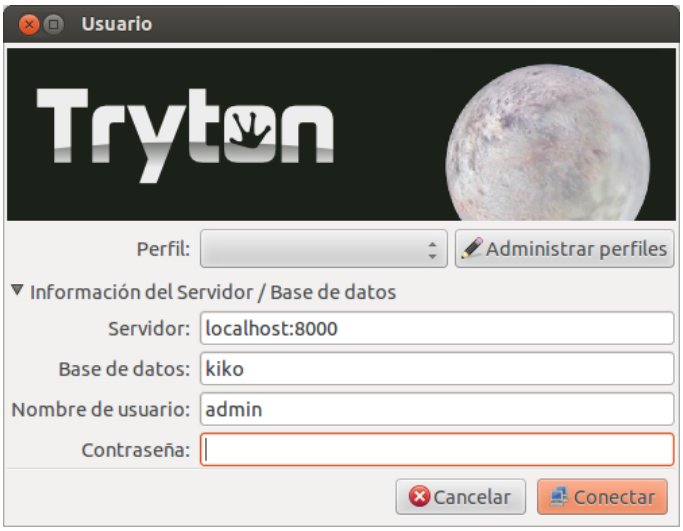



*Figura I.1. Ventana de login Tryton*

Si la autenticación es correcta, se abre la vista principal del ERP, Fig I.2., en ella podremos observar que se encuentran los módulos por defecto y el módulo que se ha creado en este proyecto, en el menú se encuentra bajo el nombre de “Gestión de Instalaciones”.

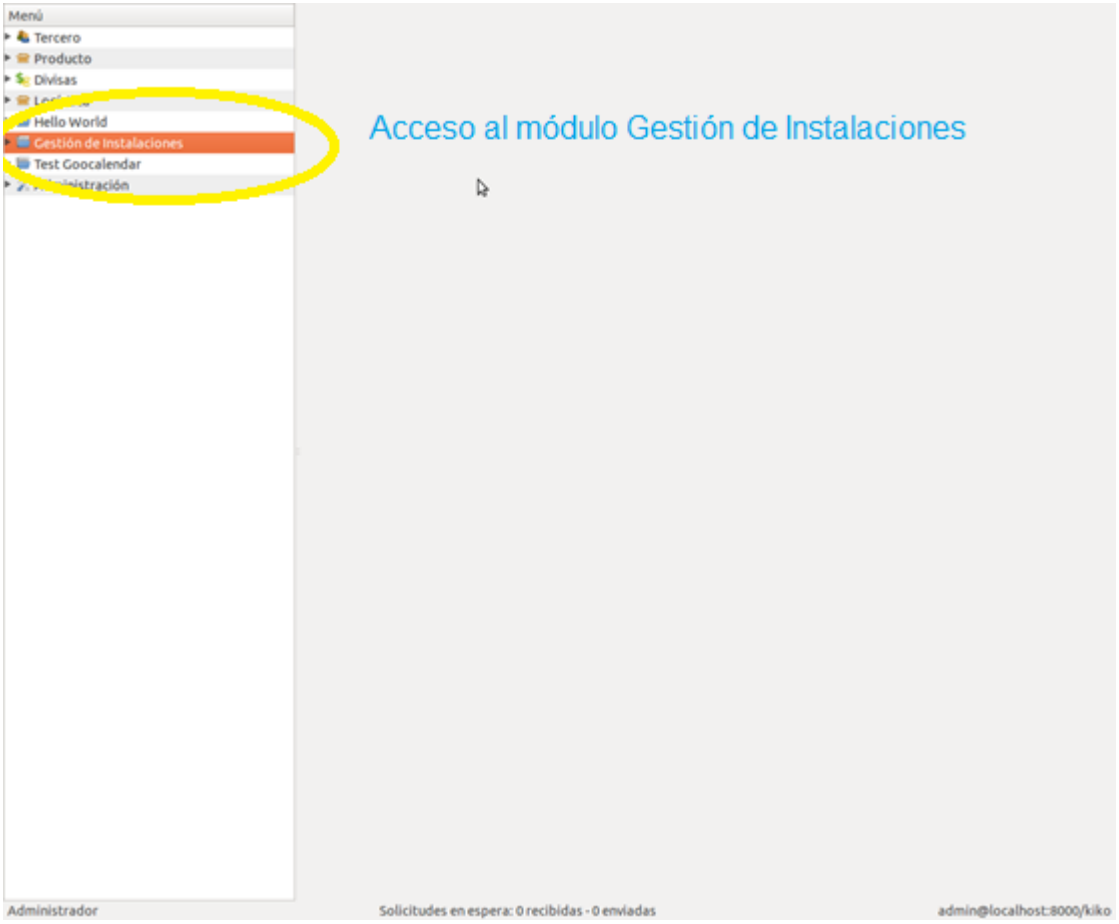



*Figura I.2. Menú de Tryton*

Dentro de “Gestión de Instaladores”, Fig I.3. se dispone de los accesos a todas las entidades que intervienen en este proyecto. Se puede ver el acceso a los instaladores, empresas, responsables, instalaciones y a los pedidos generados.


*Figura I.3. Menú desplegado de Tryton*

Pulsando sobre uno de los accesos se abre la vista de tipo listado correspondiente, Fig I.4., en la que se pueden observar todos los registros creados y desde el cual se permiten crear nuevos registros pulsando el icono de “Crear Registro”.

En este tipo de vistas se dispone de la importante funcionalidad de poder filtrar por cualquiera de los campos mostrados en la vista, CIF, razón social, correo electrónico o teléfono.

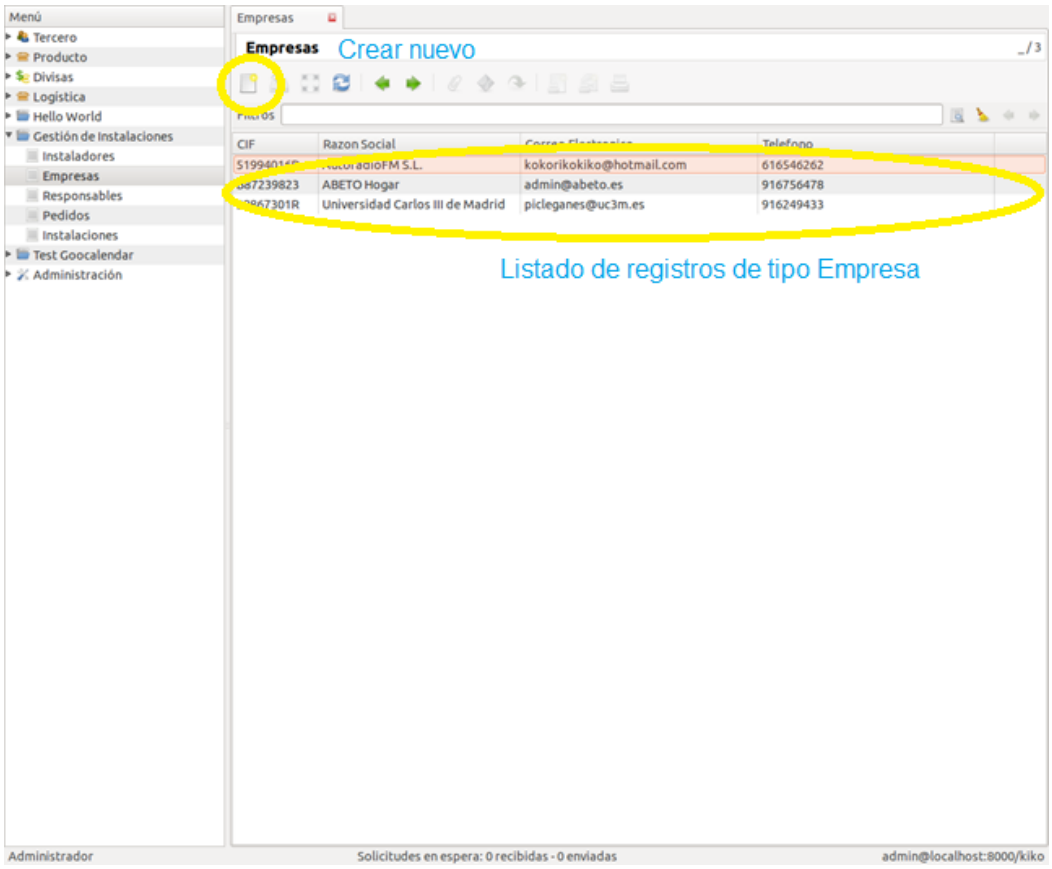



*Figura I.4. Vista listado de entidad Empresa*

Pulsando sobre el icono “Crear Registro” se nos muestra la vista de tipo formulario, Fig I.5., en la que se debe introducir todos los datos referentes a la nueva empresa a crear.

