Departamento de Informática

PROYECTO DE FIN DE CARRERA

Análisis, Diseño e Implementación de un generador de páginas web estáticas

Autor: ÁLVARO DURÁN TOVAR

Tutor: JOSÉ LUIS LÓPEZ CUADRADO

Leganés, octubre 2015

A mis padres y a mi mujer.

Índice de contenidos

Índice de ilustraciones

Índice de tablas

1. Introducción

En este proyecto de fin de carrera se plantea, analiza e implementa una página web que facilite la creación de otras páginas web estáticas mediante la técnica de arrastrar y soltar bloques de contenido, de forma que al finalizar el proceso se obtendrá un fichero comprimido con el código html y css necesario.

La idea inicial surgió ante la inexistencia de repositorios de plantillas que hagan uso del framework Foundation [1], a pesar de que hay muchas para el framework de Bootstrap[2].

El presente proyecto se llevará a cabo de forma que una vez acabado sea puesto en un entorno de producción real y pueda ser utilizado como una herramienta SAAS.

1.1 Objetivos

El objetivo principal del proyecto es facilitar y agilizar la creación de una página web estática de forma que el usuario obtenga una plantilla, que con unos cambios mínimos (colores, textos…) pueda ponerlo en producción con el menor coste en tiempo y recursos.

El enfoque utilizado para la elección de tecnologías y herramientas es el de una Startup que desea sacar adelante un proyecto en el menor tiempo posible. En el mundo de las startup es importante poder monetizar una idea cuanto antes, por lo que hay que hacer notar que es más importante la facilidad a la hora de implementar el código que el rendimiento de la aplicación o la facilidad de utilización de librerías de software libre creadas por terceros.

1 http://foundation.zurb.com/
2 http://getbootstrap.com/

El presente documento seguirá una versión adaptada de la metodología Métrica v3, que ha sido utilizada en diferentes asignaturas de la carrera y cumple con los mínimos estándares exigibles a un producto software.

1.2 Definición del problema

Actualmente no existe ninguna página web que disponga de plantillas para empezar proyectos con el framework de Foundation. Se creará una aplicación web que ayudará a resolver dicho problema.

Hay diversas razones para la utilización de plantillas de páginas web, algunas de ellas son poder tener un boceto rápido que presentar a un cliente (para que pueda hacerse una idea rápida del resultado final) o también para tener un producto inicial al que solo hay que cambiar el contenido.

La forma de hacerlo será creando bloques conceptuales de código html y css, que una vez juntos formarán la plantilla deseada por el usuario. Dichos bloques son los correspondientes a la mayor parte de páginas de internet: Cabeceras con menus, tabla con lista de precios, zona de contacto, etc.

Dado que se trata de realizar una página web se utilizará el framework de Ruby on Rails, uno de los más utilizados para este tipo de proyectos. Además una vez terminado se subirá a un servidor de internet. Todo lo anterior implica la correcta selección de herramientas para desarrollar el código, un entorno que facilite la creación y ejecución de tests y por último un proveedor de hosting

Dicho proveedor de hosting debe estar en consonancia con el enfoque de Startup que se quiere dar, por lo que debe ser fácil de usar y a un precio razonable.

1.3 Definiciones

A continuación se van a definir los conceptos principales del presente proyecto:

- **Framework**: Conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.

- **Plantilla o template** : Conjunto de ficheros que juntos definen un componente genérico de una página web (tabla, encabezado, pie de página…).

1.4 Recursos

La creación de una aplicación web implica la utilización de tecnologías que optimicen tanto el proceso de desarrollo, como su desempeño en producción. A continuación se explicará que elecciones se han hecho y porque:

Sublime Text 2

Es un editor de texto multiplataforma muy potente que admite la instalación de plugins. Dichos plugins sirven de ayuda a la hora de comprobar la sintaxis del código y de ejecutar tareas repetitivas.

Foundation

Es un framework para maquetado de página web que facilita su creación al incorporar muchos estilos de css. Además está enfocado en la creación de páginas web para móviles.

Ruby on Rails

Framework web que se ha elegido para esta aplicación. Actualmente es uno de los más populares. Permite la utilización de innumerables herramientas que facilitan la creación y reutilización de código css, html, optimiza las conexiones http agregando cabeceras de cacheo, entre otras muchas cosas.

Está estructurado según el patrón MVC (modelo - vista - controlador), ayudando a organizar el código. Además en este framework se intenta seguir la filosofía DRY (don't repeat your self), tratando de maximizar la reutilización de código.

Brackets

Es un editor de texto enfocado a la creación de páginas web estáticas. Es una herramienta muy potente que permite, entre otras cosas, la ejecución del código que se está escribiendo en el navegador, sin tener que depender de un servidor web. Esta herramienta será la que se utilice para la creación de las plantillas.

Google Chrome

El proyecto será desarrollado y probado usando este navegador que facilita enormemente la tarea del programador con su herramienta Dev Tools.

1.5 Estructura

La estructura que se va a seguir para presentar la información del proyecto consta de cinco documentos principales:

- Estudio de Viabilidad del Sistema (EVS).
- Análisis del Sistema de Información (ASI).
- Diseño del Sistema de Información (DSI).
- Manual de Implantación.
- Manual de Usuario.

Como se puede comprobar, los tres primeros documentos corresponden a las fases homónimas previstas en la metodología Métrica v3. En ellos, se expondrán detalladamente los productos construidos, y su desarrollo a lo largo de los meses de trabajo. En cuanto a los dos últimos, el primero presenta un manual para la implantación de la aplicación web en un entorno de producción, ya dispuesta para ser utilizada por el usuario final. Seguidamente, el segundo ofrece un manual de usuario para aprender a manejar la herramienta, con toda su funcionalidad explicada.

Por último, indicar que se finaliza presentando las conclusiones recogidas tras las realización del proyecto, ofreciendo una reflexión sobre su consecución.

1.6 Planificación y Presupuesto

Es necesario la elaboración de un presupuesto para la realización del proyecto. Se estimaran los costes de las horas empleadas por las diferentes personas que trabajarán en el mismo, así como de los equipos informáticos y otros costes asociados.

1.6.1 Plan de trabajo

Estas son las fases en las que se van a dividir las tareas a realizar:

- Estudio de viabilidad del sistema (EVS)

Se determina el alcance del sistema propuesto por este proyecto, además de estudiar cual es la situación actual en el marco de las soluciones a las necesidades del cliente, para posteriormente, especificar los motivos que han hecho que este proyecto haya sido elegido.

- Análisis del sistema de información (ASI)

El objetivo de esta fase es la obtención de todos los requisitos particulares asociados, tales como requisitos de interfaz, funcionales, hardware, etc.

- Diseño del sistema de información (DSI)

Esta es la fase previa a la implementación del proyecto, en la que se describirá la arquitectura del sistema y sus dependencias.

- Implementación

En esta fase se procederá a desarrollar el código necesario haciendo uso de la información recogida en las fases previas.

- Documentación

Esta tarea se realizará a la vez que el resto, dejando constancia de los elementos necesarios para la correcta interpretación del proyecto.

1.6.2 Identificación de participantes

Jefe de proyecto

Encargado de la gestión, es decir, de la organización, planificación y seguimiento del mismo.

Analista

Se encarga de obtener la especificación de requisitos de usuario, establecer los casos de uso, redactar los requisitos software y modelar los procesos y tareas a implementar.

Diseñador

Se encarga de obtener los casos de uso reales, el modelo físico de datos, las distintas interfaces de la aplicación, además de definir la arquitectura del sistema a un nivel más bajo.

Programador

Se encarga de escribir el código necesario para la implementación del análisis propuesto por analista y las directrices fijadas por el diseñador.

1.6.3 Estimación de tareas

El siguiente cuadro muestra la estimación en horas de los participantes mencionados en el apartado anterior:

Fase	Jefe de proyecto	Analista	Diseñador	Programador	Total
Estudio de viabilidad del sistema (EVS)	10	15	0	0	25
Análisis del sistema de información (ASI)	8	15	0	0	23
Diseño del sistema de información	5	10	0	0	15
Implementación	0	10	50	100	150
Documentación	10	20	20	10	60
Total	33	60	70	110	273

Tabla 1: Estimación de tareas

1.6.4 Planificación

A continuación se mostrará el diagrama Gantt elaborado con la planificación del presente proyecto mostrando la duración y consecución de las diferentes tareas a desempeñadas.

Diagrama de Gantt

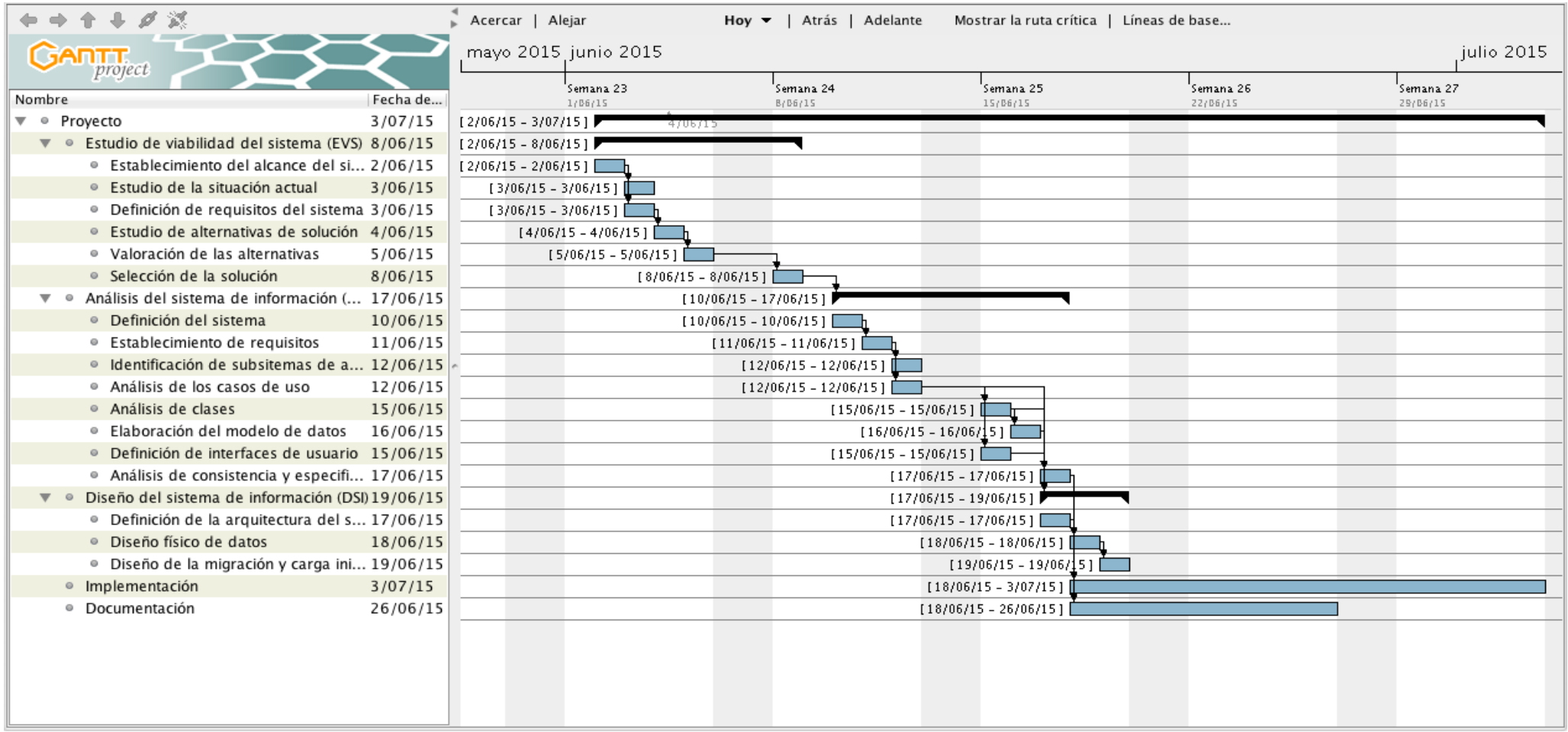


Ilustración 1: Diagrama de Gantt

1.6.5 Cálculo de costes

En el siguiente presupuesto aparecen los costes asociados a las horas dedicadas por el personal del proyecto, además de los relacionados con equipos informáticos. El hosting utilizado para la aplicación en el entorno de producción es gratuito. Se ha utilizado heroku como proveedor de hosting, que permite tener un plan mínimo sin coste asociado.

