



# ESCUELA SUPERIOR DE INGENIERÍA

## Ingeniería Informática

SiteUp: plataforma para la vigilancia de la disponibilidad de servicios de internet

José Tomás Tocino García

Cádiz, 21 de abril de 2014





# ESCUELA SUPERIOR DE INGENIERÍA

## Ingeniería Informática

**SiteUp: plataforma para la vigilancia de la disponibilidad de servicios de internet**

DEPARTAMENTO: Lenguajes y Sistemas Informáticos.  
DIRECTOR DEL PROYECTO: Ivan Ruiz Rube  
AUTOR DEL PROYECTO: José Tomás Tocino García.

Cádiz, 21 de abril de 2014

Fdo.: José Tomás Tocino García



Este documento se halla bajo la licencia FDL (Free Documentation License). Según estipula la licencia, se muestra aquí el aviso de copyright. Se ha usado la versión inglesa de la licencia, al ser la única reconocida oficialmente por la FSF (Free Software Foundation).

Copyright © 2014 José Tomás Tocino García.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".



## Agradecimientos

Quiero agradecer y dedicar la presente memoria y, por extensión, el proyecto Si-teUp al completo:



## Resumen

Aquí va el resumen no superior a 500 palabras.

**Palabras clave:** Lorem, ipsum, dillum, sit, amet



# Índice general

Índice general	11
Índice de figuras	15
Índice de cuadros	17
<b>1. Introducción</b>	<b>19</b>
1.1. Contexto social . . . . .	19
1.2. Motivación . . . . .	19
1.3. Objetivos . . . . .	20
1.3.1. Funcionales . . . . .	20
1.3.2. Transversales . . . . .	21
1.4. Estado del arte . . . . .	21
1.5. Alcance . . . . .	22
1.5.1. Limitaciones del proyecto . . . . .	22
1.5.2. Licencia . . . . .	23
1.6. Estructura del documento . . . . .	24
1.7. Acrónimos . . . . .	24
<b>2. Planificación</b>	<b>27</b>
2.1. Metodología de desarrollo . . . . .	27
2.1.1. Primera iteración: adquisición de conocimientos y elicita- ción de requisitos . . . . .	27
2.1.2. Segunda iteración: desarrollo de módulo de herramientas básicas de chequeo . . . . .	28
2.1.3. Tercera iteración: inicio de proyecto Django y CRUD básico	28
2.1.4. Cuarta iteración: integración del motor de tareas asíncronas	29
2.1.5. Quinta iteración: desarrollo de la aplicación Android . . . . .	29
2.2. Planificación temporal . . . . .	29
2.3. Organización . . . . .	29
2.4. Recursos inventariables . . . . .	29
2.5. Costes . . . . .	31
2.6. Riesgos . . . . .	32
2.6.1. Compromiso de la plataforma de despliegue . . . . .	32
2.6.2. Limitación de recursos . . . . .	32
2.6.3. Rechazo de los servicios vigilados . . . . .	32

<b>3. Requisitos del sistema</b>	<b>35</b>
3.1. Objetivos del sistema . . . . .	35
3.2. Catálogo de requisitos . . . . .	36
3.2.1. Requisitos funcionales . . . . .	36
3.2.2. Requisitos de información . . . . .	39
3.2.3. Requisitos no funcionales . . . . .	41
3.3. Alternativas de solución . . . . .	43
3.3.1. Frameworks para el desarrollo de la plataforma web . . . . .	43
3.3.2. Sistemas de gestión de bases de datos . . . . .	43
3.3.3. Sistemas para la ejecución asíncrona de tareas . . . . .	44
3.3.4. Plataformas para el desarrollo de la aplicación móvil . . . . .	44
<b>4. Análisis</b>	<b>47</b>
4.1. Modelo conceptual . . . . .	47
4.1.1. Usuario . . . . .	47
4.1.2. Grupo de chequeos . . . . .	49
4.1.3. Chequeo . . . . .	49
4.1.4. Registro de chequeo . . . . .	49
4.1.5. Chequeo - Ping . . . . .	49
4.1.6. Chequeo - Port . . . . .	50
4.1.7. Chequeo - DNS . . . . .	50
4.1.8. Chequeo - HTTP . . . . .	50
4.2. Modelo de casos de uso . . . . .	50
4.2.1. Actores . . . . .	50
4.2.2. Subsistema de gestión de usuarios . . . . .	51
4.2.3. Subsistema de gestión de grupos de chequeos . . . . .	53
4.2.4. Subsistema de gestión de chequeos . . . . .	55
4.3. Modelo de comportamiento . . . . .	59
4.4. Modelo de interfaz de usuario . . . . .	59
4.4.1. Modelo de navegación . . . . .	59
4.4.2. Prototipos visuales de la interfaz web . . . . .	59
4.4.3. Prototipos visuales de la aplicación móvil . . . . .	61
<b>5. Diseño</b>	<b>69</b>
5.1. Arquitectura del sistema . . . . .	69
5.1.1. Arquitectura física . . . . .	69
5.1.2. Arquitectura lógica . . . . .	70
5.2. Diseño físico de datos . . . . .	76
5.2.1. User . . . . .	76
5.2.2. UserExtra . . . . .	78
5.2.3. CheckGroup . . . . .	78
5.2.4. BaseCheck . . . . .	78
5.2.5. DnsCheck . . . . .	79
5.2.6. HttpCheck . . . . .	79
5.2.7. PortCheck . . . . .	79
5.2.8. PingCheck . . . . .	80
5.2.9. CheckLog . . . . .	80