Desde esta pantalla se podrá acceder a otros registros ya creados pulsando sobre los iconos con flechas verdes.

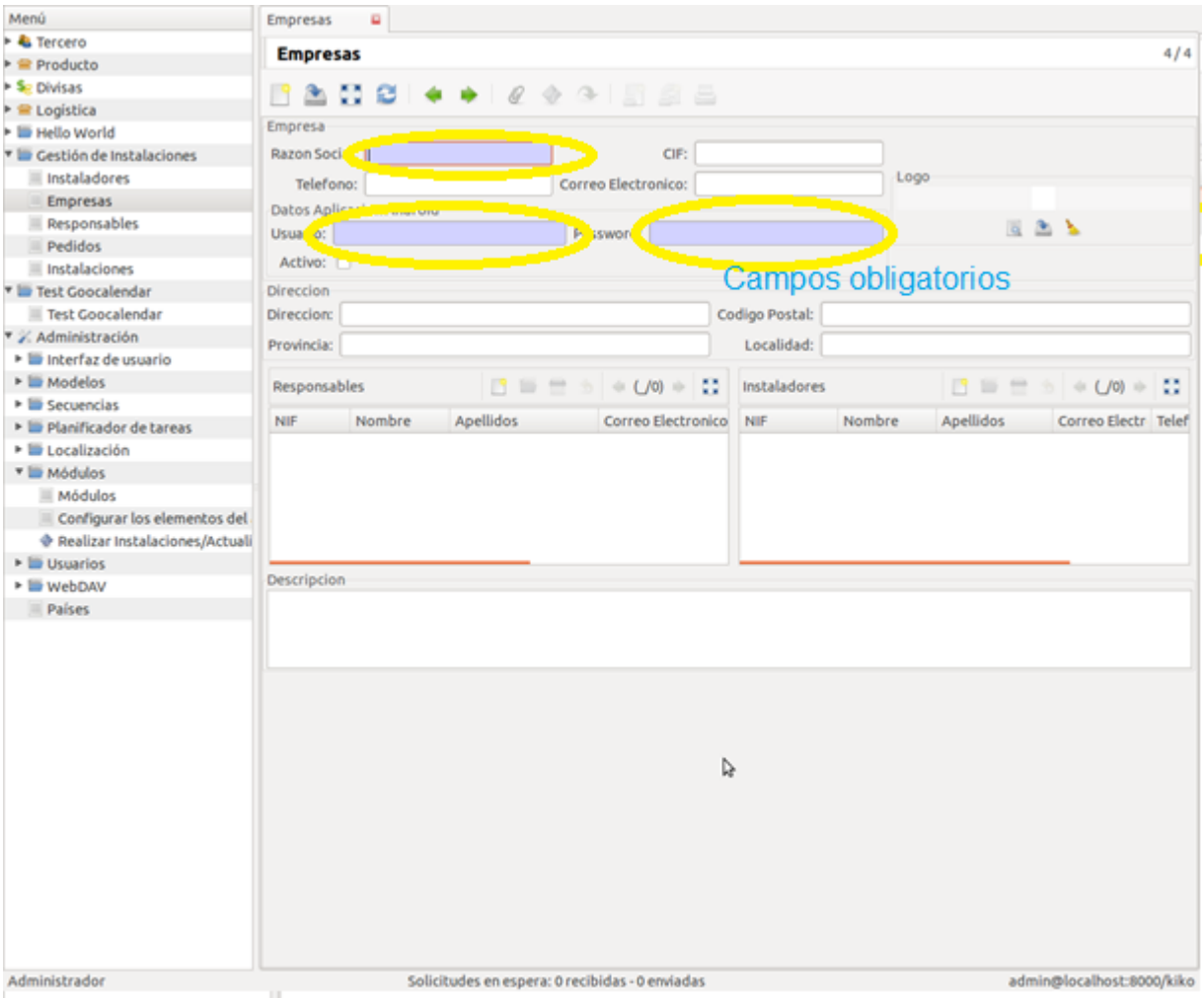



*Figura I.5. Vista formulario de entidad Empresa*

Los campos marcados en color azul significan que son campos obligatorios y que sin su correcta creación no será posible almacenar el nuevo registro. Si pulsamos el botón de “Guardar” sin rellenar los campos obligatorios se ofrece al usuario una indicación informando de la ausencia de contenido en estos campos, Fig I.6..

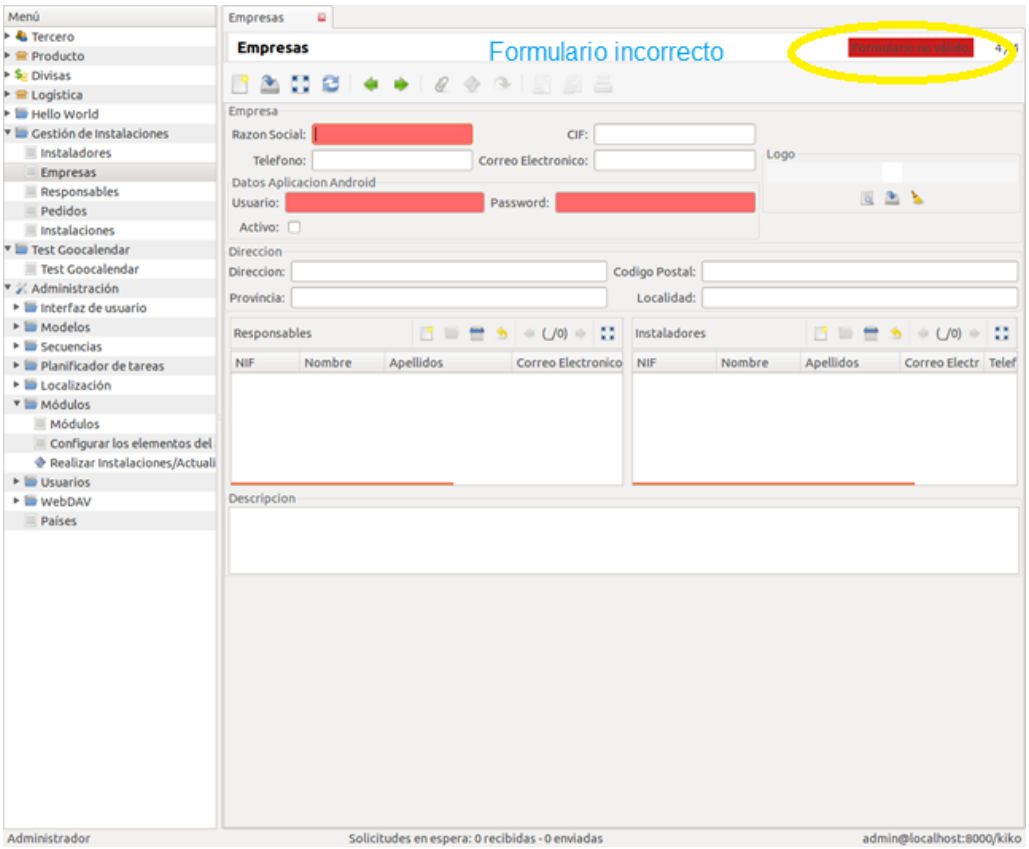



*Figura I.6. Vista formulario incorrecto de entidad Empresa*

En el caso de que la entidad disponga de alguna restricción de contenido, como es el caso del campo teléfono, cuyo primer carácter debe contener un “6” o “9” y con una longitud de nueve caracteres, y que su contenido no cumpla la restricción desarrollada, se observará una ventana como la capturada en la Fig I.7..