Rol	Coste / Hora	Horas	Total
Jefe de proyecto	16€	33	528€
Analista	14€	60	840€
Diseñador	12€	70	840€
Programador	12€	110	1320€
		TOTAL	3528€

Tabla 2: Coste de personal

A continuación se presentan los costes de los equipos informáticos utilizados:

Producto	Unidades	Coste / Unidad	Años amortización	Tiempo	Total
Ordenador	1	1500,00€	3	3 meses	500€
Impresora	1	200€	3	3 meses	66,67€
				TOTAL	566,67€

Tabla 3: Coste de soporte informático

Por último se va a proceder al calculo total:

Descripción	Coste
Gastos personal	3528€
Gastos soporte informático	566,67€
Total costes directos	**4094,67€**
Riesgos (10%)	409,46€
Beneficios (10%)	409,46€
Total sin IVA	**4915,59€**
IVA (21%)	1032,27€
TOTAL	5947,86€

Tabla 4: Coste total

2. Estudio de viabilidad del sistema (EVS)

2.1 ACTIVIDAD EVS 1: Establecimiento del alcance del sistema

En este apartado se hará una exposición general del problema que se presente resolver.

Se presentará el estado actual del entorno, los interesados y se recogerán los requisitos del sistema.

2.1.1 Tarea EVS 1.1: Estudio de la Solicitud

Se plantea la creación de una aplicación web que permita la creación de páginas estáticas de forma sencilla e intuitiva.

La forma de crear dichas páginas será mediante la técnica de arrastrar y soltar bloques de código, también llamados templates, de forma que una vez que se ha creado la estructura deseada se pueda descargar un fichero comprimido con todo lo necesario (fichero html, imágenes y fichero de estilos).

Al empezar una página web suele ser costoso crear los estilos iniciales, por lo que este proyecto se basará en uno de los frameworks más utilizados, foundation.

Este framework provee de unos estilos de css que dan un estilo por defecto a las páginas web, así como una estructura en rejilla que facilita enormemente la tarea del programador.

Se debe permitir la compartición de las páginas generadas mediante una url, de forma que la propia url indique los bloques de los que se componen.

Como se ha comentado al inicio del documento se va a usar un enfoque como si se estuviese realizando un proyecto para una startup. Esto obliga a cumplir una serie de requisitos que van asociados a este tipo de empresas:

Rapidez

Las startups requieren poder recuperar su inversión en el menor tiempo posible, porque lo que siempre es importante ahorrar en los tiempos de desarrollo y puesta en producción. Actualmente existen diferentes proveedores de hosting que nos permiten que con un par comandos podemos subir nuestros proyectos a internet, algunos incluso de forma gratuita. Dichos proveedores serán elegidos por encima de otros en los que se requiera dedicar tiempo a la configuración del servidor.

También es importante recalcar que por como están implementados diferentes frameworks y lenguajes pueden ayudar a reducir (o incrementar) el tiempo de desarrollo. En el caso de aplicaciones web ruby on rails es una de las tecnologías con más aceptación actualmente y posee un ecosistema enorme de herramientas que facilitan la tarea de la implementación.

Bajo coste

Todo el mundo está interesado en gastar lo menos posible, pero especialmente las startups que suelen empezar con fondos aportados por los propios socios.

Software libre

El uso de software libre nos aporta rapidez (no hay que reinvatar la rueda), código testeado y perfectamente funcional y poder modificar el código lo creemos

necesario.

2.1.2 Tarea EVS 1.2: Identificación del Alcance del Sistema

- La herramienta debe facilitar la creación de páginas web mediante la creación de bloques comunes en el entorno web.
- Se debe facilitar la compartición de una url que represente los bloques de la página web generada.
- Debe usar tecnologías estándares para asegurar el funcionamiento en la mayor parte de navegadores modernos.
- La puesta en producción debe ser fácil y rápida.

2.1.3 Tarea EVS 1.3: Especificación del Alcance del EVS

Se llevará a cabo un estudio de las tecnologías que se van a usar en el sistema, y se evaluarán diversas alternativas de solución. Será necesario realizar el estudio de la situación actual, con la pretensión de identificar la funcionalidad que cubre la aplicación de escritorio utilizada para definir los procesos.

2.2 ACTIVIDAD EVS 2: ESTUDIO DE LA SITUACIÓN ACTUAL

Actualmente existen múltiples páginas que ofrecen plantillas con las que empezar a trabajar basadas en el framework de bootstrap. También existen herramientas que facilitan la creación de páginas web mediante el paradigma de arrastrar y soltar de forma que se crea una página web mediante la interconexión de los bloques que la componen (cabecera, tablas, tablas de precios, etc).

Sin embargo no existe ninguna página web que ayude a crear páginas estáticas basadas en el framework de foundation.

Gracias a la evolución de los navegadores web es posible crear páginas con contenido altamente interactivo mediante código javascript, tecnología que será necesario para la creación del presente proyecto.

2.3 ACTIVIDAD EVS 3: DEFINICIÓN DE REQUISITOS DEL SISTEMA

2.3.1 Tarea EVS 3.2: Identificación de Requisitos

En este apartado se debe definir los requisitos generales del sistema mediante entrevistas con los usuarios especificados en el apartado **1.6.2 Identificación de participantes.**

Se realizarán sucesivas reuniones, en función de la disponibilidad de los participantes, con el objetivo de resolver posible dudas. Mediante la obtención de estos requisitos se obtendrá una visión general de las características del sistema a desarrollar.

Los requisitos obtenidos se pueden clasificar en dos grandes categorías:

- **Requisitos de capacidad:** representan lo que necesitan los usuarios para resolver un problema o lograr un objetivo.

- **Requisitos de restricción:** son las restricciones impuestas por los usuarios sobre cómo se debe resolver el problema o cómo se debe alcanzar el objetivo.

A su vez cada requisito tiene asociado los siguientes atributos:

- **Identificación:** cada requisito de usuario incluirá una identificación, para facilitar la trazabilidad, que es valor único identificativo del requisito que lo

diferencia del resto.

- **Necesidad:** los requisitos esenciales del usuario son no negociables; el resto pueden ser menos importantes y sujetos a la negociación. Es decir, define la importancia de que el requisito sea implementado por el sistema. Esta propiedad puede orientar el éxito o fracaso del proyecto. Los valores posibles son: Esencial, Deseable u Opcional.

- **Prioridad:** cada requisito de usuario incluirá una medida de la prioridad que posee, para que el desarrollador pueda decidir la planificación del desarrollo. Sus valores pueden ser: Alta, Media o Baja.

- **Estabilidad:** algunos requisitos pueden no estar sujetos a cambios durante el proyecto y otros puede que si, según sean sus dependencias en las fases de desarrollo del proyecto. Los requisitos se clasificarán en: estables, inestables.

- **Fuente:** identifica el origen del requisito, que puede estar en el usuario, una fuente externa como un documento o el propio equipo de desarrollo durante la elaboración de requisitos.

- **Claridad:** identificará la falta o existencia de ambigüedad en un requisito. Cuanta menos ambigüedad existe, más claridad hay. La clasificación por tanto será: claros, ambiguos.

- **Verificabilidad:** indicará si un requisito incorporado a la aplicación, es fácilmente reconocible y por tanto es verificable su presencia en diseño y aplicación. La verificabilidad se clasificará en: alta, media y baja.

A continuación se presenta el catálogo de requisitos de usuario:

Identificador: RU-C001	
Prioridad: ☒ Alta ☐ Media ☐ Baj	Fuente: Cliente
Necesidad: ☒ Esencial ☐ Deseable ☐ Opcional	
Claridad: ☒ Alta ☐ Media ☐ Baj	Verificabilidad: ☒ Alta ☐ Media ☐ Baj
Estabilidad	Durante toda la vida del sistema
Descripción	El administrador podrá gestionar las categorías de los bloques

Tabla 5: Requisitos de Usuario de Capacidad RU-C001

Identificador: RU-C002	
Prioridad: ☒ Alta ☐ Media ☐ Baj	Fuente: Cliente
Necesidad: ☒ Esencial ☐ Deseable ☐ Opcional	
Claridad: ☒ Alta ☐ Media ☐ Baj	Verificabilidad: ☒ Alta ☐ Media ☐ Baj
Estabilidad	Durante toda la vida del sistema
Descripción	El administrador del sistema podrá administrar las plantillas de la aplicación

Tabla 6: Requisitos de Usuario de Capacidad RU-C002

Identificador: RU-C003	
Prioridad: ☒ Alta ☐ Media ☐ Baj	Fuente: Cliente
Necesidad: ☒ Esencial ☐ Deseable ☐ Opcional	
Claridad: ☒ Alta ☐ Media ☐ Baj	Verificabilidad: ☒ Alta ☐ Media ☐ Baj
Estabilidad	Durante toda la vida del sistema
Descripción	El usuario podrá gestionar los bloques del esquema que se está creando

Tabla 7: Requisitos de Usuario de Capacidad RU-C003

Identificador: RU-C004	
Prioridad: ☒ Alta ☐ Media ☐ Baj	Fuente: Cliente
Necesidad: ☒ Esencial ☐ Deseable ☐ Opcional	
Claridad: ☒ Alta ☐ Media ☐ Baj	Verificabilidad: ☒ Alta ☐ Media ☐ Baj
Estabilidad	Durante toda la vida del sistema
Descripción	El usuario podrá compartir la url del proyecto generado

Tabla 8: Requisitos de Usuario de Capacidad RU-C004

Identificador: RU-C005	
Prioridad: ☒ Alta ☐ Media ☐ Baj	Fuente: Cliente
Necesidad: ☒ Esencial ☐ Deseable ☐ Opcional	
Claridad: ☒ Alta ☐ Media ☐ Baj	Verificabilidad: ☒ Alta ☐ Media ☐ Baj
Estabilidad	Durante toda la vida del sistema
Descripción	El usuario podrá descargar un archivo comprimido con el proyecto generado

Tabla 9: Requisitos de Usuario de Capacidad RU-C005

2.4 ACTIVIDAD EVS 4: ESTUDIO DE ALTERNATIVAS DE SOLUCIÓN

Se estudiarán ahora las diferentes partes de las que estará compuesto nuestro sistema y entorno de desarrollo, analizando distintas alternativas eligiendo las que mejor se adecuen a los requisitos antes especificados.

Proveedor de hosting: Actualmente existen múltiples tipos de proveedores, facilitándonos la tarea de configurar el entorno de producción. Las opciones van desde configurar todo desde cero hasta contratar una plataforma ya configurada que permite utilizar su sistema por una cantidad de horas. La primera opción suele ser la más barata y complicada, la última la más fácil y generalmente más cara.

- **Conectar nuestro propio servidor a internet:** Implica tener un amplio conocimiento de seguridad y conocer a fondo la plataforma elegida. Normalmente esta es la opción que más horas necesita. Realmente esta

opción no es la más cara si hay que pagar las horas dedicadas a la configuración del servidor.

- **Contratar un servidor dedicado:** Hay diferentes opciones y diferentes precios, algunos ejemplos serían 1and1 (aproximadamente 30€ / mes) y hostalia (70€ / mes). Además al igual que en la opción anterior hay que tener altos conocimientos sobre administración de sistemas.

- **Contratar un servidor virtual:** Los servidores virtuales son más baratos que los servidores dedicados, pero ofrecen menos capacidad y potencia. Algunos ejemplos: brightbox (9.5€ / mes), digital ocean (5€ / mes). Esta opción sigue requiriendo experiencia en la administración de sistemas.

- **Contratar un servicio tipo PAAS (Platform as a service):** Esta opción permite desplegar aplicaciones con unos conocimientos mínimos, en un tiempo muy pequeño y generalmente con un coste pequeño. Además son muy fáciles de escalar, por lo que si se necesita de más capacidad o de más potencia es un proceso trivial, eso sí, a un precio más elevado que las opciones anteriores. Algunos ejemplos: Heroku (gratis la opción más básica), modulus (25€ / mes).