5.2.10. CheckStatus . . . . .	80
5.2.11. Session . . . . .	81
5.2.12. MigrationHistory . . . . .	81
5.2.13. ContentType . . . . .	81
5.3. Diseño de la imagen corporativa . . . . .	82
5.3.1. Diseño del logotipo . . . . .	82
5.3.2. Elección de la gama cromática . . . . .	83
5.3.3. Elección de la tipografía . . . . .	83
5.4. Diseño de la interfaz de usuario de la plataforma web . . . . .	84
5.4.1. Pantalla de inicio . . . . .	84
5.4.2. Pantalla de inicio de sesión . . . . .	86
5.4.3. Pantalla de recuperación de contraseña . . . . .	86
5.4.4. Pantalla de registro de usuario . . . . .	87
5.4.5. Pantalla de lista de chequeos . . . . .	87
5.4.6. Pantalla de creación de grupo . . . . .	89
5.4.7. Pantallas de creación de chequeo . . . . .	90
5.4.8. Pantalla genérica de borrado . . . . .	90
5.4.9. Pantalla del perfil de usuario . . . . .	90
5.5. Diseño de la interfaz de usuario de la aplicación móvil . . . . .	92
5.5.1. Aspecto desde el lanzador . . . . .	92
5.5.2. Pantalla de carga . . . . .	93
5.5.3. Pantalla de inicio de sesión . . . . .	93
5.5.4. Pantalla de listado de chequeos . . . . .	93
5.5.5. Notificación . . . . .	96
<b>6. Implementación</b>	<b>97</b>
6.1. Entorno de construcción . . . . .	97
6.1.1. Aplicación web . . . . .	97
6.1.2. Aplicación móvil . . . . .	103
6.1.3. Herramientas de desarrollo . . . . .	104
6.1.4. Servicio de notificaciones . . . . .	104
6.2. Organización del código fuente . . . . .	104
6.2.1. Plataforma web . . . . .	105
6.2.2. Aplicación móvil . . . . .	107
6.3. Detalles de implementación . . . . .	108
6.3.1. Uso de herencia de modelos . . . . .	108
6.3.2. Workflow de front-end . . . . .	109
6.4. Gestión de la base de datos . . . . .	113
6.4.1. Cambios en la estructura . . . . .	115
<b>7. Pruebas</b>	<b>117</b>
7.1. Estrategia . . . . .	117
7.2. Entorno de pruebas . . . . .	117
7.3. Roles . . . . .	118
7.3.1. Desarrollador principal . . . . .	118
7.3.2. Probadores externos . . . . .	118
7.4. Niveles de pruebas . . . . .	118

7.4.1. Pruebas unitarias . . . . .	118
7.4.2. Pruebas de integración . . . . .	119
7.4.3. Pruebas de sistema . . . . .	119
7.4.4. Pruebas de aceptación . . . . .	119
7.5. Implementación de pruebas . . . . .	120
<b>8. Conclusiones</b>	<b>121</b>
8.1. Objetivos cumplidos . . . . .	121
8.2. Conclusiones personales . . . . .	122
8.2.1. Lecciones aprendidas . . . . .	122
8.3. Trabajo futuro . . . . .	123
8.3.1. SiteUp Probes . . . . .	123
8.3.2. Monetización . . . . .	123
8.3.3. Ampliación de los tipos de cheques . . . . .	124
8.3.4. Ampliación de tipos de notificación . . . . .	124
<b>A. Manual de instalación de la plataforma web</b>	<b>125</b>
A.1. Instalación inicial . . . . .	125
A.1.1. Descarga de código . . . . .	125
A.1.2. Entorno virtual y dependencias . . . . .	126
A.1.3. Creación de la base de datos . . . . .	126
A.1.4. Instalación del servidor . . . . .	126
A.1.5. Instalación del bróker de mensajes . . . . .	127
A.1.6. Ejecución de los servicios . . . . .	127
A.1.7. Configuración del servidor frontal . . . . .	128
A.1.8. Lanzamiento final . . . . .	128
A.2. Mejora del rendimiento . . . . .	129
<b>B. Manual de usuario</b>	<b>131</b>
<b>C. GNU Free Documentation License</b>	<b>133</b>
1. APPLICABILITY AND DEFINITIONS . . . . .	133
2. VERBATIM COPYING . . . . .	135
3. COPYING IN QUANTITY . . . . .	135
4. MODIFICATIONS . . . . .	136
5. COMBINING DOCUMENTS . . . . .	138
6. COLLECTIONS OF DOCUMENTS . . . . .	138
7. AGGREGATION WITH INDEPENDENT WORKS . . . . .	138
8. TRANSLATION . . . . .	139
9. TERMINATION . . . . .	139
10. FUTURE REVISIONS OF THIS LICENSE . . . . .	140
11. RELICENSING . . . . .	140
ADDENDUM: How to use this License for your documents . . . . .	141
<b>Bibliografía y referencias</b>	<b>143</b>

# Índice de figuras

2.1. Diagrama de Gantt . . . . .	30
4.1. Modelo conceptual . . . . .	48
4.2. Diagrama de actores . . . . .	50
4.3. Casos de uso del subsistema de gestión de usuarios . . . . .	51
4.4. Casos de uso del subsistema de gestión de grupos de chequeos . . . . .	53
4.5. Casos de uso del subsistema de gestión de chequeos . . . . .	56
4.6. Modelo de navegación de la aplicación web . . . . .	60
4.7. Modelo de navegación de la aplicación Android . . . . .	61
4.8. Home . . . . .	62
4.9. Login . . . . .	62
4.10. Recuperación de contraseña . . . . .	63
4.11. Registro de usuario . . . . .	63
4.12. Dashboard . . . . .	64
4.13. Detalle de chequeo . . . . .	64
4.14. Creación de grupo de chequeos . . . . .	65
4.15. Formulario genérico de creación de chequeo . . . . .	65
4.16. Pantalla de carga . . . . .	66
4.17. Pantalla de login . . . . .	66
4.18. Pantalla de listado de chequeos . . . . .	67
4.19. Pantalla de notificación . . . . .	67
5.1. Arquitectura lógica del sistema web . . . . .	71
5.2. Arquitectura lógica de la aplicación Android . . . . .	75
5.3. Diseño de la base de datos . . . . .	77
5.4. Logotipo de SiteUp . . . . .	83
5.5. Detalle de la paleta de colores de SiteUp . . . . .	83
5.6. Muestra de Open Sans en diferentes pesos . . . . .	84
5.7. Pantalla de inicio . . . . .	85
5.8. Pantalla de inicio de sesión . . . . .	86
5.9. Pantalla de recuperación de contraseña . . . . .	87
5.10. Pantalla de registro de usuario . . . . .	87
5.11. Pantalla de lista de chequeos . . . . .	88
5.12. Pantalla de creación de grupo de chequeos . . . . .	89
5.13. Pantalla de elección de tipo de chequeo . . . . .	90
5.14. Pantalla de introducción de datos del chequeo . . . . .	91