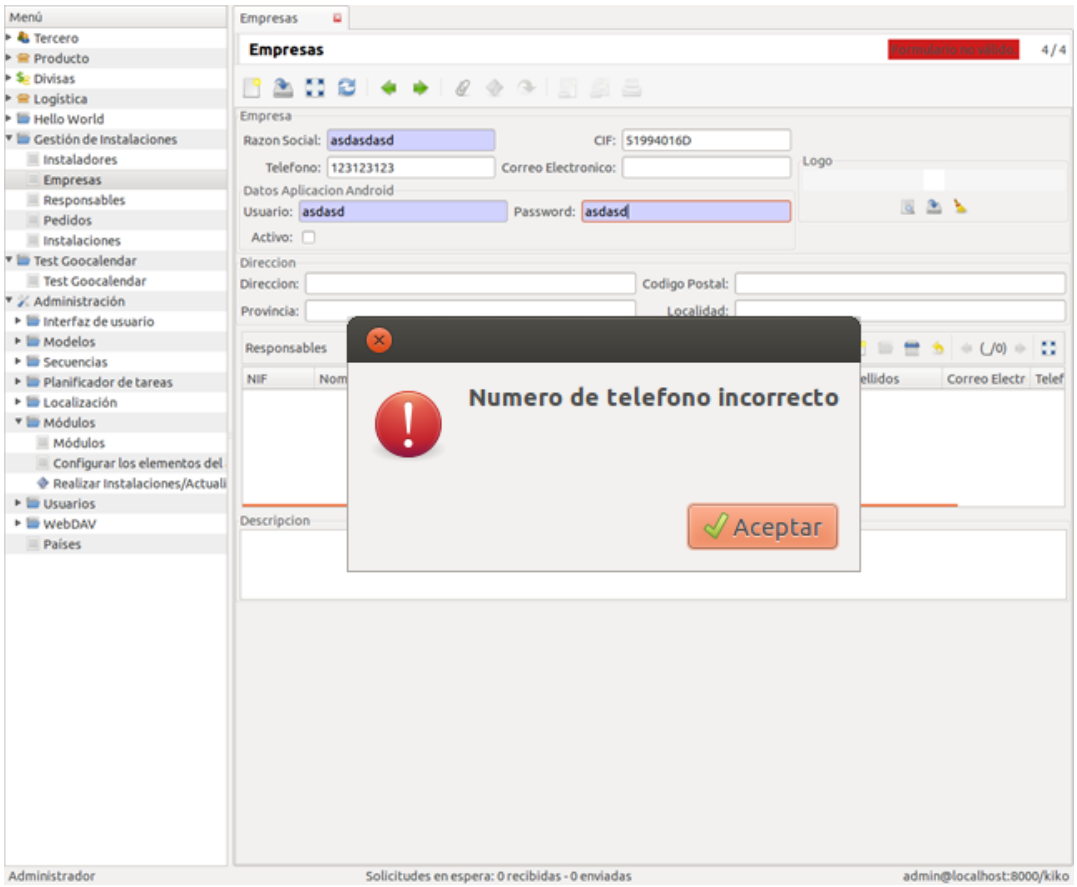



*Figura I.7. Vista formulario con teléfono incorrecto de entidad Empresa*

Desde el menú principal igualmente se permite acceder a la creación de un registro de tipo Instalador, Fig I.8..

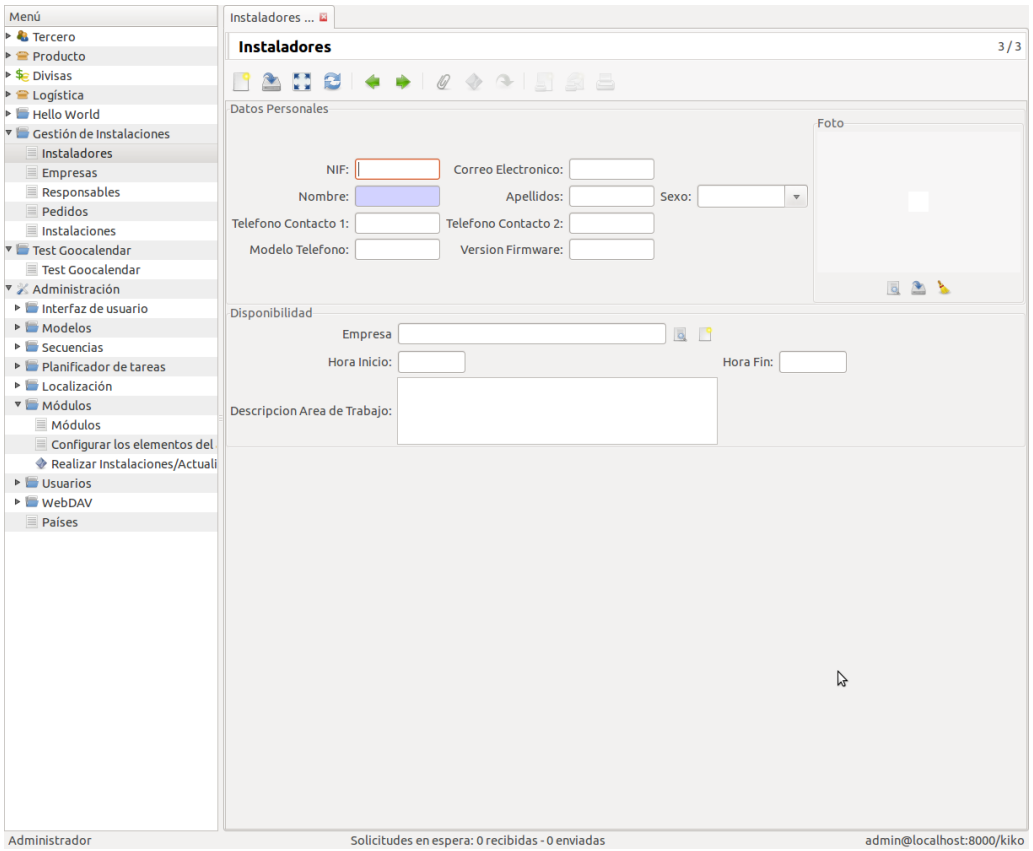



*Figura I.8. Vista formulario de entidad Instaladores*

Si introducimos los datos correspondientes dispondremos de un registro nuevo, tal y como se observa en la Fig I.9.

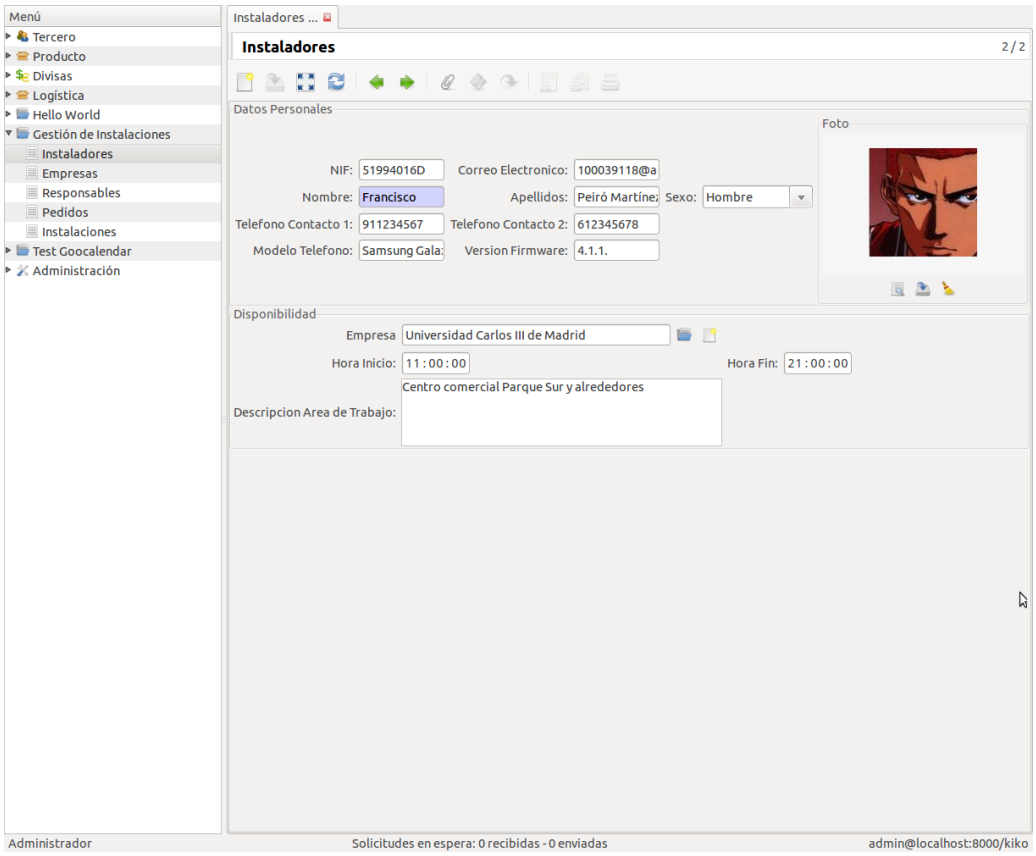



*Figura I.9. Vista formulario completado de entidad Instaladores*

Y si se pulsa sobre el botón “Cambiar de vista”, podremos alternar entre la vista de tipo listado, Fig I.10., y la de tipo formulario que ya se ha podido observar en figuras anteriores.