Lenguaje de programación: La elección es quizá la opción más importante, puesto que nos va a limitar el resto de partes de nuestro sistema y nos proveerá de diferentes herramientas que nos ayuden con el desarrollo de la aplicación web.

- **Java:** Lenguaje multiplataforma que permite ejecutar código a velocidades cercanas al código nativo. Existen muchas librerías para este lenguaje y varios frameworks tales como Hibernate, Spring y struts.

- **Ruby:** Lenguaje multiplataforma interpretado que posee uno de los frameworks web más utilizado: Ruby on Rails. Al ser interpretado es menos eficiente en tareas pesadas con gran cantidad de cálculos. Sin embargo la sintaxis de este lenguaje facilita enormemente el desarrollo rápido de código. El framework Ruby on Rails provee una cantidad enorme de herramientas para el desarrollo de páginas web.

- **Javascript:** Tradicionalmente javascript ha estado limitado a ser ejecutado en los navegadores web, pero recientemente se han creado frameworks que permiten ejecutar código javascript en cualquier máquina que disponga del interprete. El framework más usado es Node.js y a pesar de ser un lenguaje interpretado es capaz de ejecutar código casi a la misma velocidad de el código nativo. Al igual que Ruby on Rails posee gran cantidad de herramientas y librerías, aunque aún es un entorno muy variable y no todas las herramientas están igual de maduras.

- **PHP:** Puede que sea el lenguaje de programación web por excelencia. En cuanto a capacidades y facilidades que provee es similar a ruby on rails.

Sistema de plantillas: Los bloques de los que se componen la página web a ser generada por esta aplicación son plantillas formadas por html, imágenes y css. Hay dos posibilidades a la hora de desarrollar estas plantillas:

- **Guardar las plantillas en base de datos:** Esta opción permite que cualquier persona pueda generar plantillas nuevas, sin necesitar acceso al código. Por contra puede dificultar la modificación de plantillas ya creadas. Además obviamente necesita de una base de datos, lo que puede aumentar la complejidad del sistema, así como el coste.

- **Guardar las plantillas en ficheros:** Esta opción facilita la modificación de plantillas ya creadas, al tener fácil acceso a los ficheros, aunque obliga a dar acceso a la persona encargada de generar nuevo contenido. Además obliga a tener un mayor cuidado en la implementación para evitar cualquier tipo de falla de seguridad al integrar los ficheros de plantillas con la aplicación web.

2.5 ACTIVIDAD EVS 5: VALORACIÓN DE LAS ALTERNATIVAS

A continuación se valorarán las alternativas teniendo siempre presente que se busca la rapidez y el ahorra de costes, por ejemplo de la optimización y fiabilidad, a una startup le interesa monetizar la antes posible, no que su plataforma sea la mejor del mercado.

Las características a valorar para cada alternativa son: coste, tiempo, complejidad, compatibilidad. La puntuación irá de 1 a 3, siendo siempre preferible el valor más bajo.

Hosting					
	Coste	Tiempo	Complejidad	Compatibilidad	**TOTAL**
Servidor propio	2	3	3	1	9
Servidor dedicado	2	3	3	1	9
Servidor virtual	1	2	2	1	6
PAAS	1	1	1	3	5

Tabla 10: Valoración de hostings

Esta valoración no necesita de muchas aclaraciones, puesto que se han explicado los pros y contras ampliamente en la sección 2.4

Lenguaje de programación					
	Coste	Tiempo	Complejidad	Compatibilidad	**TOTAL**
Java	1	3	2	1	7
Ruby	1	1	1	1	4
Javascript	1	2	2	2	7
PHP	1	1	1	1	4

Tabla 11: Valoración de lenguajes de programación

Explicación sobre las valoraciones:

- Java: Debido a que sigue la filosofía configuración antes que convención requiere más tiempo a la hora de desplegar una aplicación, además a la hora de integrar diferentes librerías hace que pueda llegar a ser complicada

su utilización en entornos web. Por otra parte dispone de una cantidad inmensa de librerías.

- Ruby: No tiene coste de licencia y debido a la expresividad del lenguaje agiliza la tarea del desarrollador. Con unos pocas comandos es posible tener una aplicación web lista. Al igual que java dispone de una cantidad enorme de herramientas y librerías.

- Javascript: En el entorno de Node.js se dispone de cada vez más herramientas y más maduras, pero por la misma razón puede ser complicado encontrar información actualizada y muchas librerías están pobremente documentadas.

- PHP. Al ser un lenguaje que con una trayectoria larga a la hora de desarrollar aplicaciones web dispone de una gran cantidad de herramientas y librerías.

Sistema de plantillas					
	Coste	Tiempo	Complejidad	Compatibilidad	**TOTAL**
Base de datos	1	2	2	1	6
Ficheros	1	1	1	1	4

Tabla 12: Valoración de sistema de plantillas

Explicación sobre las valoraciones:

Como ya se ha comentado previamente el agregar una base de datos aumenta la complejidad y posiblemente el tiempo de desarrollo.

2.6 ACTIVIDAD EVS 6: SELECCIÓN DE LA SOLUCIÓN

Hosting: La mejor opción es usar un servicio PAAS, heroku en este caso es perfecto puesto que ofrece una solución gratuita aunque limitada.

Lenguaje de programación: En este caso se elige ruby al ser un lenguaje muy popular y agradable de utilizar.

Sistema de plantillas: Obviamente la opción elegida es usar el sistema de ficheros para almacenar las plantillas. También se podría haber optado por almacenarlas en una base de datos en vez de en fichero directamente. Esto da lugar a algunos problemas, como que para editar las plantillas ya creadas habría que integrar alguna solución como Ace[3] para poder editar desde el navegador.

3 http://ace.c9.io/#nav=about

3. ANÁLISIS DEL SISTEMA DE INFORMACIÓN (ASI)

3.1 ACTIVIDAD ASI 1: Definición del sistema

En este apartado usaremos la información obtenida en el apartado sobre la viabilidad del sistema, con la intención de obtener una descripción de los problemas que el sistema va a resolver.

3.1.1 Tarea ASI 1.1: Determinación del alcance del sistema

El sistema a desarrollar está pensando para ayudar a la creación e páginas web estáticas, desde el punto de vista de una startup, como se ha comentado anteriormente.

Dicho sistema facilitará la tarea mediante el patrón de arrastrar y soltar, hasta que el usuario obtenga la estructura de página deseada. Posteriormente puede descargar un archivo comprimido con los ficheros necesarios. Concretamente el archivo contendrá un único fichero html con todos los componentes previamente seleccionados, las imágenes asociadas, así como un único fichero css con los estilos de los diferentes bloques seleccionados.

Los bloques que puede seleccionar el usuario estarán ordenados por categorías: cabeceras, contenidos, tablas de precios, proyectos, contactos, equipo y pies de página.

Según vaya el usuario generando la estructura deseada la url de la página debe cambiar, para que pueda ser compartida en las redes sociales.

Las plantillas que representan los bloques deben ser independientes entre sí,

por lo que siempre que sea necesario aplicar estilos de css habrá que asignarlos a un id único, de forma que pueda haya colisiones entre diferentes plantillas.

Este sistema no permitirá la modificación de estilos antes de generar el fichero comprimido. Sería posible elegir la fuente, colores, logos, etc. pero no entra dentro del ámbito de este proyecto.

3.1.2 Tarea ASI 1.2: Identificación del entorno tecnológico

El sistema desarrollará mediante el framework de ruby on rails, en un ordenador mac book pro. La versión del sistema operativa es la 10.10 (OS X Yosemite).

La aplicación se ejecutará en el servicio online heroku. Para poder utilizarlo hace falta tener instalado git, herramienta que es utilizada para desplegar la aplicación en sus servidores.

Se usará el navegador web Google Chrome para su desarrollo.

Para las interacciones con la página web se usará la librería javascript jquery y jquery-ui.

3.1.3 Tarea ASI 1.3: Especificación de estándares y normas

Se ha utilizado Métrica 3 para la documentación y organización del proyecto.

3.1.4 Tarea ASI 1.4: Identificación de los usuarios participantes y finales

- **Jefe de Proyecto** : Es el responsable de la gestión del proyecto, que debe encargarse de mantener la cordialidad y el buen ambiente de trabajo en el equipo, así como saber sacar el máximo rendimiento a cada uno de los miembros del mismo. Es el encargado de que el proyecto siga el camino marcado, asegurando el éxito del mismo.
- **Equipo de desarrollo** : Será el encargado de construir físicamente la aplicación.
- **Cliente**: Es la persona u organización que financia la construcción del sistema.
- **Usuarios**: Son las personas que hacen uso de la aplicación final. Se pueden definir tres tipos de usuarios:
  - Administrador General de la aplicación que se encargará de la gestión de los usuarios.
  - Experto que podrá crear procesos.
  - Experto invitado que podrá visualizar los procesos a los que haya sido invitado, así como mostrar su opinión sobre los mismos.

3.2 ACTIVIDAD ASI 2: Establecimiento de requisitos

La definición de los requisitos software es tarea del desarrollador, ya que en esta actividad se construye una descripción sobre lo que el software debe hacer, apoyándose en los requisitos de usuario identificados anteriormente.

3.2.1 Tarea ASI 2.1: Obtención de requisitos

A continuación, se definirá los requisitos de software que concretan y delimitan el desarrollo de la aplicación, ya que definen qué debe hacer el producto final.

Los requisitos de software nos permiten verificar el diseño y el producto, pues ambos deben cubrir los requisitos que se hayan marcado. En este apartado, como normal general, no se incluyen los aspectos relativos al “cómo”, salvo que supongan una restricción para el sistema.

Previamente a su redacción se debe establecer el formato que se seguirá para su especificación. Los requisitos software deben incluir los siguientes atributos en su definición:

- Identificación: cada requisito software incluirá una identificación para facilitar su traza por las fases subsiguientes.

- Necesidad: los requisitos esenciales de software se marcarán como tales. Los requisitos esenciales no son negociables. El resto pueden estar sujetos a negociación.

- Prioridad: cada requisito de software incluirá una medida de la prioridad para que el desarrollador pueda decidir la planificación de la producción.

- Estabilidad: algunos requisitos se pueden saber fijos sobre la vida esperada del software, mientras que otros pueden depender de las decisiones de diseño o implementación que se tomen durante el desarrollo.

- Descripción: descripción del requisito. Puede ser textual o acompañarse de diagramas y prototipos. La descripción debe ser

  - Clara: un requisito es claro si tiene una y sólo una interpretación. Si un término utilizado en cierto contexto tiene múltiples interpretaciones, se debe aclarar su significado o debe ser reemplazado con un término más específico.

  - Verificable: debe ser posible que se pueda verificar que el requisito ha sido incorporado en el diseño y que se pueda demostrar que el software aplicar el requisito.

Identificador: RS-F001	
Prioridad: ☒ Alta ☐ Media ☐ Baj	Fuente: Cliente
Necesidad: ☒ Esencial ☐ Deseable ☐ Opciona	
Claridad: ☒ Alta ☐ Media ☐ Baj	Verificabilidad: ☒ Alta ☐ Media ☐ Baj
Estabilidad	Durante toda la vida del sistema
Descripción	Existirá un menú que contendrá las categorías de los bloques.