5.15. Pantalla genérica de borrado . . . . .	91
5.16. Pantalla del perfil de usuario . . . . .	92
5.17. Icono de SiteUp en el lanzador . . . . .	93
5.18. Pantalla de carga de la aplicación móvil . . . . .	94
5.19. Pantalla de inicio de sesión de la aplicación móvil . . . . .	94
5.20. Pantalla de chequeos de la aplicación móvil . . . . .	95
5.21. Aspecto de las notificaciones . . . . .	95
6.1. Diagrama de flujo de trabajo del front-end . . . . .	110
6.2. Diagrama ER de ejemplo 1 . . . . .	113

# Índice de cuadros

3.1. OBJ-1 . . . . .	35
3.2. OBJ-2 . . . . .	36
3.3. RQF-1 . . . . .	36
3.4. RQF-2 . . . . .	36
3.5. RQF-3 . . . . .	36
3.6. RQF-4 . . . . .	37
3.7. RQF-5 . . . . .	37
3.8. RQF-6 . . . . .	37
3.9. RQF-7 . . . . .	37
3.10. RQF-8 . . . . .	37
3.11. RQF-9 . . . . .	38
3.12. RQF-11 . . . . .	38
3.13. RQF-11 . . . . .	38
3.14. RQF-12 . . . . .	38
3.15. RQF-13 . . . . .	38
3.16. RQF-14 . . . . .	39
3.17. IRQ-1 . . . . .	39
3.18. IRQ-2 . . . . .	39
3.19. IRQ-3 . . . . .	40
3.20. IRQ-4 . . . . .	40
3.21. IRQ-5 . . . . .	40
3.22. IRQ-6 . . . . .	41
3.23. IRQ-7 . . . . .	41
3.24. IRQ-8 . . . . .	41
3.25. NRQ-1 . . . . .	41
3.26. NRQ-2 . . . . .	42
3.27. NRQ-3 . . . . .	42
3.28. NRQ-4 . . . . .	42
3.29. NRQ-5 . . . . .	42
3.30. Cuota de mercado de plataformas móviles en enero de 2014 . . . . .	46



# Capítulo 1

## Introducción

### 1.1. Contexto social

Las tecnologías de la información en general e Internet en particular son ya parte integral de la sociedad. Casi todos los ámbitos de la vida, desde las interacciones sociales hasta la búsqueda de empleo, cuentan ya con su reflejo en las tecnologías de la información, a menudo mediante el uso de servicios web a través de Internet.

No solo los aspectos tradicionales de la sociedad tienen su presencia en las redes, también han surgido nuevos modelos empresariales propios de Internet que han crecido de manera importante y se han situado a niveles comparables a los de las empresas tradicionales. Empresas puramente digitales como Facebook o Twitter ya cotizan en bolsa y realizan operaciones bursátiles del orden de miles de millones de dólares [16].

Se pone pues de manifiesto la importancia de la fiabilidad de los servicios e infraestructuras de los que dependen estos nuevos modelos de negocio. El *uptime* – un término inglés que describe el porcentaje de tiempo que un servicio se mantiene disponible – debe ser siempre cercano al 100%, dado que en caso contrario los potenciales usuarios del servicio se encontrarán con que no pueden acceder a él, dando lugar incluso a pérdidas económicas. Es el caso de Amazon, que llegó a perder 4.8 millones de dólares al sufrir un fallo que dejó inaccesible su web durante 40 minutos [2].

De todo lo expuesto se extrae la necesidad de contar con sistemas para monitorizar la disponibilidad de estos servicios y, en caso necesario, actuar de manera que puedan subsanarse las causas de los problemas.

### 1.2. Motivación

La idea de desarrollar este proyecto surge a raíz de una necesidad personal del desarrollador del proyecto.

En octubre de 2013 tomé parte en un importante proceso de selección en el que las comunicaciones se estaban llevando a cabo a través de correo electrónico. En particular, la cuenta de correo electrónico que estaba usando ([info@josetomastocino.com](mailto:info@josetomastocino.com)) tenía un dominio personalizado y estaba gestionada a través de Google Apps [20]. Esto era posible gracias a que en el dominio se dieron de alta unos registros de tipo MX y TXT que hacían que el correo se redirigiese a los servidores de Google.

A mitad del proceso de selección, la empresa que gestionaba el dominio de mi web personal (y, por extensión, mi correo electrónico) tuvo un problema y reemplazó los registros DNS personalizados por unos por defecto. En particular, los registros MX y TXT quedaron en blanco, y los registros A apuntaron a una página de parking<sup>1</sup>, de forma que los chequeos de tipo ping que tenía puestos no dieron error, ya que efectivamente la web devolvía el ping, pero el contenido de los registros no era el correcto, ni los correos se estaban direccionando bien.

Esta situación se prolongó durante varios días, en los cuales perdí varios mensajes importantes. Esto podría haberse detectado rápidamente si hubiera tenido alguna clase de vigilancia que verificase que los registros DNS tenían el contenido correcto.

De ahí nace la necesidad de realizar un estudio de las alternativas existentes para este tipo de vigilancia y, al concluir que no existía ninguna alternativa libre, se decide iniciar el proyecto SiteUp para la vigilancia de servicios de internet.

## 1.3. Objetivos

A la hora de definir los objetivos de un sistema, podemos agruparlos en dos tipos diferentes: **funcionales** y **transversales**. Los primeros se refieren a qué debe hacer la aplicación que vamos a desarrollar, e inciden directamente en la experiencia del usuario y de potenciales desarrolladores.

Por otro lado, los objetivos transversales son aquellos invisibles al usuario final, pero que de forma inherente actúan sobre el resultado final de la aplicación y sobre la experiencia de desarrollo de la misma.

### 1.3.1. Funcionales

- Crear un conjunto de herramientas para la monitorización y el chequeo de diversos aspectos del estado de un servicio de Internet.
- Crear una aplicación online, de acceso público, que permita la creación y gestión de chequeos de manera sencilla, basada internamente en las herramientas mencionadas en el punto anterior.

---

<sup>1</sup>Una página de parking de dominio sirve como página temporal mientras un servicio web no es lanzado, de forma que el dominio no esté en blanco.

- Habilitar esta aplicación de un sistema de notificaciones mediante correo electrónico que alerte a los usuarios de posibles cambios en la disponibilidad de los servicios monitorizados.
- Crear una aplicación móvil para que los usuarios tengan la opción de recibir notificaciones instantáneas provenientes de la aplicación web con información de sus cheques.