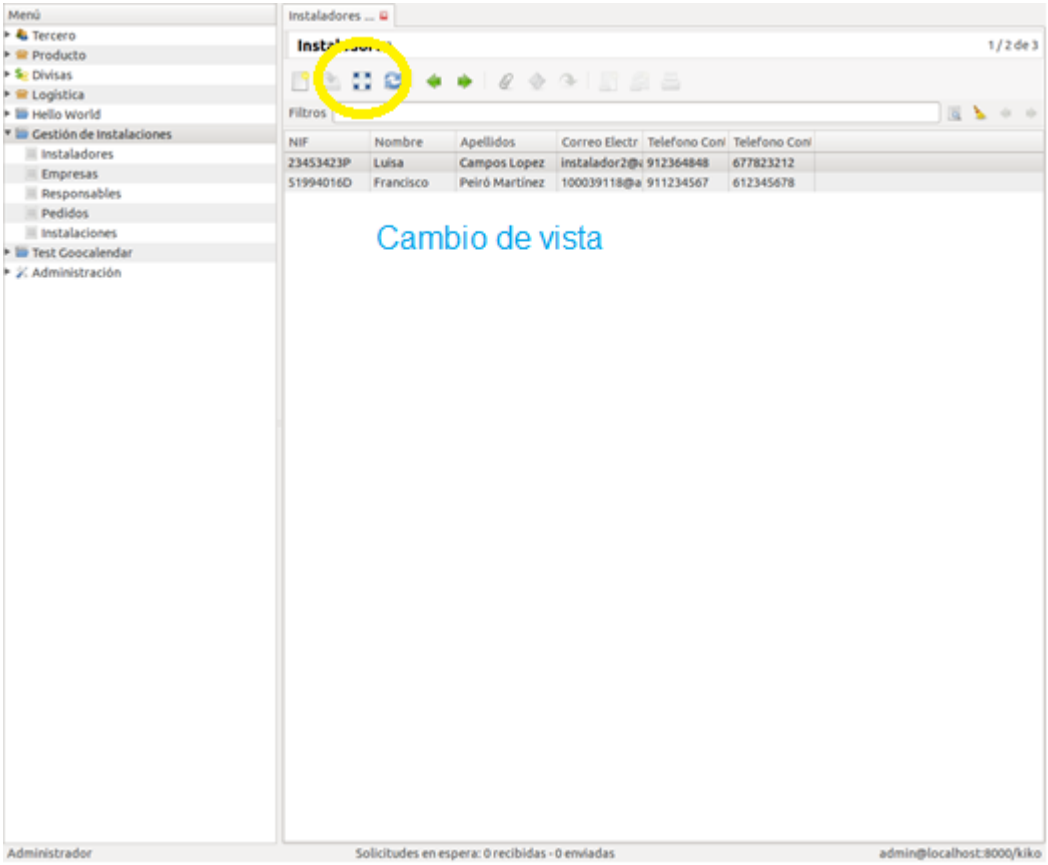



*Figura I.10. Vista listado de entidad Instaladores*

El procedimiento para la creación de registros de tipo “Responsable” es equivalente al realizado para la creación de nuevos instaladores, Fig I.11 y Fig I.12.

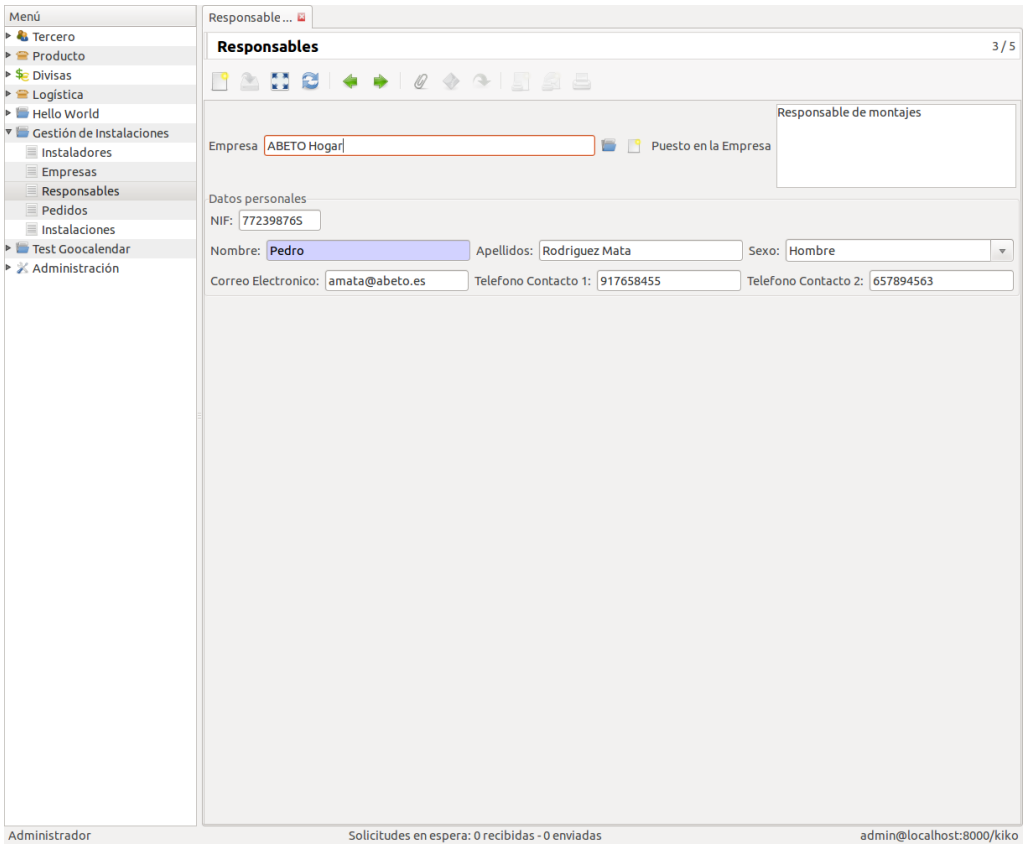



*Figura I.11. Vista formulario de entidad Responsable*

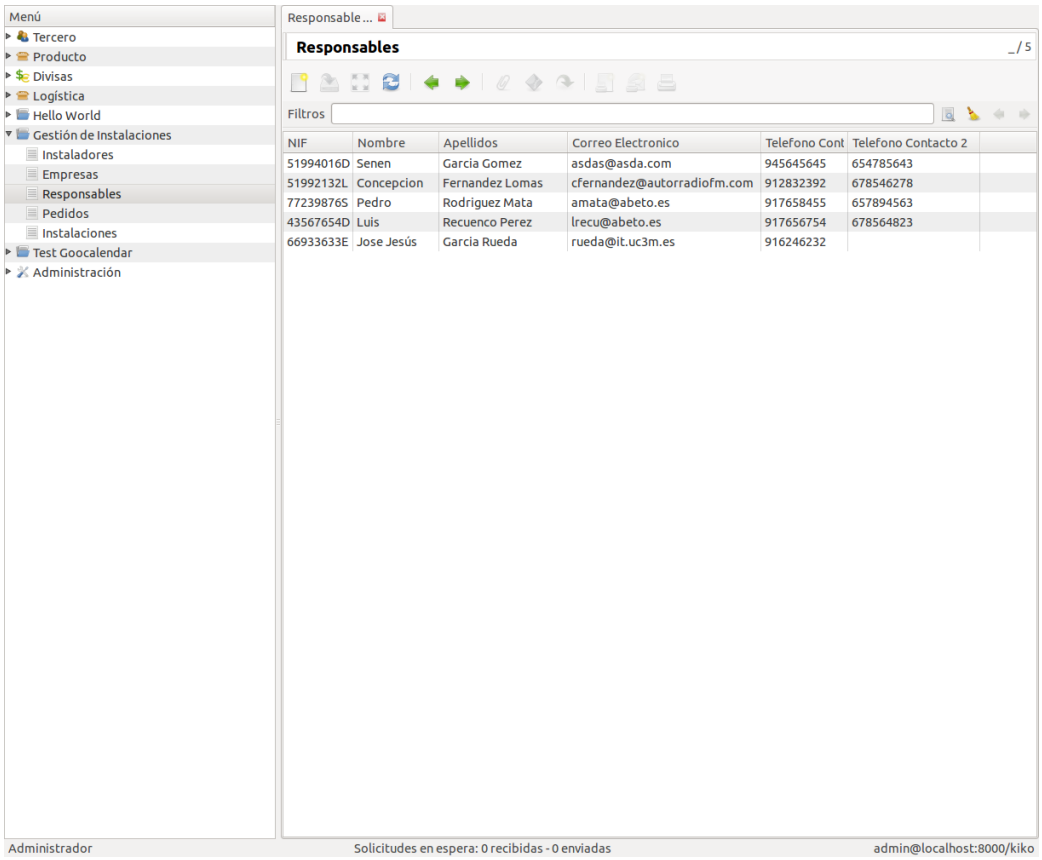



*Figura I.12. Vista listado de entidad Responsable*

Si en el campo *empresa* se ha seleccionado un registro de tipo Empresa, en la vista de tipo formulario se verá automáticamente está asociación, Fig I.13., y pulsando sobre un responsable o un instalador de los asociados se abrirá la vista de tipo formulario del propio registro, Fig I.14.