Tabla 13: Requisito de Software Funcional RS-F001

Identificador: RS-F002	
Prioridad: ☒ Alta ☐ Media ☐ Baj	Fuente: Cliente
Necesidad: ☒ Esencial ☐ Deseable ☐ Opciona	
Claridad: ☒ Alta ☐ Media ☐ Baj	Verificabilidad: ☒ Alta ☐ Media ☐ Baj
Estabilidad	Durante toda la vida del sistema
Descripción	El administrador podrá añadir categorías al sistema

Tabla 14: Requisito de Software Funcional RS-F002

Identificador: RS-F003	
Prioridad: ☒ Alta ☐ Media ☐ Baj	Fuente: Cliente
Necesidad: ☒ Esencial ☐ Deseable ☐ Opciona	
Claridad: ☒ Alta ☐ Media ☐ Baj	Verificabilidad: ☒ Alta ☐ Media ☐ Baj
Estabilidad	Durante toda la vida del sistema
Descripción	El administrador podrá quitar categorías del sistema

Tabla 15: Requisito de Software Funcional RS-F003

Identificador: RS-F004	
Prioridad: ☒ Alta ☐ Media ☐ Baj	Fuente: Cliente
Necesidad: ☒ Esencial ☐ Deseable ☐ Opciona	
Claridad: ☒ Alta ☐ Media ☐ Baj	Verificabilidad: ☒ Alta ☐ Media ☐ Baj
Estabilidad	Durante toda la vida del sistema
Descripción	El administrador podrá añadir plantillas al sistema

Tabla 16: Requisito de Software Funcional RS-F004

Identificador: RS-F005	
Prioridad: ☒ Alta ☐ Media ☐ Baj	Fuente: Cliente
Necesidad: ☒ Esencial ☐ Deseable ☐ Opciona	
Claridad: ☒ Alta ☐ Media ☐ Baj	Verificabilidad: ☒ Alta ☐ Media ☐ Baj
Estabilidad	Durante toda la vida del sistema
Descripción	El administrador podrá eliminar plantillas del sistema

Tabla 17: Requisito de Software Funcional RS-F005

Identificador: RS-F006	
Prioridad: ☒ Alta ☐ Media ☐ Baj	Fuente: Cliente
Necesidad: ☒ Esencial ☐ Deseable ☐ Opciona	
Claridad: ☒ Alta ☐ Media ☐ Baj	Verificabilidad: ☒ Alta ☐ Media ☐ Baj
Estabilidad	Durante toda la vida del sistema
Descripción	Cada plantilla debe estar asociada a una categoría

Tabla 18: Requisito de Software Funcional RS-F006

Identificador: RS-F007	
Prioridad: ☒ Alta ☐ Media ☐ Baj	Fuente: Cliente
Necesidad: ☒ Esencial ☐ Deseable ☐ Opciona	
Claridad: ☒ Alta ☐ Media ☐ Baj	Verificabilidad: ☒ Alta ☐ Media ☐ Baj
Estabilidad	Durante toda la vida del sistema
Descripción	Los usuarios podrán agregar bloques al esquema del proyecto

Tabla 19: Requisito de Software Funcional RS-F007

Identificador: RS-F008	
Prioridad: ☒ Alta ☐ Media ☐ Baj	Fuente: Cliente
Necesidad: ☒ Esencial ☐ Deseable ☐ Opciona	
Claridad: ☒ Alta ☐ Media ☐ Baj	Verificabilidad: ☒ Alta ☐ Media ☐ Baj
Estabilidad	Durante toda la vida del sistema
Descripción	Los usuarios podrán eliminar bloques del esquema del proyecto

Tabla 20: Requisito de Software Funcional RS-F008

Identificador: RS-F009	
Prioridad: ☒ Alta ☐ Media ☐ Baj	Fuente: Cliente
Necesidad: ☒ Esencial ☐ Deseable ☐ Opciona	
Claridad: ☒ Alta ☐ Media ☐ Baj	Verificabilidad: ☒ Alta ☐ Media ☐ Baj
Estabilidad	Durante toda la vida del sistema
Descripción	Los usuarios podrán ordenar los bloques del esquema

Tabla 21: Requisito de Software Funcional RS-F009

Identificador: RS-F010	
Prioridad: ☒ Alta ☐ Media ☐ Baj	Fuente: Cliente
Necesidad: ☒ Esencial ☐ Deseable ☐ Opciona	
Claridad: ☒ Alta ☐ Media ☐ Baj	Verificabilidad: ☒ Alta ☐ Media ☐ Baj
Estabilidad	Durante toda la vida del sistema
Descripción	El estado del esquema del proyecto se guardará en la URL del navegador

Tabla 22: Requisito de Software Funcional RS-F010

Identificador: RS-F011	
Prioridad: ☒ Alta ☐ Media ☐ Baj	Fuente: Cliente
Necesidad: ☒ Esencial ☐ Deseable ☐ Opciona	
Claridad: ☒ Alta ☐ Media ☐ Baj	Verificabilidad: ☒ Alta ☐ Media ☐ Baj
Estabilidad	Durante toda la vida del sistema
Descripción	El usuario podrá compartir la estructura generada mediante una url

Tabla 23: Requisito de Software Funcional RS-F011

Identificador: RS-F012	
Prioridad: ☒ Alta ☐ Media ☐ Baj	Fuente: Cliente
Necesidad: ☒ Esencial ☐ Deseable ☐ Opciona	
Claridad: ☒ Alta ☐ Media ☐ Baj	Verificabilidad: ☒ Alta ☐ Media ☐ Baj
Estabilidad	Durante toda la vida del sistema
Descripción	El usuario podrá descargar la estructura generada

Tabla 24: Requisito de Software Funcional RS-F012

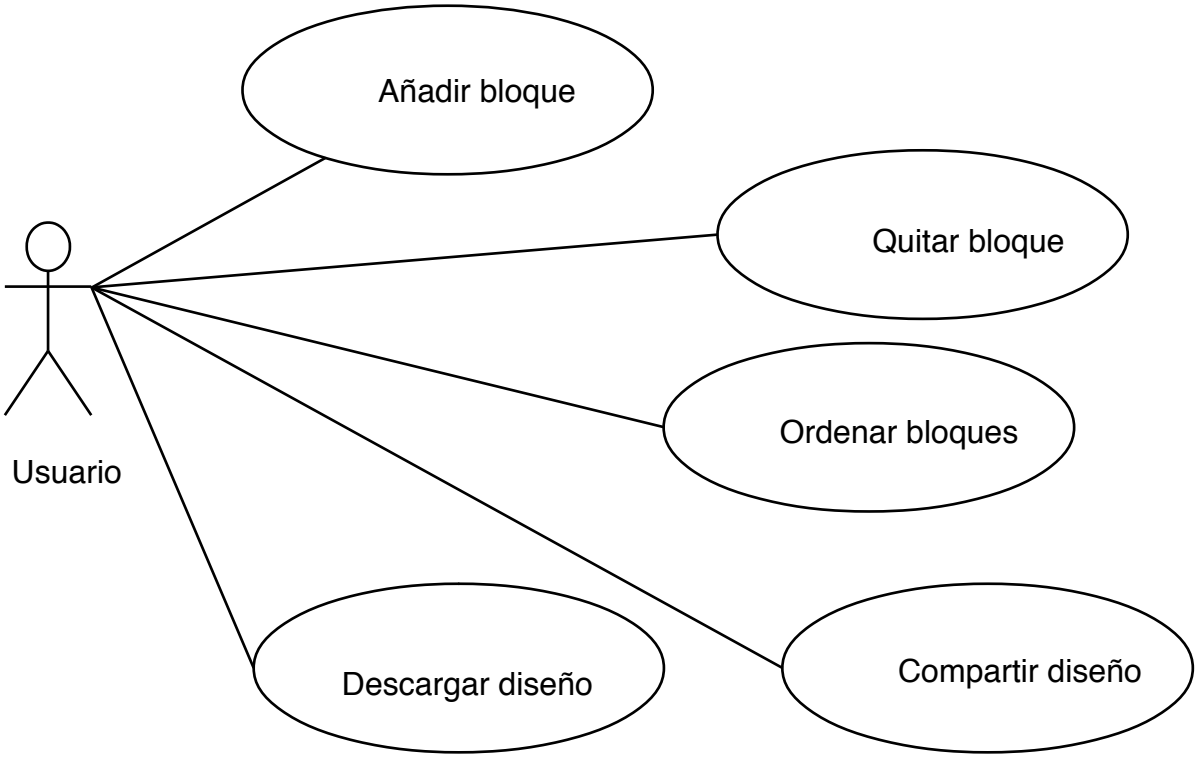
3.2.2 Tarea ASI 2.2: Obtención de casos de uso

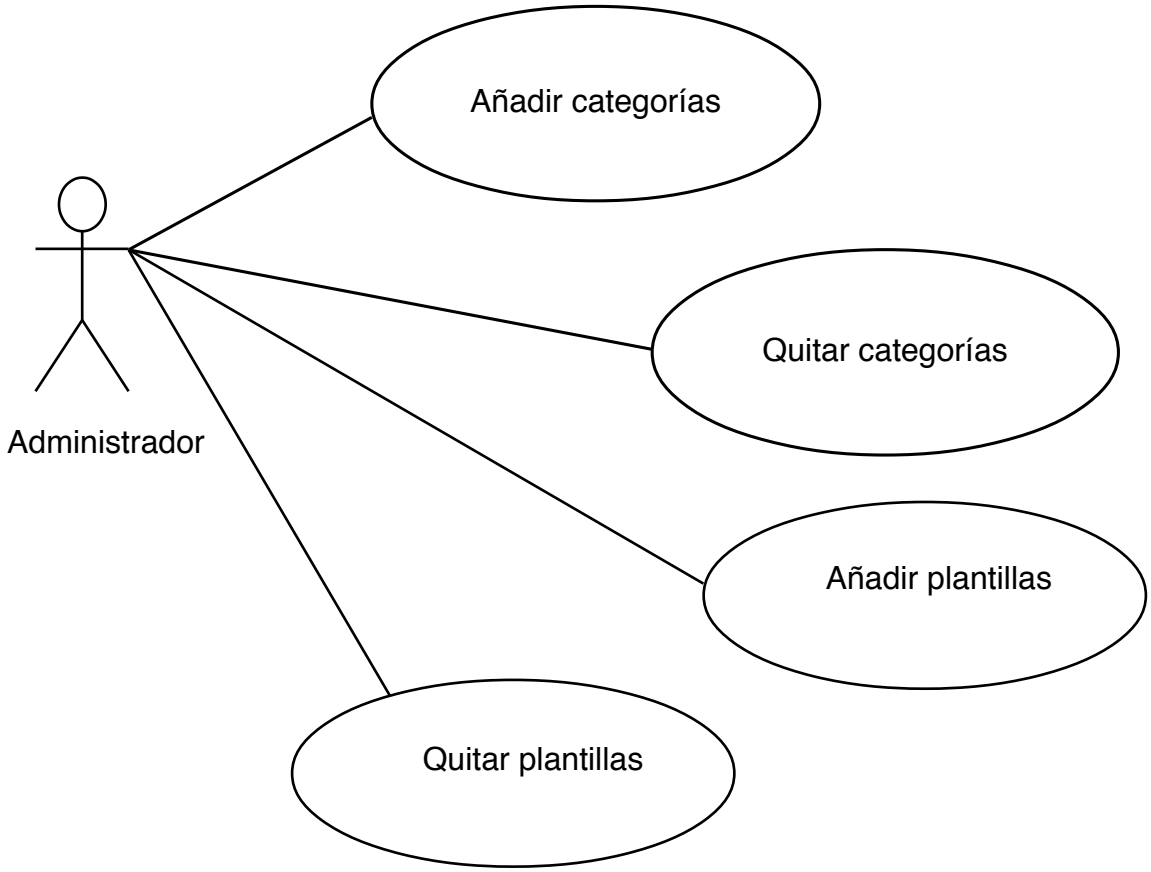
A continuación se realiza el estudio de los casos de uso obtenidos a partir de los requisitos de usuario identificados durante el estudio de viabilidad del sistema.

En primer lugar, se muestra el diagrama de casos de uso diseñado en el

lenguaje de modelación UML, ya que es una herramienta que permite mostrar de una manera sencilla la representación de las necesidades que va a solucionar el sistema.

Seguidamente, para completar los casos de uso se especificará de cada uno los actores que participan de ese caso de uso, la descripción del escenario, las precondiciones y poscondiciones que se presentan y finalmente el escenario básico del caso de uso.





- Añadir plantilla:
  - Identificador: CU-001
  - Actores: Administrador
  - Descripción: El administrador añade una nueva categoría al sistema.
  - Precondiciones: -
  - Poscondiciones: Hay una nueva categoría en el sistema.
  - Escenario básico: El administrador añade una nueva categoría al sistema.

- Añadir plantilla:
- Identificador: CU-002
- Actores: Administrador
- Descripción: El administrador borra una categoría del sistema.
- Precondiciones: Hay, al menos, una categoría.
- Poscondiciones: Hay una categoría menos en el sistema.
- Escenario básico: El administrador borra una categoría del sistema.