### 1.3.2. Transversales

- Investigar y conocer los vectores de vigilancia usados habitualmente para monitorizar servicios de Internet.
- Ampliar mis conocimientos sobre desarrollo web en general y las tecnologías de back-end en particular.
- Adquirir soltura en el uso del lenguaje de programación Python en entornos web.
- Obtener una base de conocimientos mínima sobre el desarrollo de aplicaciones sobre la plataforma móvil Android.
- Utilizar un enfoque de análisis, diseño y codificación orientado a objetos, de una forma lo más clara y modular posible, para permitir ampliaciones y modificaciones sobre la aplicación por terceras personas.
- Hacer uso de herramientas básicas en el desarrollo de software, como son los **Sistemas de Control de Versiones** para llevar un control realista del desarrollo del software, así como hacer de las veces de sistema de copias de seguridad.

## 1.4. Estado del arte

En la actualidad existen bastantes servicios web que ofrecen algunas características similares a las que se desean en SiteUp, pero no todas. Se presentan a continuación algunos de estos servicios, junto a los problemas que se han detectado.

**Pingdom** <http://pingdom.com> – La opción más veterana y popular, con clientes muy importantes. La cuenta gratuita solo permite añadir un check. No ofrece chequeo de registros DNS.

**Alertra** <http://alertra.com> – La cuenta gratuita solo dura 30 días. No ofrece chequeo de registros DNS.

**UptimeRobot.com** <http://uptimerobot.com> – Periodicidad mínima de 5 minutos. No ofrece chequeo de DNS.

**ServerCheck** <http://servercheck.in> – Aspecto amateur. No tienen cuenta gratuita. No tienen chequeo de DNS.

**StatusCake** <http://statuscake.com> – Periodicidad mínima de 5 minutos. No permite el chequeo de códigos de estado en su cuenta gratuita.

Una carencia importante que no se ha mencionado por ser generalizada es la falta de aplicaciones nativas para móviles.

Así pues, queda patente la dificultad de encontrar un servicio que aúne el mayor número de características posibles a la vez que mantiene un servicio gratuito y de calidad, quedando manifiesta la necesidad del proyecto.

## 1.5. Alcance

**SiteUp** se modela como una herramienta de monitorización de servicios de internet accesible a través de la web. Los usuarios tendrán la posibilidad de crear y gestionar una serie de *chequeos* de diversos tipos sobre los servicios web que elijan. La aplicación irá recopilando información relativa a esos chequeos, e informará al usuario en caso de que las verificaciones que se hayan dado de alta no coincidan con los resultados obtenidos.

Además, el usuario tendrá la posibilidad de recibir notificaciones de manera instantánea a través del correo electrónico y de una aplicación para la plataforma móvil **Android**. El servicio web estará totalmente adaptado para su uso en dispositivos móviles.

### 1.5.1. Limitaciones del proyecto

Aunque cubre una gran parte de los puntos de vigilancia habituales, la aplicación se limita a ofrecer chequeo de respuesta de ping, chequeo de puertos, chequeo de registros DNS y chequeo de cabeceras y contenidos HTTP.

La aplicación de Android no contará con ninguna funcionalidad para gestionar los chequeos de un usuario, sino que servirá para recibir notificaciones instantáneas provenientes de la aplicación web. Ésta, por otro lado, estará completamente adaptada para su uso a través de dispositivos móviles gracias al uso del *responsive web design*.

Idealmente los chequeos deberían hacerse simultáneamente desde diferentes máquinas colocadas en diversos puntos geográficos, para así tener unos resultados más fiables. La falta de infraestructuras y la finalidad didáctica del proyecto limitan la aplicación a una estructura monolítica en la que los chequeos se hacen desde una sola máquina, la misma que sirve el servicio web.

### 1.5.2. Licencia

El proyecto está publicado como software libre bajo la licencia GPL (General Public License) versión 3. El conjunto de bibliotecas y módulos utilizados tienen las siguientes licencias:

- El intérprete del lenguaje de programación **Python** se distribuye bajo la licencia PSFL (Python Software Foundation License), una licencia permisiva estilo BSD (Berkeley Software Distribution) y compatible con la GNU (GNU is Not Unix) GPL.
- **Django** [13], el framework de desarrollo web en el que se basa la aplicación, se distribuye bajo la licencia *BSD*.
- El servidor web **nginx** [30] está licenciado bajo la licencia *BSD* simplificada.
- Los siguientes paquetes de Python utilizan también la licencia *BSD*:
  - Celery
  - Sqlparse
  - iPython
  - dnspython
  - coverage
  - django-rest-framework
  - billiard
  - anyjson
  - Fabric
- Los siguientes paquetes de Python utilizan la licencia MIT (Massachusetts Institute of Technology):
  - PyDot
  - Gunicorn
  - Requests
  - django-extensions
  - six
  - sh
  - pip
  - virtualenvwrapper
  - factory-boy

## 1.6. Estructura del documento

El presente documento se rige según la siguiente estructura:

- **Introducción.** Se exponen el contexto, las motivaciones y objetivos detrás del proyecto **SiteUp**, así como información sobre las licencias de sus componentes, glosario y estructura del documento.
- **Planificación**, donde se explica la planificación del proyecto, la división de sus etapas, la extensión de las etapas a lo largo del tiempo y los porcentajes de esfuerzo.
- **Requisitos del sistema**, capítulo en el que se formalizan los objetivos y requisitos planteados en la introducción.
- **Análisis.** Se detalla la fase de análisis del sistema, explicando los requisitos funcionales del sistema, los diferentes casos de uso y bosquejando las interfaces visuales de los diferentes sistemas.
- **Diseño.** Seguido del análisis, se expone en detalle la etapa de diseño del sistema, con los diagramas de las arquitecturas lógicas de los sistemas y los diagramas de diseño físico de datos.
- **Implementación.** Una vez analizado el sistema y definido su diseño, en esta parte se detallan las decisiones de implementación más relevantes que tuvieron lugar durante el desarrollo del proyecto.
- **Pruebas.** Listamos y describimos las pruebas que se han llevado a cabo sobre el proyecto para garantizar su fiabilidad y consistencia.
- **Conclusiones.** Comentamos las conclusiones a las que se han llegado durante el transcurso y al término del proyecto.

Y los siguientes apéndices:

- **Manual de instalación de la plataforma web** en sistemas nuevos.
- **Manual de usuario**, donde se explica cómo usar la aplicación.