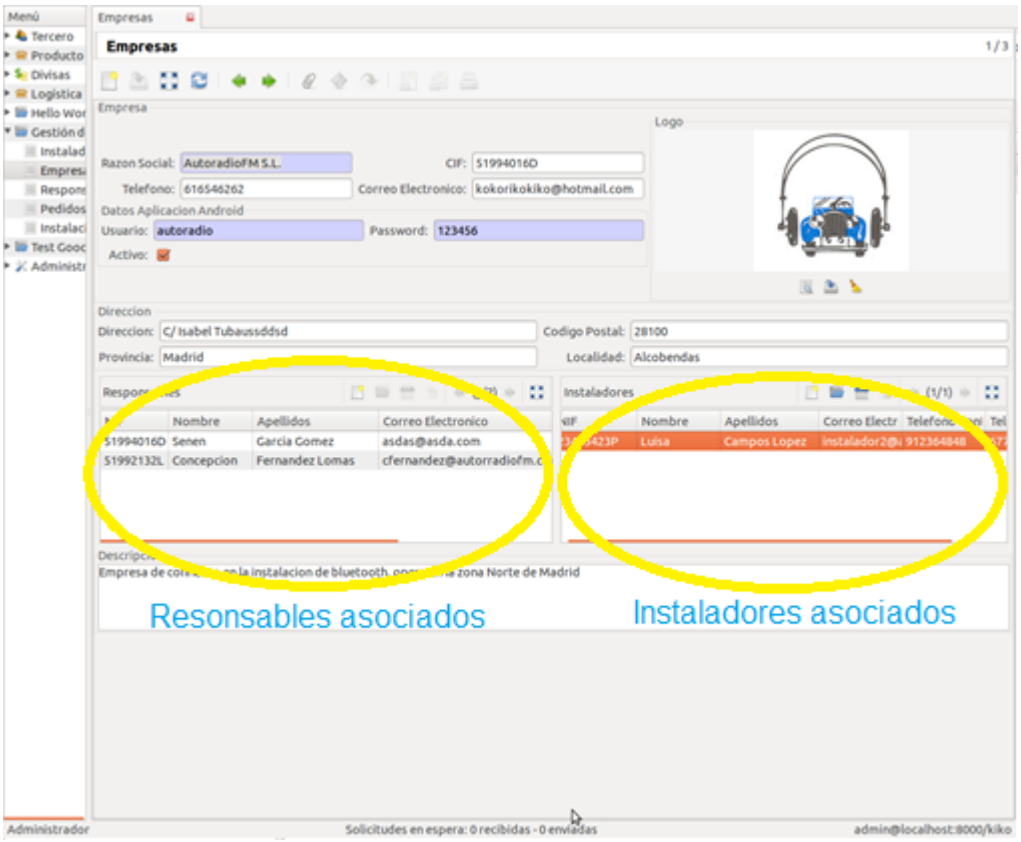



*Figura I.13. Responsables e Instaladores asociados a Empresa*

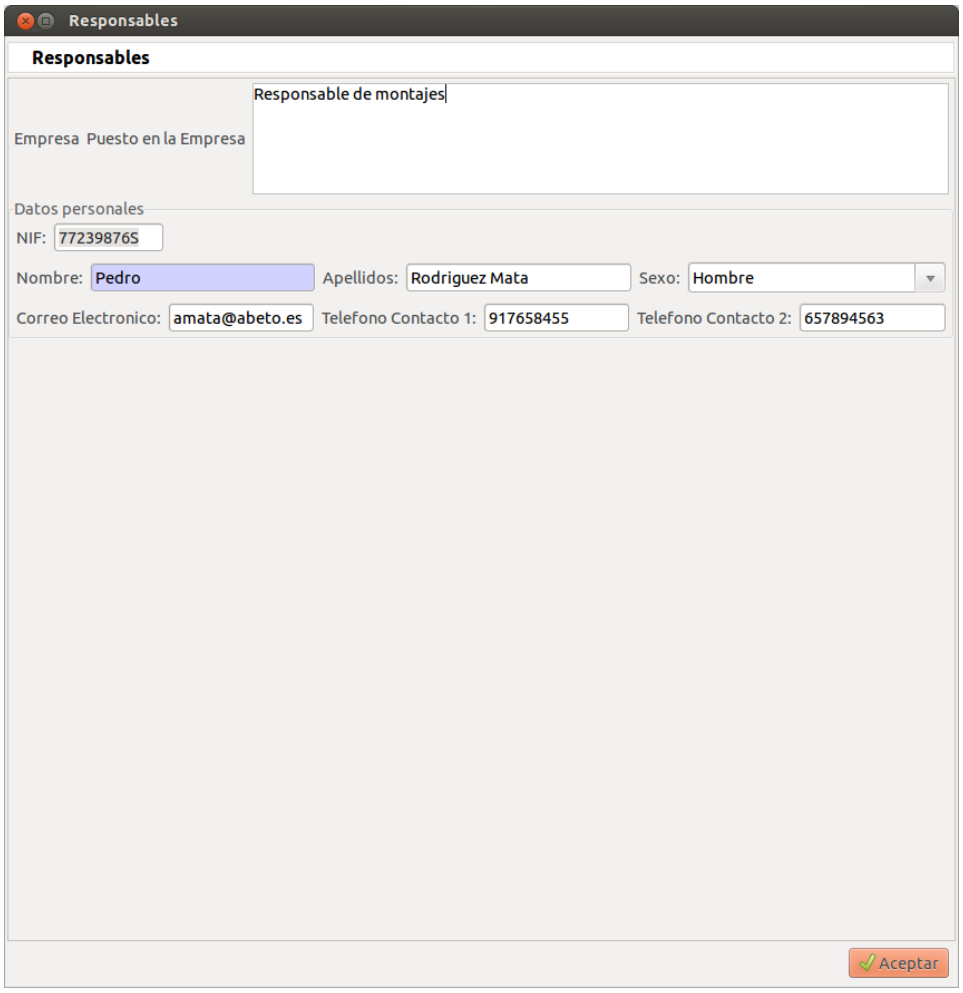



*Figura I.14. Vista de formulario de Responsable desde acceso de Empresa*

Tras la recepción de un correo electrónico de la empresa que requiere de la ejecución de una tarea, un registro de tipo “Pedido” es creado; pudiendo observarse todos ellos, realizados (con número de instalación asociado) y pendientes (sin número de instalación asociado), desde la vista de tipo listado, Fig I.15..