◆ Añadir plantilla:

- Identificador: CU-003
- Actores: Administrador
- Descripción: El administrador añade una nueva plantilla al sistema.
- Precondiciones: -
- Poscondiciones: Hay una nueva plantilla disponible en el sistema.
- Escenario básico: El administrador diseña, copia y agrega al sistema una nueva plantilla que puede ser usada como bloque, estando ubicado en una de las categorías existentes.

◆ Quitar plantilla:

- Identificador: CU-004
- Actores: Administrador
- Descripción: El administrador quita una plantilla del sistema

- Precondiciones: Existencia de al menos una plantilla en el sistema
- Poscondiciones: Una plantilla menos en el sistema
- Escenario básico: El adminstrador elige una plantilla del sistema y la elimina.

◆ Añadir bloque:

- Identificador: CU-005
- Actores: Usuario
- Descripción: El usuario añadir un bloque a su estructura
- Precondiciones: Deben existir una o más plantillas en el sistema
- Poscondiciones: El usuario dispone de un bloque más en la estructura de su proyecto.
- Escenario básico: El usuario añade un bloque a la estructura del proyecto con el que está trabajando actualmente.

◆ Quitar bloque:

- Identificador: CU-006
- Actores: Usuario
- Descripción: El usuario elimina un bloque de su esquema actual.
- Precondiciones: El usuario había añadido un bloque a la estructura de su proyecto.
- Poscondiciones: Existe un bloque menos en la estructura del proyecto del usuario.
- Escenario básico: El usuario dispone de un bloque menos en la estructura de su proyecto.

- Ordenar bloques:
  - Identificador: CU-007
  - Actores: Usuario
  - Descripción: El usuario reordena la estructura de su proyecto
  - Precondiciones: El proyecto del usuario contenía varios bloques
  - Poscondiciones: El orden de los bloques del proyecto ha cambiado
  - Escenario básico: El usuario cambia el orden de los bloques de su proyecto.

- Compartir diseño:
  - Identificador: CU-008
  - Actores: Usuario
  - Descripción: El usuario comparte la url de su proyecto
  - Precondiciones: El proyecto del usuario contiene uno o varios bloques
  - Poscondiciones: Otra persona distinta al usuario puede acceder a la estructura de bloques generada por él
  - Escenario básico: El usuario comparte la url generada al agregar bloques a su proyecto.

- Descargar diseño:
  - Identificador: CU-009
  - Actores: Usuario
  - Descripción: El usuario descarga el esquema actual de su proyecto

- Precondiciones: El usuario tiene uno o varios bloques en la estructura de su proyecto
- Poscondiciones: El usuario dispone de un archivo comprimido con la estructura generada
- Escenario básico: El usuario descarga el un archivo comprimido que contiene la estructura generada a través del sistema

3.3 ACTIVIDAD ASI 3: Identificación de subsistemas de análisis

3.3.1 Tarea 3.1: Determinación de subsistemas de análisis

Los subsistemas de análisis vienen determinados por los requisitos tratados anteriormente. El sistema se puede dividir en los siguientes subsistemas:

- Gestión de plantillas
- Gestión del esquema

3.3.2 Tarea 3.2: Integración de subsistemas de análisis

Al estar desarrollando una aplicación web podemos dividirla en dos partes a la hora de analizarla:

- **Frontend**: Toda la parte visual del proyecto. Páginas html, css, la interacción del usuario mediante arrastrar y soltar, ficheros javascript, etc.

- **Backend**: Sería la parte no visible, la parte del servidor web encargada de administrar las plantillas, generar el fichero comprimido, crear capturas de pantalla de las plantillas…

La gestión de plantillas estaría situado en la parte de backend y se encarga de las siguientes tareas:

- Estructura de ficheros que deben tener las plantillas
- Generar una captura de pantalla de la plantilla
- Generar archivo comprimido con la estructura de bloques seleccionada por el usuario

La gestión del esquema generado por el usuario se llevará a cabo en la parte frontend, en la parte visual de la aplicación, que será la encargada de facilitar una interfaz sencilla para que el usuario pueda interactuar con la aplicación. Dicha interfaz tendrá las siguientes partes:

- **Secciones**: Este será el primer nivel de organización de las plantillas de las que dispone el sistema.

- **Plantillas**: Cada sección dispone de varias plantillas, diferentes entre sí, que se podrán colocar en forma de bloques, una debajo de la otra.

- **Esquema de bloques** : Será la zona de la interfaz donde el usuario podrá colocar las partes que haya seleccionado para su proyecto.

El uso de Ruby on Rails obliga a articular todo lo anterior mediante una arquitectura de tipo MVC (modelo - vista – controlador), que es lo que nos servirá

para integrar las dos partes antes mencionadas, frontend y backend.

3.4 ACTIVIDAD ASI 4: Análisis de los casos de uso

A continuación, se presenta una tabla en la que quedan reflejadas las distintas clases que van a estar asociados a los casos de uso definidos con anterioridad:

Caso de Uso	Clase
Descargar diseño	Templates::Generator
Añadir plantillas	Templates::Thumbnailer
Quitar plantillas	Templates::Thumbnailer

Tabla 25: Clases por cada Caso de Uso

El resto de casos de uso no se incluyen en este apartado puesto que se hacen referencia a interacciones en la interfaz, implementadas con programación estructural en lenguaje javascript.

3.5 ACTIVIDAD ASI 5: Análisis de clases

En este apartado se trata la parte del proyecto que debe ser implementada usando **Orientación a Objetos** . Sin embargo no toda la implementación será llevada a cabo de esa forma y habrá varias tareas que son manuales como se ha comentado anteriormente.

La tarea manual será la de crear los templates (ficheros html, css). La forma de añadirlos al proyecto será colocándolos en el lugar correcto de la estructura de archivos, no será guardado en ninguna base de datos.

Además la parte de frontend en la que se ubica el código javascript hace uso de programación estructurada, por lo que será analizado en los apartados 3.6 y 3.7.

En la parte de backend tenemos dos clases principales encargadas de la generación del fichero comprimido y de la imágenes de los templates, además de las clases correspondientes a los controladores y modelos.

Las clases correspondientes a la tarea del fichero comprimido y de las imágenes en realidad serán utilizadas con objetos de servicio y tienen una gran importancia.

3.5.1 Servicios

Clase	Templates::Generator
Responsabilidades	Genera el fichero comprimido correspondiente a la estructura solicitada
Atributos	No.
Operaciones	run():bytes

Tabla 26: Clase Template::Generator

Clase	Templates::Thumbnailer
Responsabilidades	Genera las capturas de pantalla de los templates
Atributos	No.
Operaciones	run()

Tabla 27: Clase Template::Thumbnailer

3.5.2 Modelos

Clase	Template
Responsabilidades	Representa el concepto template y facilita la obtención de su información asociada (código html, etc)
Atributos	type:String, id:Integer
Operaciones	all_by_type():Array from_template_name():Template read_html():String read_style():String image_paths():Array thumbnail_path():String

Table 28: Clase Template

3.5.3 Controladores

Clase	ApplicationController
Responsabilidades	Clase base de la que heredan el resto de controladores
Atributos	No
Operaciones	Las propias del framework de rails que heran de ActionController::Base

Tabla 29: Clase ApplicationController

Clase	GeneratorController
Responsabilidades	Clase que recibe la petición http de generar un fichero comprimido y hace uso de la clase Templates::Thumbnailer para crearlo.
Atributos	No
Operaciones	show()

Tabla 30: Clase GeneratorController

Clase	TemplatesController
Responsabilidades	Clase principal que recibe la primera petición de navegación por la aplicación. Es la encarga de devolver el código de html de la aplicación.
Atributos	No
Operaciones	index()

Tabla 31: Clase TemplatesController

Clase	ThumbnailsController
Responsabilidades	Clase encargada de devolver la captura de pantalla correspondiente al template solicitado.
Atributos	No
Operaciones	show()

Tabla 32: Clase ThumbnailsController

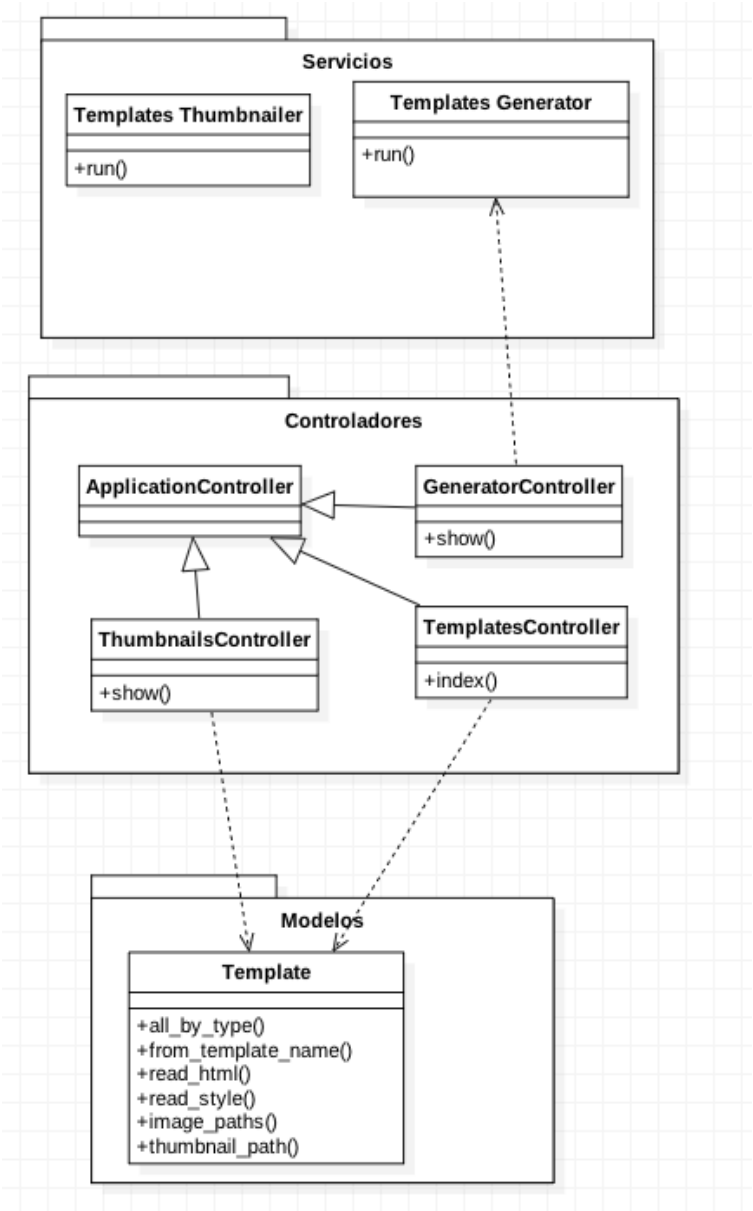


Ilustración 2: Diagrama de clases

3.6 ACTIVIDAD ASI 6: Elaboración del modelo de datos

La versión actual del proyecto no contempla ninguna integración con una base de datos.

3.6.1 ACTIVIDAD ASI 7: Elaboración del modelo de procesos

Esta actividad solo tiene sentido en caso de realizar una Análisis correspondiente a Programación estructurada. En este caso será necesario tanto para especificar las tareas que se vayan a ejecutar mediante un script como la parte del frontend.

La parte del frontend utilizará código javascript, que si bien es orientado a objetos también permite programación funcional. Esto último unido a la utilización de la librería jQuery hace que la organización sea estructurada.

Existen diferentes opciones para que la interfaz fuese altamente modular y que siga las reglas de POO (Programación orientada a objetos), como por ejemplo los frameworks de AngularJS [4], Backbone.js [5] o Ember.js [6]. Debido a que la interfaz no tiene una complejidad grande en cuanto a su organización, sino en cuanto a la interacción del usuario, la utilización de dichos frameworks no aporta nada interesante.