## 1.7. Acrónimos

**ORM** Object-relational mapping

**AMQP** Advanced Message Queuing Protocol

**LOPD** Ley Orgánica de Protección de Datos

**WSGI** Web Server Gateway Interface

**PSFL** Python Software Foundation License

**API** Application Programming Interface

**BSD** Berkeley Software Distribution

**CRUD** Create-Read-Update-Delete

**CSS** Cascading Style Sheet(s)

**DNS** Domain Name Server

**FDL** Free Documentation License

**FSF** Free Software Foundation

**GNU** GNU is Not Unix

**GPL** General Public License

**HTML** Hyper Text Markup Language

**HTTP** Hyper Text Transfer Protocol

**ICMP** Internet Control Message Protocol

**MIT** Massachusetts Institute of Technology

**MVC** Model View Controller

**PDF** Portable Document Format

**SVG** Structured Vector Graphics

**URL** Uniform Resource Locator



# **Capítulo 2**

## **Planificación**

En este capítulo procederemos a detallar la planificación que ha seguido el proyecto. En particular, se describe la metodología seguida, las etapas por las que ha pasado el proyecto, la planificación temporal y la organización del personal involucrado.

Tras ello se exponen los recursos inventariables de los que se han hecho uso a lo largo del desarrollo y puesta en producción del producto, haciendo una estimación de los costes y, por último, explicando los posibles riesgos que pueden tener lugar.

En general, el desarrollo del proyecto se ha ajustado razonablemente bien al calendario inicialmente dispuesto en la planificación, con algunas etapas durando más de lo previsto pero otras terminando antes de lo planificado.

### **2.1. Metodología de desarrollo**

Para la realización del proyecto se ha utilizado un modelo de desarrollo iterativo incremental, definiendo en cada etapa las funcionalidades a elaborar y completando el ciclo completo de desarrollo y pruebas.

A continuación se detallan cada una de las iteraciones por las que ha ido pasando el proyecto.

#### **2.1.1. Primera iteración: adquisición de conocimientos y elicidadación de requisitos**

Antes de poder comenzar con el análisis y diseño del propio proyecto, era esencial adquirir una serie de conocimientos para poder afrontar su desarrollo con todas las garantías. Durante esta iteración, se llevaron a cabo labores de documentación y aprendizaje autodidacta con las que se asentaron los conocimientos necesarios.

Además, durante este periodo también se definieron los requisitos que se impondrían al sistema, y se barajaron las diferentes posibilidades de implementación del proyecto, así como las posibles herramientas y bibliotecas de terceros que pudieran ser de ayuda, tal y como queda reflejado en el capítulo 3.

En particular, en esta etapa se decidió el uso del framework web **Django**, las tecnologías de desarrollo *front-end*<sup>1</sup> (entre otras, Sass, Compass, jQuery y D3), el motor de tareas asíncronas para lanzar los chequeos (Celery y RabbitMQ) y el stack de soporte del servidor (nginx, supervisord y gunicorn).

### 2.1.2. Segunda iteración: desarrollo de módulo de herramientas básicas de chequeo

Una vez adquiridos los conocimientos necesarios, y decididas las técnicas y herramientas para llevar aquellos a la práctica, fue obvia la necesidad de empezar por diseñar una serie de herramientas que fuesen capaces de lanzar chequeos contra servicios web de forma simple y aislada.

Así, se desarrolló un módulo en Python capaz de lanzar los cuatro tipos de chequeos con los que posteriormente contaría el proyecto:

- Chequeo mediante paquetes ICMP (Internet Control Message Protocol) (ping) con verificación de tiempo de espera máximo.
- Chequeo de coherencia de registros DNS (Domain Name Server).
- Chequeo del estado de puertos remotos.
- Chequeo de peticiones HTTP (Hyper Text Transfer Protocol), tanto en su cabecera como en contenido.

Este módulo sería el motor de los chequeos que posteriormente se darían de alta en el proyecto.

### 2.1.3. Tercera iteración: inicio de proyecto Django y CRUD básico

Con el módulo de chequeo desarrollado, sólo restaba desarrollar el resto de la aplicación alrededor de sus funcionalidades. En esta tercera iteración se creó la estructura básica del proyecto y se inició el desarrollo de la funcionalidad CRUD (Create-Read-Update-Delete) básica.

También en esta etapa se definió el diseño visual de la aplicación: logotipo, esquema de colores y tipografías.

---

<sup>1</sup>El desarrollo web *front-end* hace referencia a la parte de una web con la que el usuario interacciona directamente en el navegador.

### 2.1.4. Cuarta iteración: integración del motor de tareas asíncronas

Con la aplicación teniendo la funcionalidad básica para la creación y edición de chequeos, el siguiente paso fue integrar un motor de tareas asíncronas que se dedicase a revisar y lanzar los chequeos dados de alta en el sistema, guardando el resultado de cada uno de ellos en la base de datos y generando estadísticas.

### 2.1.5. Quinta iteración: desarrollo de la aplicación Android

Tras concluir el desarrollo de la aplicación web, en esta etapa se desarrolló una aplicación móvil para el sistema operativo Android que recibe notificaciones con información sobre los resultados de los chequeos dados de alta en el sistema.

## 2.2. Planificación temporal

Se ha diseñado un diagrama de Gantt para reflejar la distribución de las tareas a lo largo del tiempo (figura 2.1, en la página 30).

## 2.3. Organización

El proyecto ha sido desarrollado en su totalidad por el que escribe, alumno de la Universidad de Cádiz, llevando a cabo las labores de análisis, diseño y desarrollo, así como de diseño visual de las interfaces y *branding* del proyecto. El desarrollo del proyecto ha sido revisado y guiado de forma continua por el tutor Iván Ruiz Rube.

## 2.4. Recursos inventariables

Los recursos de hardware utilizados durante el desarrollo y la implantación del proyecto engloban aquellos empleados por el alumno para la elaboración del proyecto y los necesarios para la implantación y puesta en marcha del proyecto.