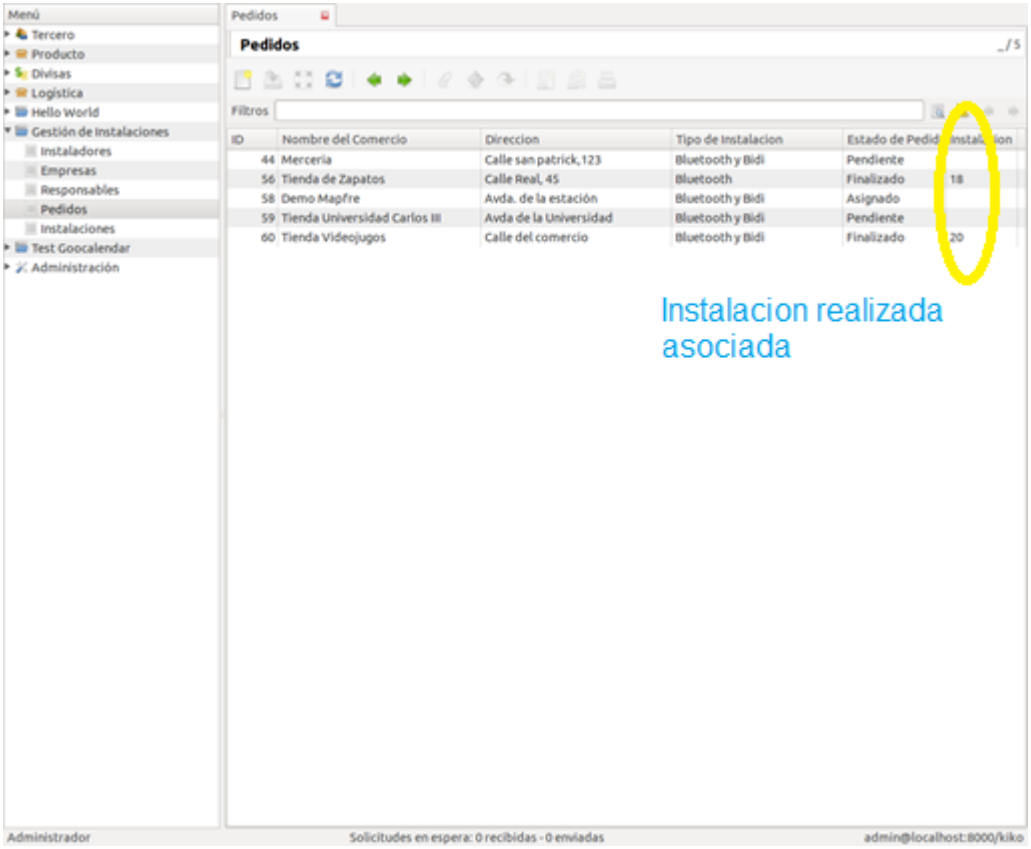



*Figura I.15. Vista listado de entidad Pedido*

Pulsando sobre alguno de estos registros se puede acceder al registro en cuestión del modo tradicional con los datos ya rellenados, pudiéndose dar el caso de ser una petición de instalación de tipo sencillo, Fig I.16. o completo, Fig I.17. Desde esta vista del registro se debe imprimir la hoja de pedido que se deberá enviar a la Empresa que se encargará de ejecutar la instalación.

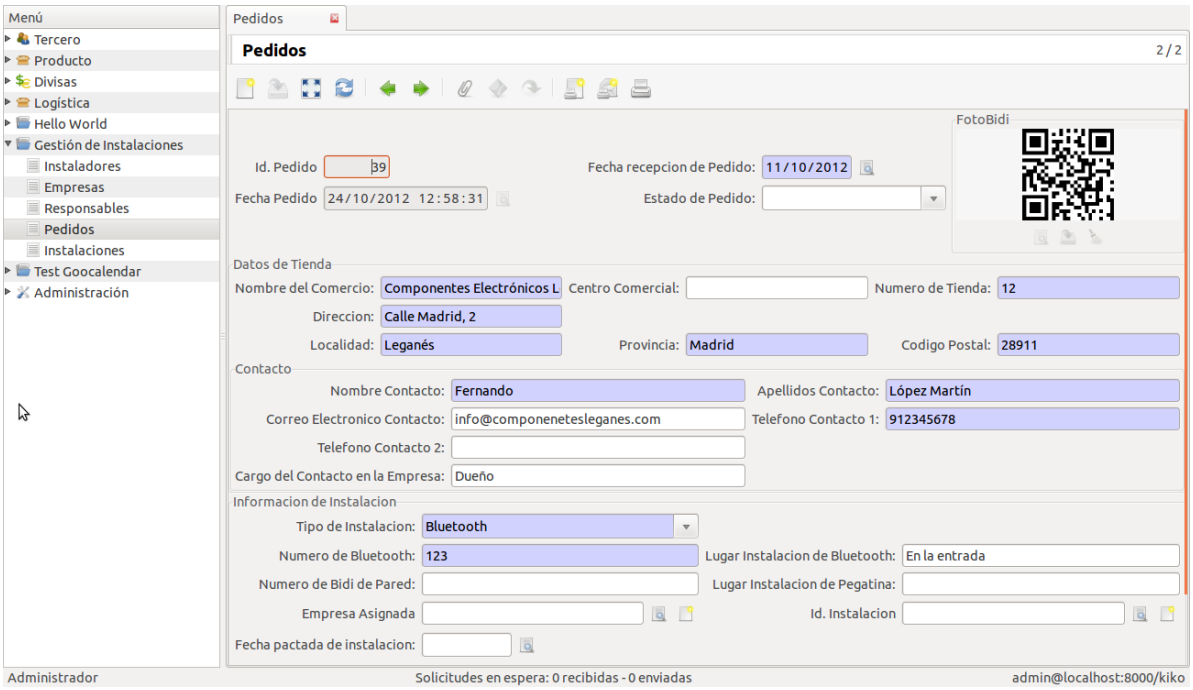



*Figura I.16. Vista formulario tipo sencillo de entidad Pedido*

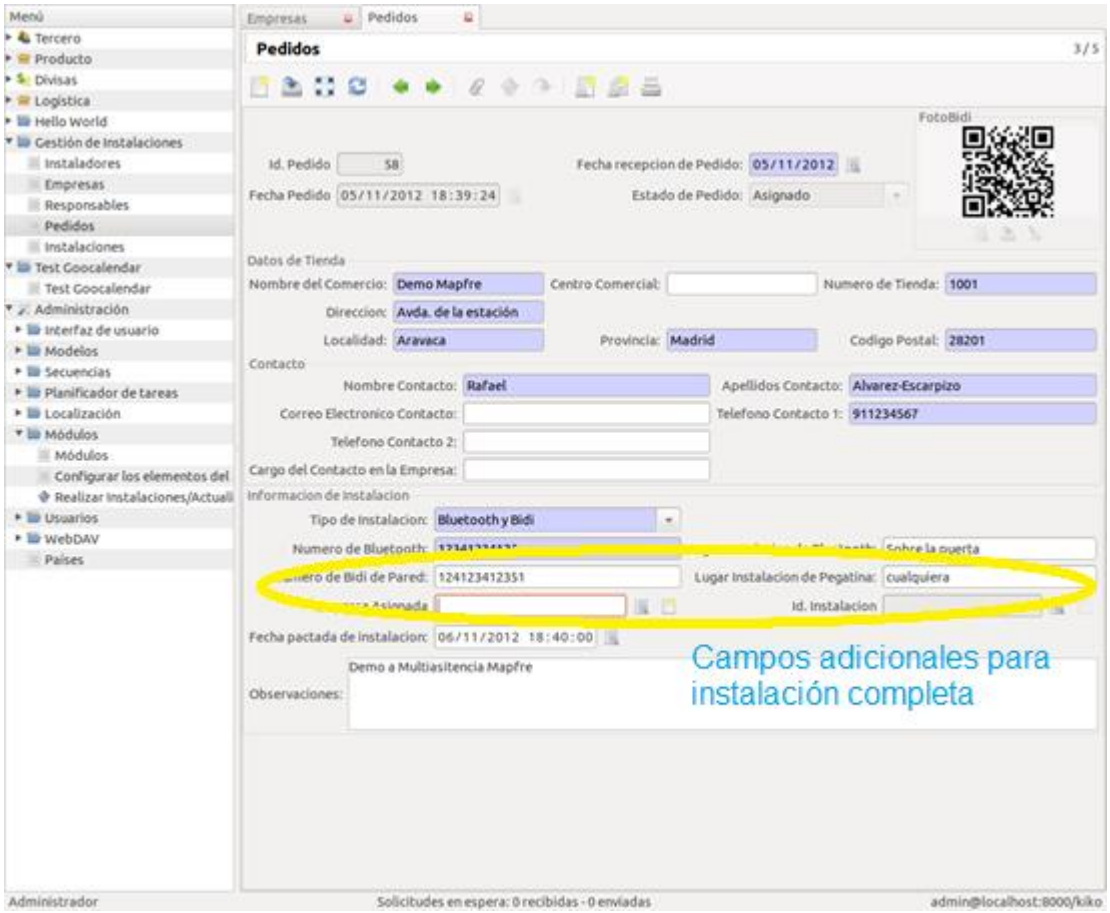



*Figura I.17. Vista formulario tipo completo de entidad Pedido*

En cuanto se haya completado la tarea, se podrá observar un nuevo registro de tipo Instalación desde la vista de tipo listado de la entidad “Instalación”, Fig I.18. y evidentemente desde el listado de pedidos dispondrá de número de instalación asociado, confirmando que la instalación del dispositivo se ha llevado a cabo de forma correcta.