4 https://angularjs.org/
5 http://backbonejs.org/
6 http://emberjs.com/

3.6.2 Tarea ASI 7.1: Obtención del modelo de procesos del sistema

A continuación se mostrarán los involucrados en la parte de frontend:

Tarea para generar las capturas de pantalla de las plantillas:

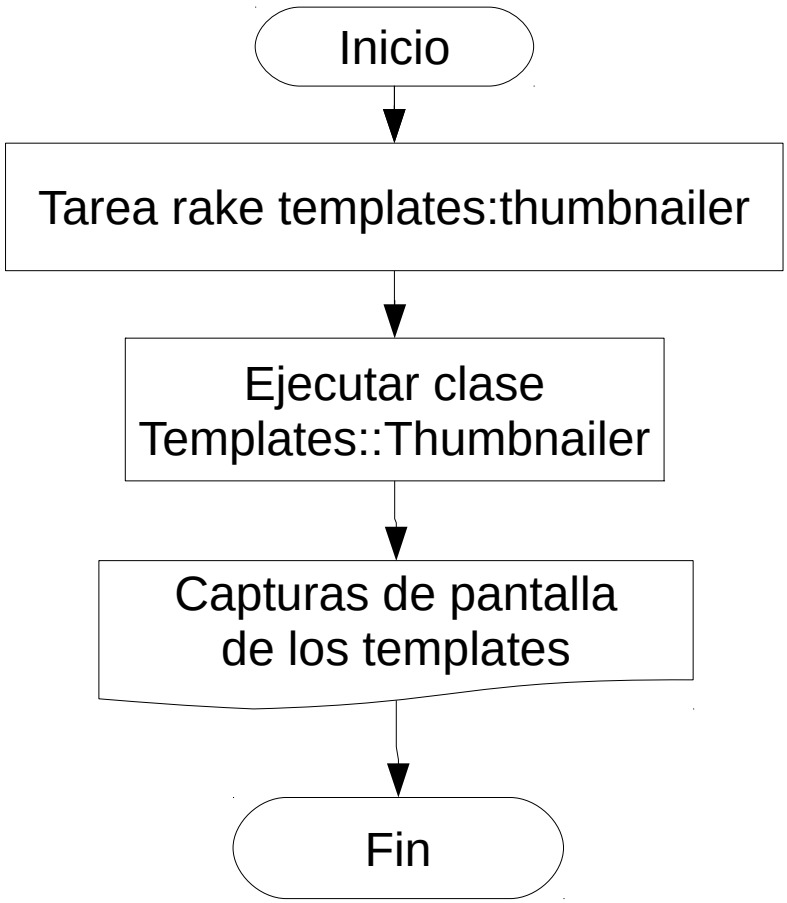


Ilustración 3: Generación de capturas de pantalla de los templates

Interacción para agregar bloques al esquema:

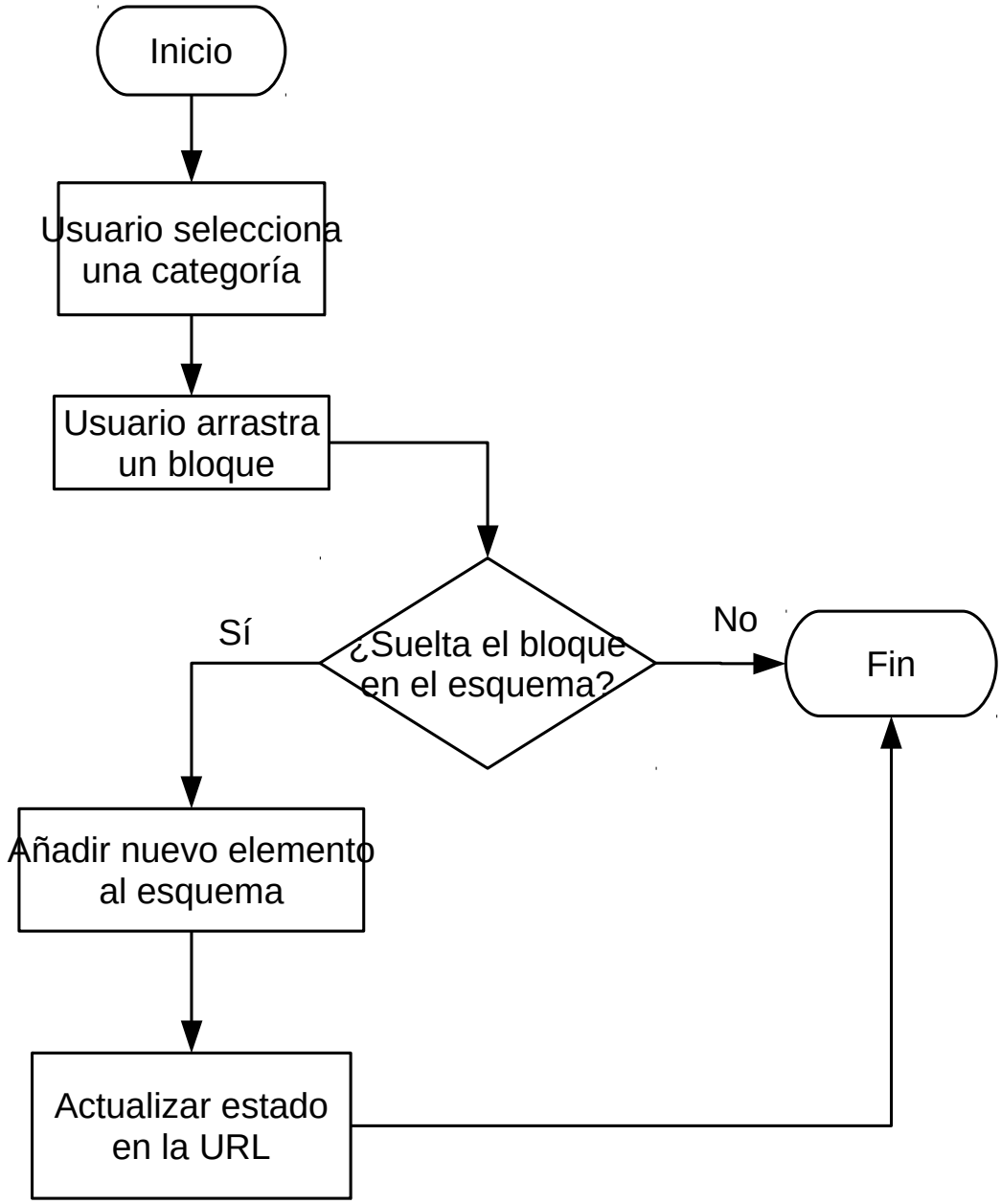


Ilustración 4: Agregar bloques al esquema

3.7 ACTIVIDAD ASI 8: Definición de interfaces de usuario

Las interfaces de usuario son el vehículo a través del cual el sistema y el usuario interaccionan entre sí. Por ello, es de vital importancia que las mismas se adecuen a las necesidades de éste, y permitan un óptimo desempeño de sus tareas.

3.7.1 Tarea ASI 8.1: Especificación de Principios Generales de la Interfaz

La interfaz debe register por los principios de usabilidad y claridad en la información. Los elementos meramente decorativos que entorpezcan la navegación e interacción con el sistema deben ser eliminados. La simplicidad y el uso de textos que sirvan de tutorial deben ser una característica indispensable.

Los mensajes de error deben proporcionar suficiente información al usuario para que pueda actuar por su propia cuenta, o informarle de que debe pedir ayuda al administrador de la página web en caso necesario.

3.7.2 Tarea ASI 8.2: Identificación de Perfiles y Diálogos

Desde el punto de vista de la interfaz existe un único perfil, que es el del usuario. Accederá a la aplicación desde un ordenador de escritorio.

3.7.3 Tarea ASI 8.3: Especificación de Formatos Individuales de la Interfaz de Pantalla

A continuación se presentarán los esquemas de las pantallas principales con las que puede interactuar el usuario.

* Bienvenida:

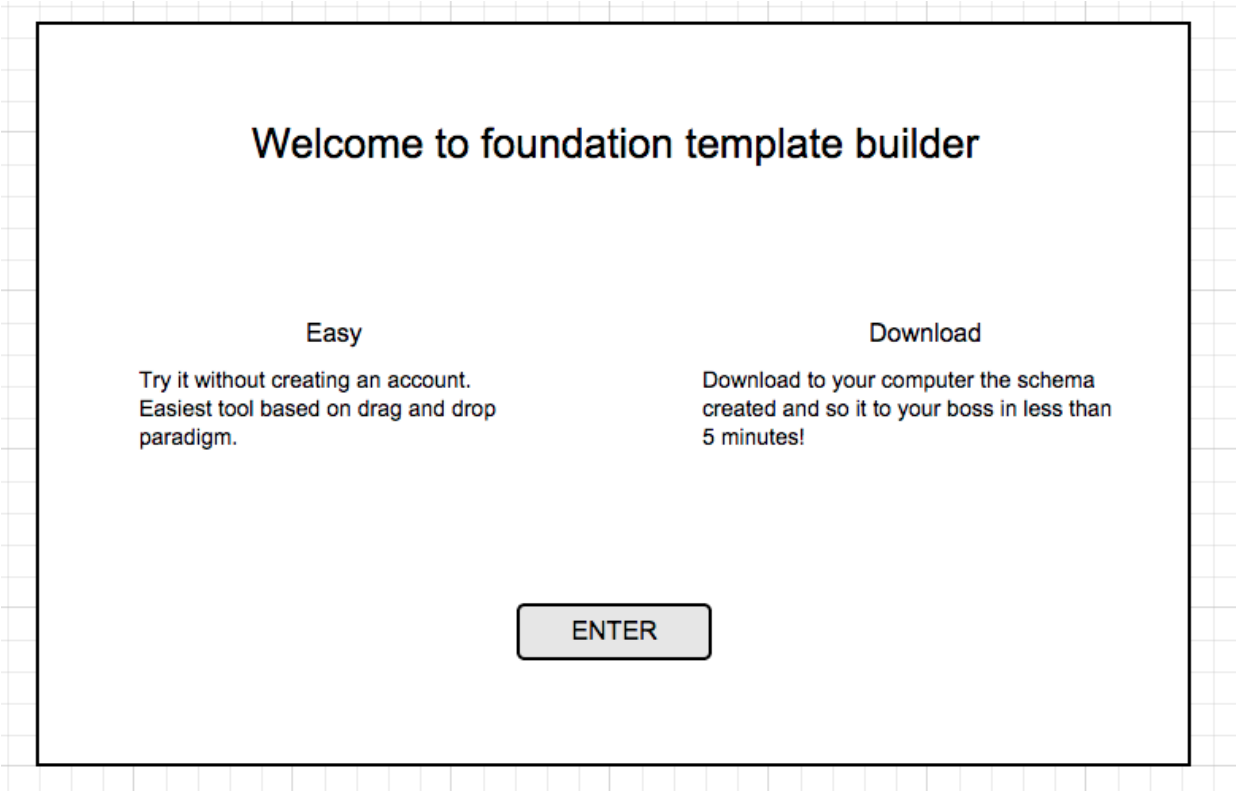


Ilustración 5: Pantalla de bienvenida

* Generador de esquemas:

Ilustración 6: Pantalla del esquema de bloques

* Descarga del fichero comprimido (se puede apreciar el cambio de la url guardando el esquema actual):