En particular:

- Como puesto de desarrollo se ha utilizado un equipo con las siguientes características:
  - Procesador Intel i5 4670k
  - Placa Gigabyte Z87X-OC

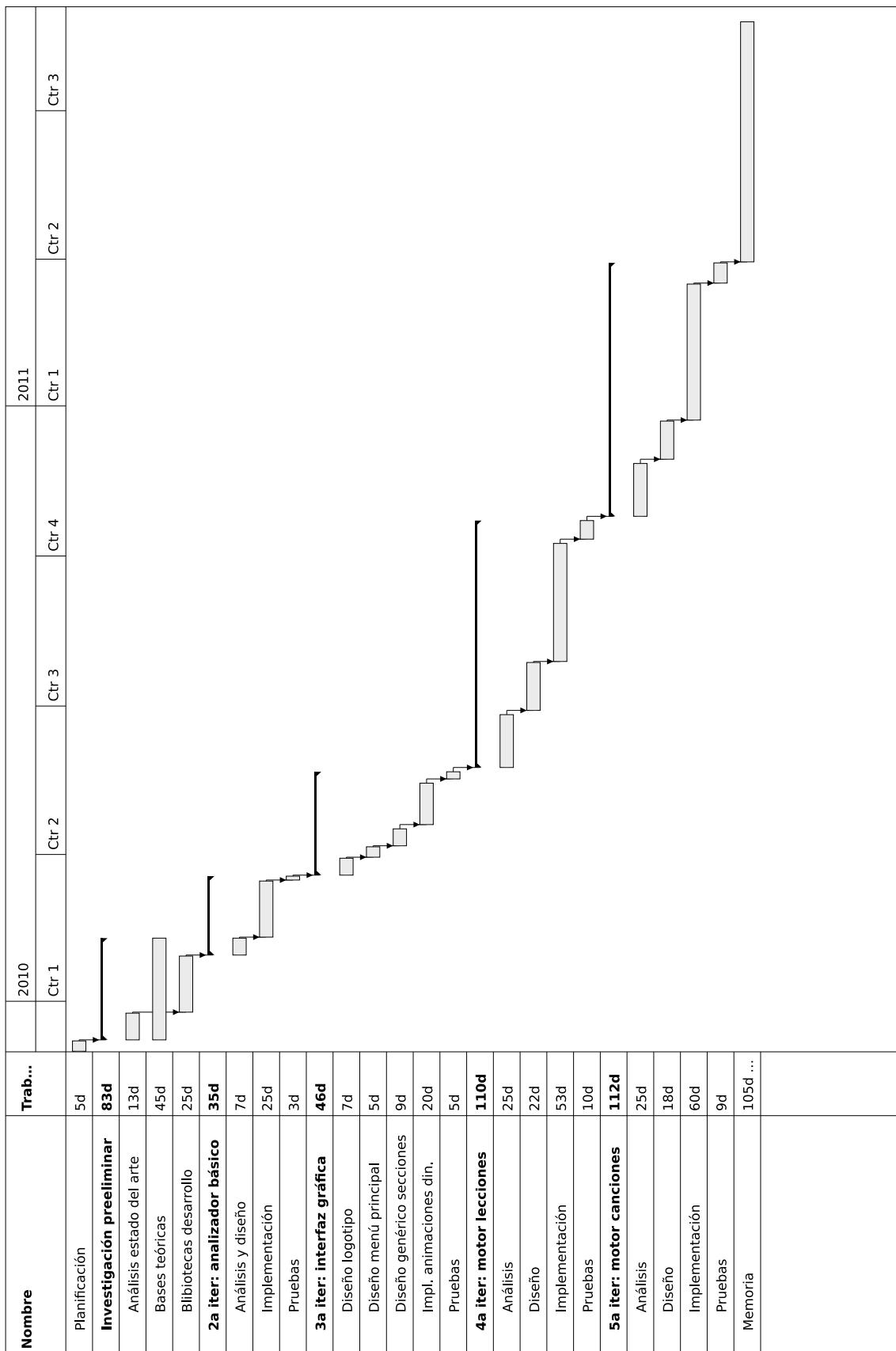


Figura 2.1: Diagrama de Gantt

- Memoria Corsair Vengeance 16GB DDR3
  - Gráfica NVidia GeForce GTX 660 Ti
  - SSD Crucial M4 128GB
  - Pantalla Dell u2713H 27"
- Como servidores para el *deployment* se han contratado los servicios de la empresa alemana Hetzner, haciendo uso del plan *vServer VQ7* que proporciona un servidor virtual con las siguientes características:
    - Procesador Single Core
    - RAM 512MB
    - Disco Duro 20GB
    - NIC 100Mbit
    - Tráfico máximo 1TB al mes
  - Como dispositivo para las pruebas de la aplicación móvil se ha utilizado un **Samsung Galaxy Nexus** con las siguientes características:
    - Sistema: Android 4.3
    - CPU: 1.2 GHz ARM Cortex-A9 de doble núcleo.
    - GPU: 307 MHz PowerVR SGX540.
    - Memoria: 1 GB.
    - Almacenamiento: 16GB
    - Pantalla: 4.65"HD Super AMOLED, 1280x720px.

## 2.5. Costes

La estimación de los costes del proyecto es divisible en dos áreas. Por un lado, el coste de personal, que en este caso particular se limita al alumno, y por otro lado el coste de infraestructuras y servicios externos. No se tiene en cuenta el coste del software ya que en todo momento se ha utilizado software con licencias libres para uso personal y comercial.

En lo que respecta al personal:

- La realización del proyecto necesitó de 480 horas de trabajo. De acuerdo a la planificación, con una carga de trabajo diaria de 4 horas, la duración del proyecto ha sido de 120 días, aproximadamente cuatro meses.
- Según el estado de mercado, el salario medio de un programador Python/Django de categoría junior ronda los 17€ por hora trabajada antes de impuestos.

- Así pues, la estimación de gastos de personal es de **8160€**.

En cuanto al gasto en infraestructuras:

- El sistema en el que se ha llevado a cabo el desarrollo estaba comprado con anterioridad al proyecto, por lo que no se ha reflejado en este despliegue de gastos.
- La plataforma de despliegue del sistema tiene un coste fijo de 7.90€ al mes [23], impuestos incluidos. Suponiendo un periodo mínimo de actividad de 12 meses, el coste asciende a **94.80€**.

## 2.6. Riesgos

Los principales riesgos presentes en SiteUp son los siguientes.