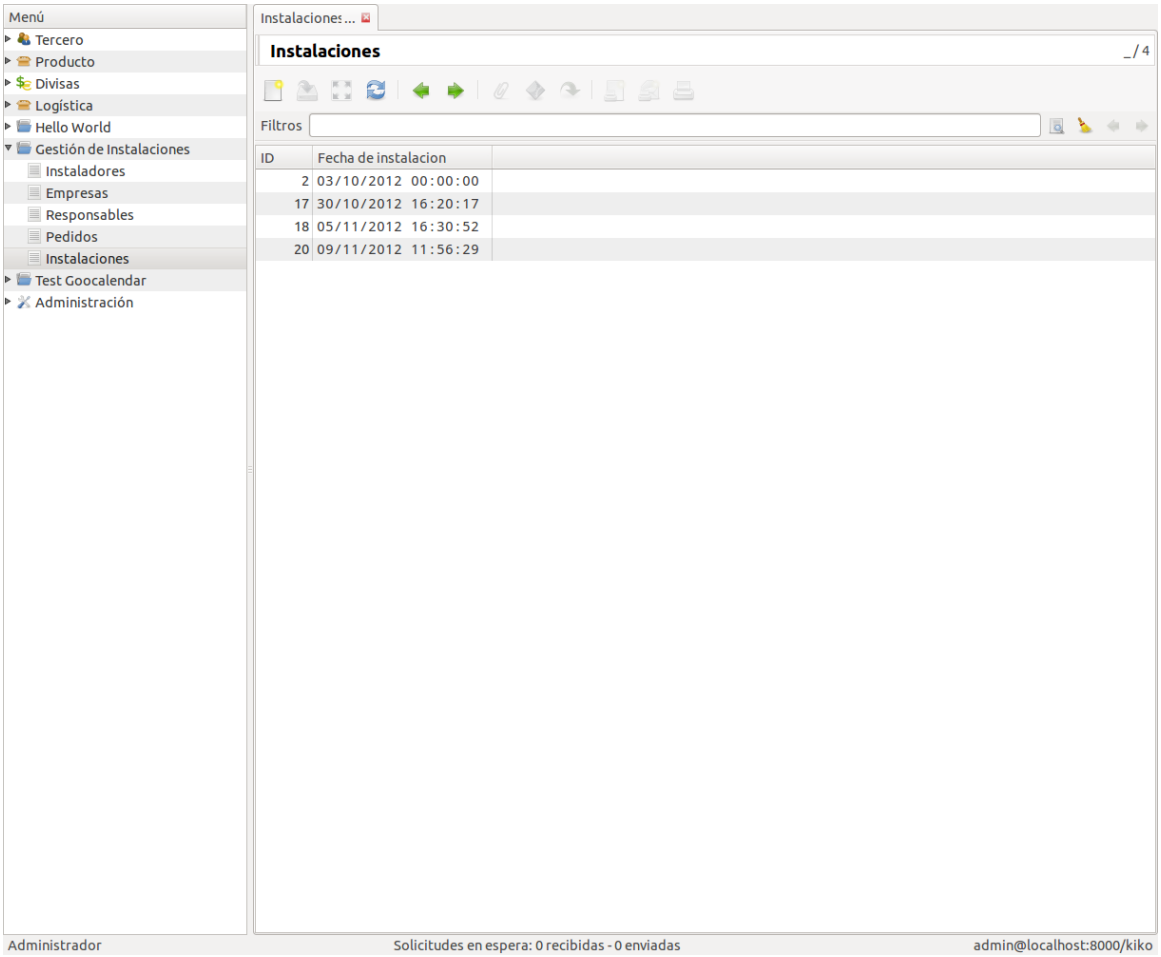



*Figura I.18. Vista listado de entidad Instalacion*

Pulsando sobre la instalación se puede consultar todos los datos introducidos por el personal instalador mediante la aplicación móvil desarrollada, Fig I.19..

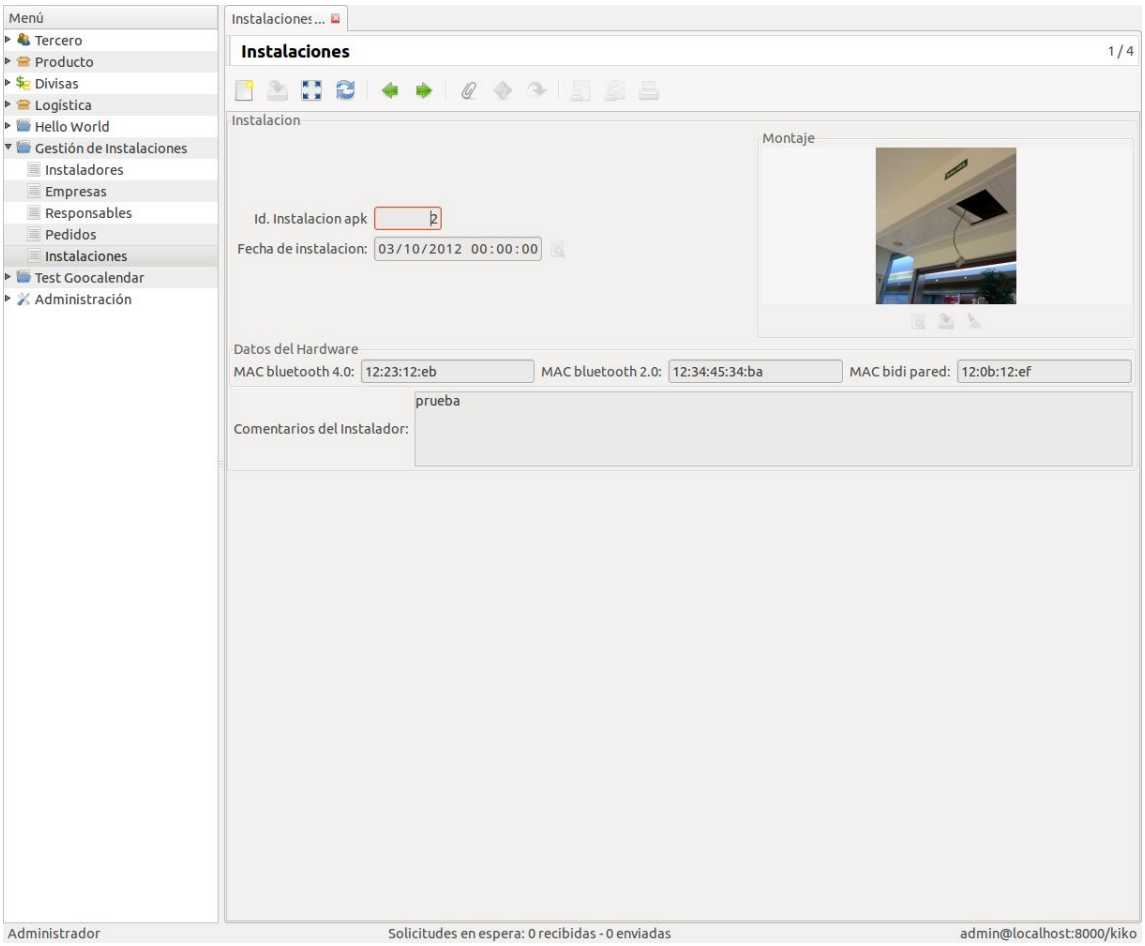



*Figura I.19. Vista formulario de entidad Instalacion*

De esta forma se trabajará y se llevará a cargo la gestión de las instalaciones con el ERP en la empresa Infortrex.

# II. Manual de la aplicación Android

Este Anexo tiene el objetivo de servir de ayuda a los instaladores de la red de proveedores para conocer los aspectos básicos de funcionamiento de la aplicación desarrollada para completar el registro de sus tareas.

Al iniciar la aplicación desde el téminal, se abre automáticamente la pantalla de login, Fig II.1.. En ella se deberá introducir el usuario y el password ofrecido por Infortrex y pulsar el botón 

para acceder a las diferentes opciones de la aplicación.


*Figura II.1. Pantalla de autenticación*

Mientras el sistema lleva a cabo el proceso de autenticación, el usuario dispondrá de información de que tal objetivo se está llevando a cabo, Fig II.2..


*Figura II.2. Proceso de autenticación*

Una vez el usuario logra autenticarse correctamente, se mostrará la pantalla de menú, Fig II.3., desde donde el usuario podrá consultar su historial de instalaciones registradas, realizar una llamada telefónica a las oficinas de Infortrex, o iniciar un registro de una nueva tarea.


*Figura II.3. Pantalla de menú*

Si el usuario decide por realizar una llamada a las oficinas de Infortrex, para aclarar cualquier duda, deberá pulsar el botón  y entonces una llamada telefónica comenzará a completarse, Fig II.4..