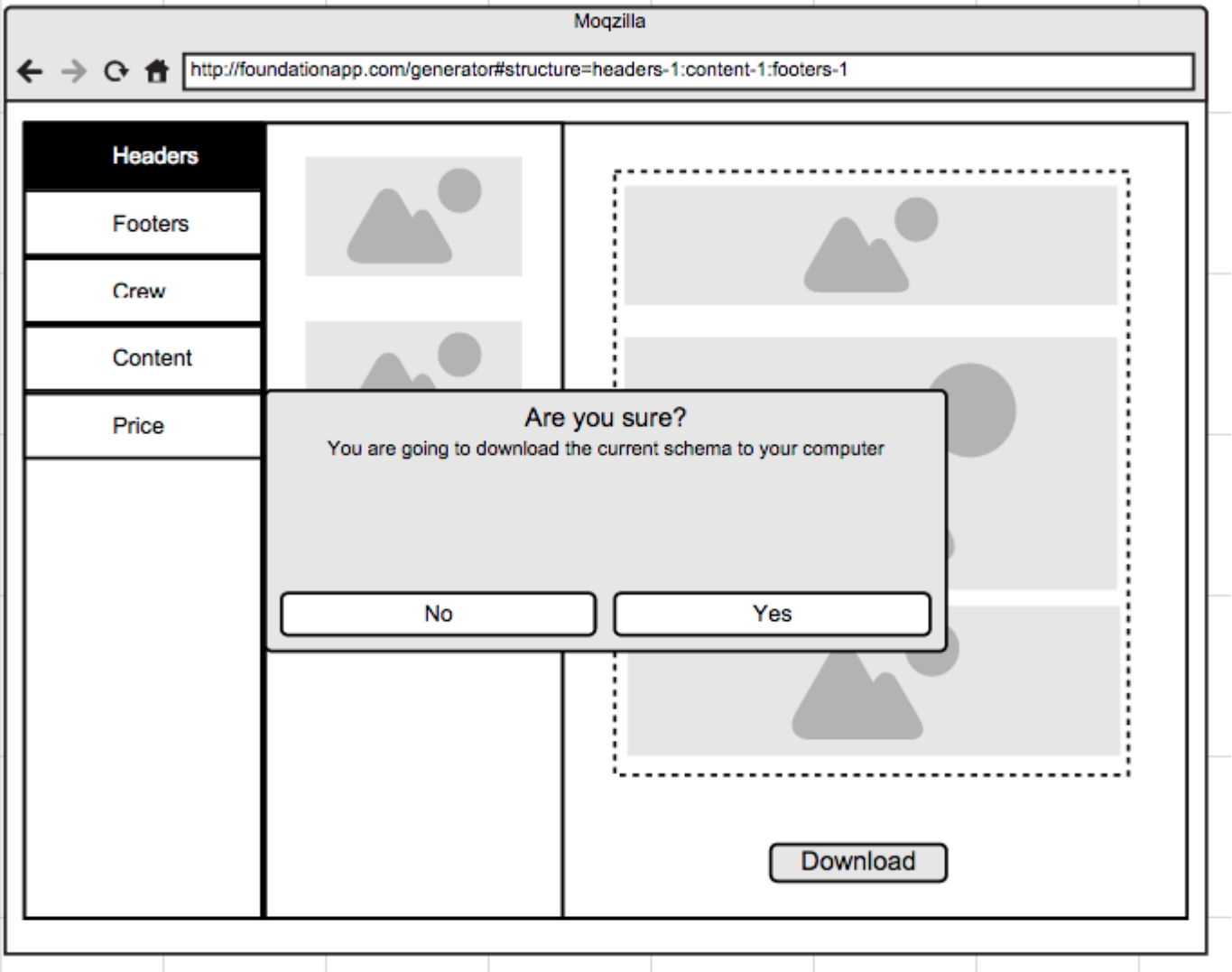


Illustration 7: Descarga del esquema

3.7.4 Tarea ASI 8.5: Especificación de Formatos de Impresión

Dado que no se va a imprimir el contenido, el formato de impresión es el mismo que el visual.

3.8 ACTIVIDAD ASI 9: ANÁLISIS DE CONSISTENCIA Y ESPECIFICACIÓN DE REQUISITOS

3.8.1 Tareas ASI 9.1 y ASI 9.2: Verificación y Análisis de Consistencia entre Modelos

Se muestra a continuación la matriz de trazabilidad entre los requisitos software y los requisitos de usuario:

Requisito de Software	**Requisito de Usuario**				
	RU-C001	RU-C002	RU-C003	RU-C004	RU-C005
RS-F001	x				
RS-F002	x				
RS-F003	x				
RS-F004		x			
RS-F005		x			
RS-F006		x			
RS-F007			x		
RS-F008			x		
RS-F009			x		
RS-F010				x	
RS-F011				x	
RS-F012					x

4. Diseño del sistema de información (DSI)

El presente documento es una versión adaptada del Diseño del Sistema de Información (DSI) de Métrica v3. Algunos de sus apartados han sido estudiados de manera exhaustiva en el documento de Análisis. Por ello, aquí no se desarrollarán.

4.1 ACTIVIDAD DSI 1: DEFINICIÓN DE LA ARQUITECTURA DEL SISTEMA

El framework Ruby on Rails hace uso del patrón de arquitectura MVC (Modelo – Vista – Controlador), patrón en el que se basa igualmente este proyecto. Dicho patrón es altamente utilizado en el entorno de las aplicaciones web.

En esta arquitectura el sistema se organiza separando los datos y la lógica de negocio por un lado de la interfaz y del módulo encargado de gestionar los eventos y las comunicaciones.

El Modelo es el encargado de albergar las estructuras de datos y la comunicación con los sistemas gestores de datos. En este proyecto no hay comunicación con una base de datos, pero si hay estructuras de datos que facilitan la manipulación de las plantillas.

El Controlador viene dado por el módulo ActionController de Ruby on Rails (entre todas cosas), que facilita enormemente la comunicación de la aplicación con Internet.

La Vista se utiliza la tecnología erb, y ActionView, encargados de la

generación de páginas html y proveer muchas facilidades a la hora de utilizar código javascript y css.

En Ruby on Rails no se utilizan clases para representar las vistas, sino que directamente se escribe el código que generará la respuesta. Por eso en el diagrama solo se incluyen los nombres de ficheros:

Vistas:

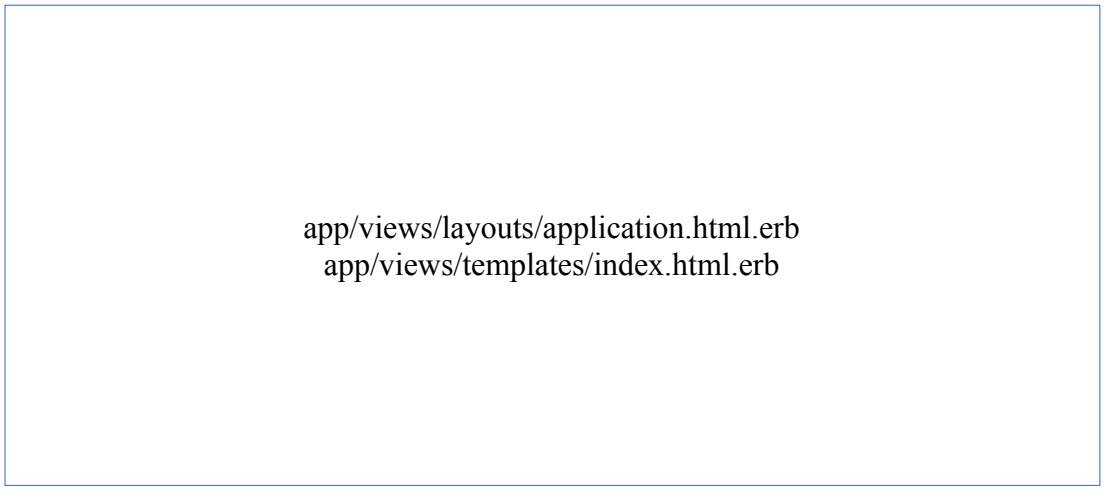


Ilustración 8: Arquitectura del sistema para Vista

Controladores:

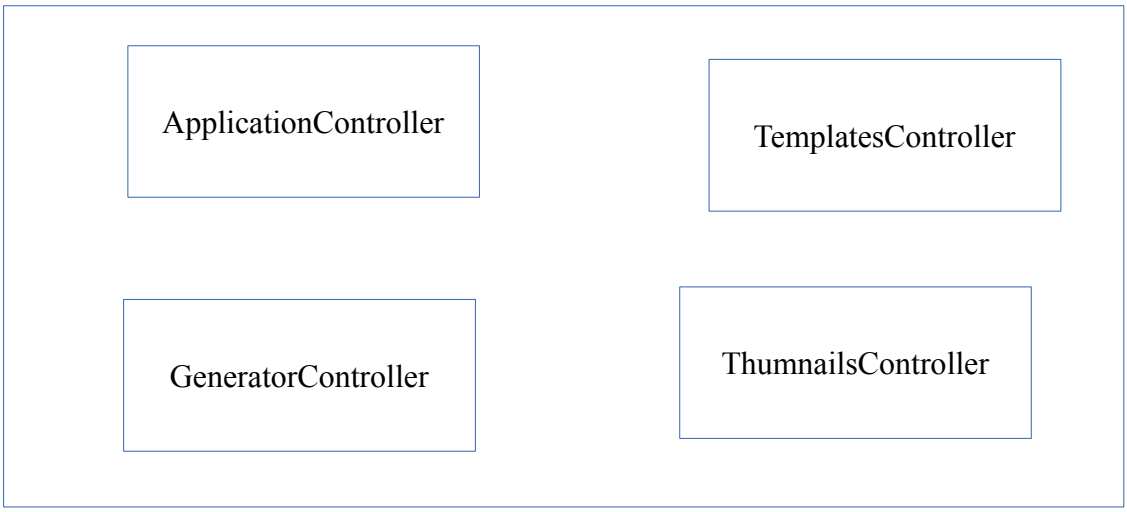


Ilustración 9: Arquitectura del sistema para Controlador

Modelos:

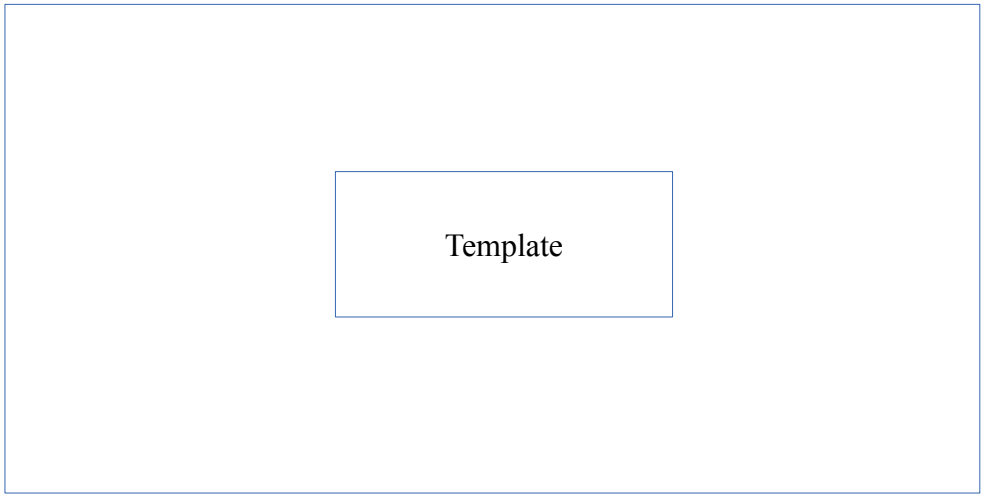


Ilustración 10: Arquitectura del sistema para Modelo

Esquema MVP.

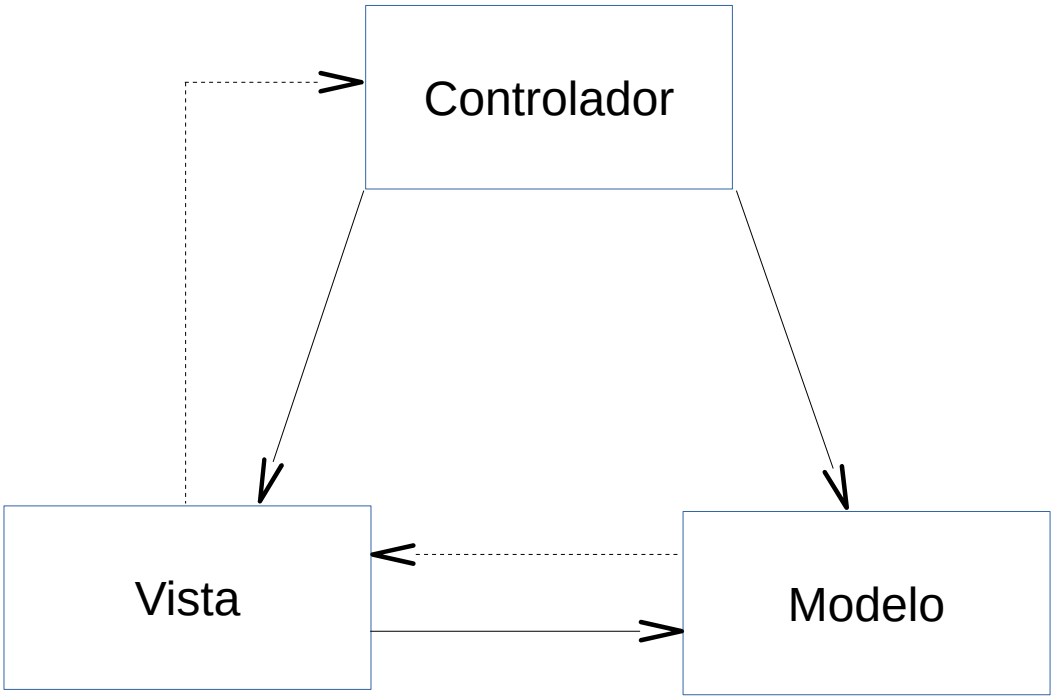


Ilustración 11: Esquema de la arquitectura MVP

4.1.1 Tarea DSI 3.3: Revisión de la Interfaz de Usuario

A continuación se muestran las diferentes pantallas de la aplicación una vez realizada su implementación.