### 2.6.1. Compromiso de la plataforma de despliegue

Dada la limitada envergadura del proyecto por ser éste un PFC y no un desarrollo con fines comerciales, los chequeos de los servicios web se hacen de forma centralizada desde la plataforma de despliegue. Cualquier problema que pueda comprometer la estabilidad de esta plataforma supone un riesgo de obtener datos falsos de los chequeos. En un entorno real, para obtener la mayor fiabilidad y datos siempre precisos, cada chequeo se lanza desde diversos puntos, tanto geográficos como lógicos, de forma que es más difícil que un falso positivo o negativo causado por un error de la plataforma pueda llegar hasta el usuario.

Lógicamente la adición de nuevas *sondas* de chequeo incrementaría enormemente la complejidad logística y el presupuesto del proyecto.

### 2.6.2. Limitación de recursos

SiteUp hace uso de un gran número de sistemas, desde el servidor web que recibe las peticiones hasta la cola de tareas asíncronas. La plataforma de despliegue cuenta con unos recursos limitados que, en situaciones de gran afluencia de peticiones, pueden no ser suficientes y dar lugar a problemas en el sistema.

Este riesgo puede superarse considerando contratar planes con más recursos para la plataforma, lo que por otro lado conllevaría un incremento en los gastos.

### 2.6.3. Rechazo de los servicios vigilados

Por definición, los sistemas de vigilancia tienen que lanzar chequeos de forma periódica y repetida a lo largo del tiempo. Algunos sistemas están configurados

para detectar esta clase de comprobaciones y bloquearlas, al identificarlas (a veces erróneamente) como ataques de denegación de servicio [43].

Este riesgo es algo que tenemos que asumir y frente al que no se puede hacer mucho, aparte de configurar la periodicidad de las comprobaciones para disminuir la frecuencia y que no se tomen como un ataque.



# Capítulo 3

## Requisitos del sistema

En este capítulo procederemos a describir de manera formal los objetivos que se presentaron de manera genérica en el apartado 1.3, *Objetivos*.

En particular, se detallan:

- Los **objetivos del sistema**, que resumen a muy grandes rasgos la funcionalidad del proyecto.
- Los **requisitos funcionales**, que definen las funciones del sistema software y sus componentes.
- Los **requisitos de información**, que detallan los datos que el sistema usará durante el desarrollo y ejecución para llevar a cabo de forma adecuada su funcionalidad.
- Los **requisitos no funcionales**, que, en general, especifican criterios de diversas categorías que el software debe cumplir para garantizar un buen nivel de calidad más allá de su funcionamiento.

Para terminar el capítulo se presentan diversas alternativas tecnológicas para cubrir las necesidades surgidas en los requerimientos del sistema, detallando las decisiones tomadas.

### 3.1. Objetivos del sistema

Los objetivos principales de SiteUp, definidos a alto nivel son los dos siguientes:

Título	Crear chequeos de varios tipos sobre servicios de internet
Descripción	SiteUp permitirá crear chequeos de diversas clases para la comprobación de servicios de internet, utilizando varias tecnologías de chequeo y múltiples vectores de verificación: ping, puertos, hipertexto y registros DNS.

Cuadro 3.1: OBJ-1

Título	Notificación a los usuarios
Descripción	SiteUp deberá notificar a sus usuarios cuando el estado de sus cheques cambie. Estas notificaciones se harán tanto por móvil como a través de una aplicación Android.

Cuadro 3.2: OBJ-2

## 3.2. Catálogo de requisitos

Se presentan a continuación los requisitos del sistema, tanto a nivel funcional como de información y no funcionales.

### 3.2.1. Requisitos funcionales

#### Gestión de usuarios

Título	Creación de cuenta de usuario
Descripción	El usuario de SiteUp deberá ser capaz de crear una cuenta de usuario para acceder a las funciones de SiteUp, proporcionando su nombre de usuario, dirección de correo electrónico y contraseña.

Cuadro 3.3: RQF-1

Título	Edición de cuenta de usuario
Descripción	Un usuario logueado deberá ser capaz de editar sus datos personales y preferencias de usuario. En particular, deberá ser capaz de editar su nombre de usuario, dirección de correo electrónico y contraseña, además de opciones como la posibilidad de recibir un resumen diario del estado de sus cheques.

Cuadro 3.4: RQF-2

#### Gestión de grupos de cheques

Título	Creación de grupos de cheques
Descripción	Un usuario logueado deberá ser capaz de crear un grupo de cheques, indicando el título que identifique al grupo.

Cuadro 3.5: RQF-3

Título	Edición de grupos de cheques
Descripción	Un usuario logueado deberá ser capaz de editar un grupo de cheques que previamente haya creado. En particular, deberá ser capaz de modificar el título del grupo.

Cuadro 3.6: RQF-4

Título	Eliminación de grupos de cheques
Descripción	Un usuario logueado deberá ser capaz de eliminar un grupo de cheques, eliminando en el proceso tanto el grupo en sí como los cheques que pertenezcan a ese grupo.

Cuadro 3.7: RQF-5

Título	Activación y desactivación de grupos de cheques
Descripción	Un usuario logueado deberá ser capaz de activar y desactivar un grupo de cheques. En la práctica, esto activará o desactivará cada uno de los cheques que pertenezcan al grupo de forma individual.

Cuadro 3.8: RQF-6

### Gestión de cheques

Título	Creación de cheques
Descripción	Un usuario logueado deberá ser capaz de añadir a un grupo un chequeo que puede ser de cuatro tipos distintos: <i>Ping check</i> , <i>Port check</i> , <i>DNS Check</i> y <i>HTTP Check</i> . Según el tipo de chequeo a dar de alta, el usuario deberá introducir una serie de detalles, siendo común a todos ellos el incluir un título que identifique el chequeo, el objetivo del chequeo, la periodicidad y las opciones de notificar mediante e-mail y Android. Los detalles particulares de cada chequeo se definen en los requisitos de información.

Cuadro 3.9: RQF-7

Título	Actualización de cheques
Descripción	Un usuario logueado deberá ser capaz de actualizar un determinado chequeo, modificando cualquiera de los detalles que lo definen (a excepción del tipo de chequeo, que es fijo).

Cuadro 3.10: RQF-8

Título	Eliminación de cheques
Descripción	Un usuario logueado deberá ser capaz de eliminar un determinado chequeo, evitando así que el sistema lo tenga en cuenta a la hora de lanzar las monitorizaciones.