*Figura II.4. Llamada telefónica*

Si el usuario decide por consultar su historial de registros, deberá pulsar el botón  y se llevará a cabo el proceso encargado de obtener la información del usuario, informando al usuario, Fig II.5., de que tal objetivo se está llevando a cabo.


*Figura II.5. Proceso de carga de Historial*

Una vez que el servidor de Infortrex haya enviado la información a la aplicación, esta mostrará el listado de todas las tareas realizadas por el usuario en el caso de que las hubiera, Fig II.6., en este listado se puede consultar la fecha y hora en el que el registro de la instalación se llevó a cabo y los datos técnicos de la misma.


*Figura II.6. Pantalla de histórico de registros*

A parte de la información que es visible directamente desde el listado del histórico, es posible acceder al comentario que el propio usuario introdujo en el momento del registro pulsando sobre el elemento a consultar. Una vez pulsado el ítem, se mostrará el comentario del modo que se contempla en la Fig II.7..

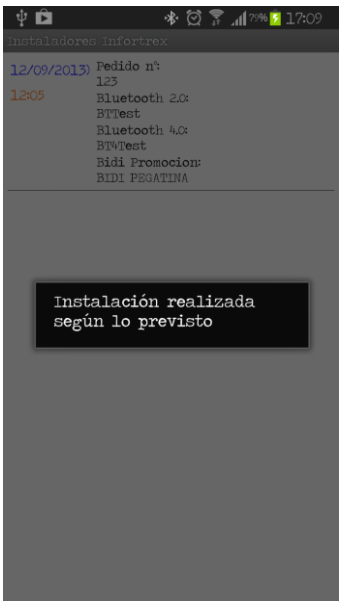



*Figura II.7. Descripción introducida en una instalación*

Cuando un usuario desea iniciar el registro de una nueva tarea, en función de ser una instalación sencilla o completa deberá pulsar el botón 

o 

según corresponda. Durante este manual se explicará el caso de una instalación sencilla, pero el procedimiento para el otro tipo de instalación es prácticamente el mismo.

Una vez pulsado el botón “Instalar Bluetooth”, se inicia el asistente que guiará durante todo el registro, Fig II.8..


*Figura II.8. Pantalla inicial en asistente de instalación sencilla*

Primeramente se pulsará el botón  ya que se debe escanear, Fig II.9., el código Bidi presente en la hoja de pedido que el instalador debe tener impresa y en la que se encuentra todo el detalle sobre cómo debe realizar la instalación.

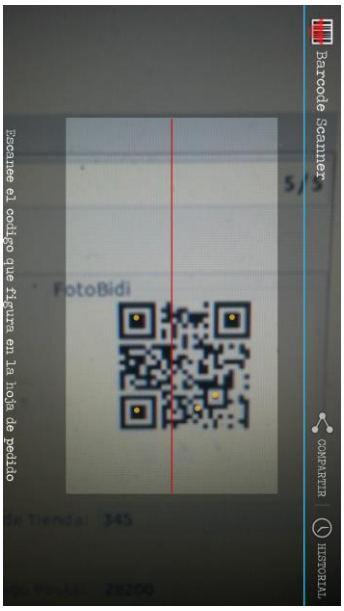



*Figura II.9. Escaneo de un código Bidi*

Tras completar con el escaneo, si el contenido del Bidi corresponde al de una hoja de pedido, entonces el asistente mostrará los siguientes pasos a realizar y la información sobre el identificador del pedido a registrar, Fig II.10..


*Figura II.10. Asistente de instalación sencilla*

El asistente ahora muestra tres opciones nuevas que se deben completar para poder enviar el registro, el orden en el que se debe completar no es importante y es posible repetir cualquiera de ellas en el caso de haber cometido algún error.

Si se pulsa el botón  , se inicia automáticamente la cámara fotográfica con la interfaz simplificada para la aplicación, Fig II.11..

*Figura II.11. Captura de imagen*

Una vez realizada la fotografía a la zona en la que se ha colocado el dispositivo dentro del local comercial, en el asistente de la aplicación, Fig II.12, podrá revisarse la fotográfia capturada, pulsando el botón 

, para poder comprobar si es plenamente visible e identificable la zona en la que se encuentra ubicado y en un futuro llevar a cabo tareas de mantenimiento o de desinstalación del propio dispositivo sin grandes problemas de localización del mismo.


*Figura II.12. Asistente tras la captura de imagen*

Si se pulsa el botón  , se inicia automáticamente el asistente de detección de dispositivos Bluetooth, Fig II.13, en esta pantalla se observa primeramente si el Bluetooth del teléfono móvil se encuentra activado y justamente después un botón que permite iniciar la esta detección.

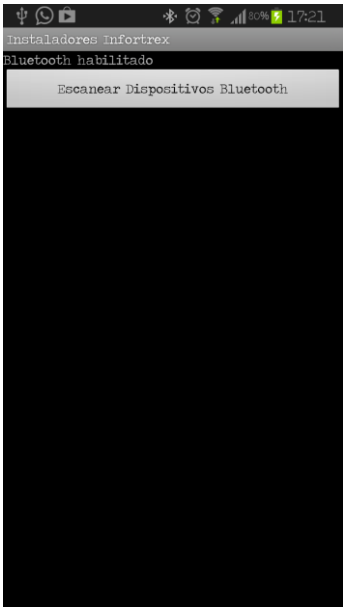



*Figura II.13. Pantalla de escaneo de dispositivos Bluetooth*

Una vez pulsado el botón “Escanear dispositivos Bluetooth”, se inicia el proceso de escaneo, Fig II.14, mostrando un mensaje al usuario informándole de que el objetivo se está llevando a cabo. Este proceso dura un tiempo fijo de unos 25 segundos.

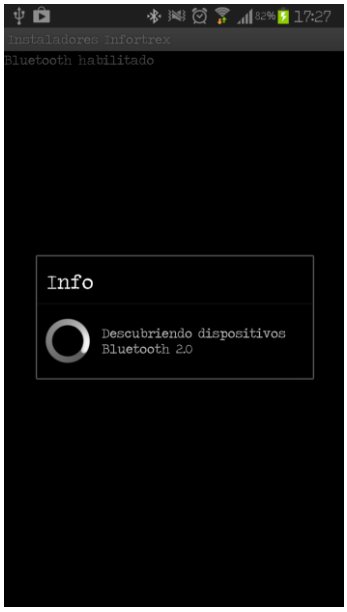



*Figura II.14. Proceso de detección de dispositivos Bluetooth*

Una vez que el proceso de detección ha terminado se muestra en la pantalla el listado de dispositivos Bluetooth propietarios de Infortrex detectados. Ya que se filtra por la información que suministran a la aplicación, en un principio tan sólo deberá aparecer reflejado un dispositivo en este listado el cual deberá ser seleccionado, Fig II.15.


*Figura II.15. Listado de dispositivos Bluetooth encontrados*

La otra tarea que se deberá realizar es la de escanear el código Bidi presente en el propio dispositivo Bluetooth, para ello se deberá pulsar el botón  de la misma forma que se realizó el escaneo del código de la hoja de pedido.

Una vez realizadas todas las tareas, Fig II.16, se permitirá al usuario enviar el registro para notificar a Infortrex que ya ha sido finalizada la tarea. Para ello se debe pulsar el botón 

.


*Figura II.16. Asistente con pasos completados*

Como último paso se permitirá al usuario introducir algún mensaje para informar o simplemente detallar algún aspecto del transcurso de la propia instalación, Fig II.17, y una vez completado se envía la instalación al servidor de Infortrex.


*Figura II.17. Introducción de comentario durante el envío de registro*

Una vez finalizado el registro se recomienda revisar el histórico de instalaciones registradas para comprobar que efectivamente se ha guardado correctamente y que Infortrex ya dispone de la tarea completada.

# III. Entorno de demostración

Durante la defensa de este proyecto final de carrera, se pretende hacer una sencilla demostración del funcionamiento del sistema completo. Para ello se intentará realizar un ejemplo de todo el ciclo de vida de una instalación de un dispositivo. Los puntos a mostrar serán los siguientes:

- Creación de un usuario con rol de instalador en el ERP.

- Envío de correo electrónico a la bandeja de correo de demostración, simulando una petición del cliente.

- Ejecución del cron de lectura de correos y observación de la creación automática de un registro de la entidad Pedido.

- Asignación del pedido al usuario creado y la generación de la hoja de pedido.

- Realización de un proceso de registro de una instalación completa con la aplicación móvil.

- Revisión de la información recibida en el ERP.