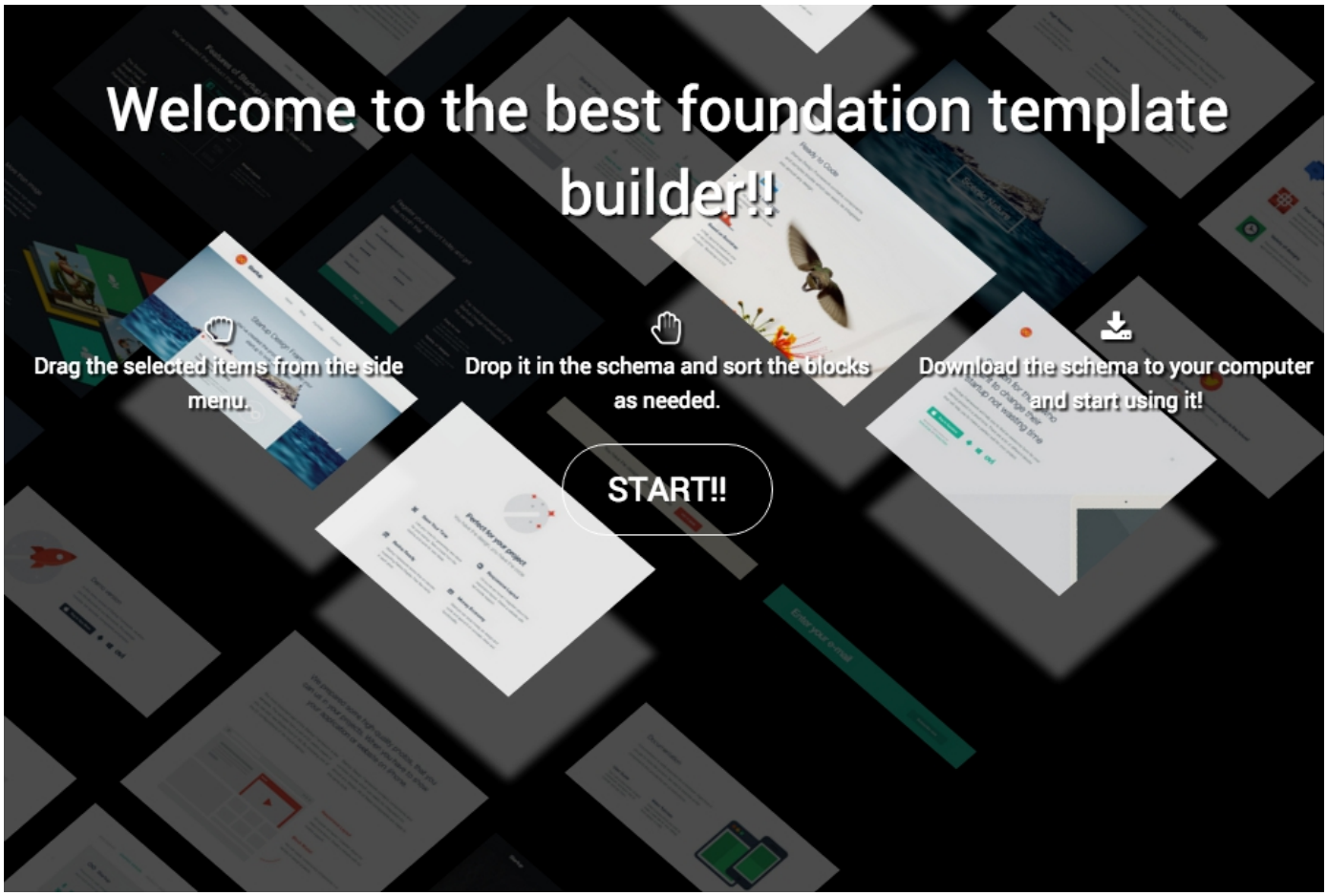


Ilustración 12: Pantalla de bienvenida

Ilustración 13: Pantalla de creación de esquemas

4.2 ACTIVIDAD DSI 6: DISEÑO FÍSICO DE DATOS

No se hay ninguna integración con una base de datos.

5. Manual de implantación

Aunque ya se ha mencionado que la aplicación será desplegada en los servidores de heroku, en este apartado se explicará como debería hacerse la instalación en caso de desplegar la aplicación en un servidor con sistema operativo Ubuntu.

Se asumirá siempre que se la posibilidad de ejecutar comandos como administrador, mediante usuario root o con el uso del comando sudo.

5.1 Instalación del interprete de ruby

Para este proyecto se ha usado la versión 2.2 de ruby por lo que será necesario instalar dicha versión. La última versión de ruby ofrecida por los repositorios de ubuntu es la 2.1, por lo que es necesario agregar una fuente externa.

Usaremos el repositorio de brightbox [7] que posee un paquete con la versión 2.2, también instalaremos los paquetes de desarrollo pertinentes:

```
sudo apt-add-repository ppa:brightbox/ruby-ng
sudo apt-get update
sudo apt-get install ruby2.2
sudo apt-get install ruby2.2-dev build-essential libxml2-dev zlib1g-dev
```

Además necesitaremos los paquetes de desarrollo de ruby y algunos paquetes de necesarios para compilar las dependencias.

7 https://www.brightbox.com/blog/2015/01/05/ruby-2-2-0-packages-for-ubuntu/

Además para poder instalar las gemas de ruby sin necesitar persmisos de administrador cambiaremos la variable de entorno GEM_HOME, que especifica donde deben instalarse.

```
echo "export GEM_HOME=~/.gem/ruby2.2" >> ~/.bashrc
echo "PATH=$PATH:~/.gem/ruby2.2" >> ~/.bashrc
source ~/.bashrc
```

Al ejecutar el comando “gem env” debería mostrarse lo siguiente:

```
INSTALLATION DIRECTORY: /home/ubuntu/.gem/ruby2.2
```

En caso contrario habría que abrir una nueva consola.

Desde el ordenador de la persona que está haciendo el depliegue hay que ejecutar el comando para desplegar el código en la máquina de producción (la máquina de producción debe estar especificada en el fichero config/deploy/production.rb):

```
cap production deploy
```

Generamos un valor aleatorio para las variables de sesión:

```
rake secret
```

Instalamos apache y el módulo passenger en el servidor de producción:

```
sudo aptitude install apache2 libapache2-mod-passenger
```

Creamos el fichero /etc/apache2/sites-available/foundationdd.conf con el siguiente contenido, usando la clave secreta generada previamente:

```
<VirtualHost *:80>
 ServerName <dominio>
 DocumentRoot /var/www/foundationdd/current/public
 RailsEnv production
 SetEnv SECRET_KEY_BASE <SECRET_KEY>

 ErrorLog ${APACHE_LOG_DIR}/error.log
 CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Y reiniciamos el servicio apache:

```
sudo service apache2 restart
```

Ahora ya dispondremos de nuestra aplicación en el servidor de producción.

6. Implantación y aceptación del sistema

6.1 Tarea IAS 1: ESTABLECIMIENTO DEL PLAN DE IMPLANTACIÓN

Como anteriormente se ha comentado este proyecto utiliza los servidores de heroku para desplegar las aplicaciones. Esto implica el conocimiento necesario para poder desplegar el sistema es mínimo, no es necesario conocer nada acerca de la administración y configuración de servidores.

Sencillamente hay que tener instalado la herramienta git [8] que suele venir preinstalada en linux y en mac. En el momento en el que se haga push al servidor de heroku se envían los datos del proyecto y el propio servidor se encarga de instalar las dependencias y generar los archivos necesarios.

Si tuviésemos como requisito la utilización de base de datos también sería una tarea trivial instalar y administrar una (casi todas las tareas son automáticas), pero como se ha comentado este proyecto hace uso del sistema de ficheros para almacenar las plantillas.

6.1.1 Tarea IAS 1.1: Definición del Plan de Implantación

Dado que implantar el sistema es una tarea muy sencilla solo se necesita una persona para llevarlo realizarlo. Esta persona puede ser el propio programador de la aplicación.

Para poder hacer push a heroku primero hay que configurar el servidor al que se quiere enviar los datos:

8 https://git-scm.com/

```
git remote add heroku https://git.heroku.com/foundationdd-rails.git
```

Una vez que hemos configurado la dirección del servidor debemos hacer lo siguiente:

```
git push heroku master
```

Y automáticamente enviará la última versión de la aplicación al servidor y realizará las tareas de despliegue. Si nos queremos bajar los últimos cambios:

```
git pull heroku master
```

Se puede encontrar más información en la documentación de heroku[9].

9 https://devcenter.heroku.com/

7. Conclusiones y líneas futuras

El presente proyecto surgió como una propuesta que le hice y que fue bien acogida. Son varios los sitios en internet en los que se pueden descargar plantillas de html ya confeccionadas (por ejemplo wix[10]). También existen otros sitios en los que se puede generar un documento arrastrando y soltando bloques, como layoutit![11].

7.1 Conclusiones

La elección de las diferentes tecnologías ha sido clave en la implementación de la aplicación. Gracias a la filosofía “convención sobre configuración” que sigue Ruby on Rails es posible crear aplicaciones web con menos recursos y menos código. Por otra parte es necesario memorizar gran cantidad de información y para poder hacer uso de su API con soltura.

También es importante destacar la facilidad con la que se pueden desplegar aplicaciones en los servidores de heroku, con un simple “git push” podemos desplegar una nueva versión de nuestra aplicación, ahorrando costes y tiempo.

Creo que es interesante hacer notar que una de las partes que fue más complicada de implementar fue la parte del drag&drop. Si bien se hace uso de la librería jquery-ui que ha facilitado la tarea, sigue siendo necesario entender en profundidad como funciona las estructuras de nodos de html y los eventos javascript que se generan.

Uniendo lo anterior a la idea de utilizar frameworks y plataformas que agilizan el desarrollo de una aplicación, tenemos una explicación de porqué hoy en día es tan fácil iniciar una startup con los recursos mínimos, sin tener que esperar a meses de desarrollo y empezar a monetizar lo antes posible.

10 http://es.wix.com/
11 http://www.layoutit.com/es

7.2 Líneas futuras

El sistema desarrollado es lo suficientemente general y fácilmente adaptable como para abrir nuevas ampliaciones, de entre las cuales pueden mencionarse las siguientes:

- Esquemas pregenerados: Sería interesante tener esquemas ya generados con los que empezar a realizar cambios, en vez de empezar desde 0.

- Plantillas colaborativas: Hacer posible que cualquier pueda crear sus propias plantillas y compartirlas con todo el mundo a través de la aplicación. Esto implica una cantidad importante de desarrollo puesto que habría que tener en cuenta cosas como integración con base de datos, restricciones de seguridad, etc.

- Personalización del esquema: Añadir la posibilidad de configurar el esquema cambiando colores, tipografía, imágenes, tamaños de fuente, etc. Otra opción similar también podría ser elegir entre diferentes frameworks (actualmente solo es posible generar el esquema usando el framework foundation), como bootstrap.