Cuadro 3.11: RQF-9

Título	Activación y desactivación de cheques
Descripción	Un usuario logueado deberá ser capaz de activar o desactivar un chequeo, de forma que mientras esté activado, el chequeo será lanzado periódicamente por el motor de chequeos, y cuando esté desactivado el chequeo será ignorado por el sistema hasta que el usuario lo active de nuevo.

Cuadro 3.12: RQF-11

Título	Revisión de cheques
Descripción	Un usuario logueado deberá ser capaz de revisar los datos de monitorización que se hayan generado para un chequeo que previamente haya dado de alta, pudiendo ver una gráfica con el estado del chequeo a lo largo del tiempo. Además debe ser capaz de elegir el período de tiempo en el que revisar los datos (últimas 24 horas, última semana, etcétera).

Cuadro 3.13: RQF-11

Título	Recepción de notificaciones por correo electrónico
Descripción	Un usuario logueado deberá ser capaz de recibir notificaciones a través del correo electrónico cuando el estado de un chequeo cambie, siempre que el usuario haya activado las notificaciones por correo para ese chequeo.

Cuadro 3.14: RQF-12

### Aplicación Android

Título	Listado de cheques en la aplicación Android
Descripción	Un usuario deberá ser capaz de ejecutar la aplicación de Android, loguearse con su cuenta de usuario y ver un listado de los cheques que ha dado de alta en el sistema, así como un resumen de su estado.

Cuadro 3.15: RQF-13

Título	Recepción de notificaciones por la aplicación Android
Descripción	Un usuario logueado deberá ser capaz de recibir notificaciones a través de la aplicación Android cuando el estado de un chequeo cambie, siempre que el usuario haya activado las notificaciones para ese chequeo.

Cuadro 3.16: RQF-14

### 3.2.2. Requisitos de información

	Usuario del sistema
Datos específicos	<ul style="list-style-type: none"> <li>■ Nombre de usuario</li> <li>■ Dirección de correo electrónico</li> <li>■ Contraseña</li> <li>■ (Opcional) Identificador de dispositivo Android</li> </ul>

Cuadro 3.17: IRQ-1

	Chequeo tipo Ping
Datos específicos	<ul style="list-style-type: none"> <li>■ Título</li> <li>■ Descripción</li> <li>■ Objetivo – Debe ser una IP o un hostname</li> <li>■ Frecuencia de chequeo en minutos</li> <li>■ Notificar por correo</li> <li>■ Notificar por Android</li> <li>■ Si se debe chequear el tiempo de respuesta</li> <li>■ Tiempo máximo de respuesta en milisegundos</li> </ul>

Cuadro 3.18: IRQ-2

Chequeo tipo DNS	
Datos específicos	<ul style="list-style-type: none"> <li>■ Título</li> <li>■ Descripción</li> <li>■ Objetivo – Debe ser un hostname</li> <li>■ Tipo de registro DNS (A, AAAA, CNAME, MX, TXT)</li> <li>■ Contenido esperado del registro DNS</li> <li>■ Frecuencia de chequeo en minutos</li> <li>■ Notificar por correo</li> <li>■ Notificar por Android</li> </ul>

Cuadro 3.19: IRQ-3

Chequeo tipo Port	
Datos específicos	<ul style="list-style-type: none"> <li>■ Título</li> <li>■ Descripción</li> <li>■ Objetivo – Debe ser una IP o un hostname</li> <li>■ Puerto objetivo</li> <li>■ <i>Opcional</i> - Cadena de caracteres que debe aparecer en la respuesta</li> <li>■ Frecuencia de chequeo en minutos</li> <li>■ Notificar por correo</li> <li>■ Notificar por Android</li> </ul>

Cuadro 3.20: IRQ-4

Chequeo tipo HTTP	
Datos específicos	<ul style="list-style-type: none"> <li>■ Título</li> <li>■ Descripción</li> <li>■ Objetivo – Debe ser una URL HTTP</li> <li>■ Código de estado HTTP</li> <li>■ <i>Opcional</i> - Cadena de caracteres que debe aparecer en la respuesta</li> <li>■ Frecuencia de chequeo en minutos</li> <li>■ Notificar por correo</li> <li>■ Notificar por Android</li> </ul>

Cuadro 3.21: IRQ-5

	Registro de un chequeo
Datos específicos	<ul style="list-style-type: none"> <li>■ Fecha y hora de registro</li> <li>■ Estado (Up, Down, Error)</li> <li>■ Tiempo de respuesta (solo para chequeos tipo Ping).</li> <li>■ Información adicional</li> <li>■ Chequeo asociado</li> </ul>

Cuadro 3.22: IRQ-6

	Estado de un chequeo
Datos específicos	<ul style="list-style-type: none"> <li>■ Fecha y hora de inicio del estado</li> <li>■ Fecha y hora de fin del estado</li> <li>■ Estado (Up, Down, Error)</li> <li>■ Información adicional</li> <li>■ Chequeo asociado</li> </ul>

Cuadro 3.23: IRQ-7

	Grupo de chequeos
Datos específicos	<ul style="list-style-type: none"> <li>■ Título</li> <li>■ Dueño</li> </ul>

Cuadro 3.24: IRQ-8

### 3.2.3. Requisitos no funcionales

#### Requisitos de seguridad

Título	Cifrado de contraseñas de usuario
Descripción	Las contraseñas de los usuarios se almacenarán cifradas con un algoritmo de un solo sentido, de acuerdo a lo establecido en la vigente LOPD.

Cuadro 3.25: NRQ-1

Título	Restricciones de acceso a usuarios
Descripción	Los datos de los chequeos dados de alta sólo serán accesibles por los usuarios que los hayan creado y los administradores de la plataforma.

Cuadro 3.26: NRQ-2

### Requisitos de fiabilidad

Título	Ejecución de chequeos
Descripción	Los retrasos en la periodicidad de los chequeos no puede superar el 200 % del tiempo entre chequeos. Esto es, si un chequeo tiene una periodicidad de 1 minuto, en caso de sobrecarga del sistema el tiempo máximo entre chequeos será de 3 minutos (un retraso del 200 %).

Cuadro 3.27: NRQ-3

### Requisitos de accesibilidad y usabilidad

Título	Adaptación a dispositivos
Descripción	La plataforma web deberá ser accesible desde cualquier clase de dispositivo móvil con acceso a internet: móviles, tablets, etcétera, manteniendo siempre la accesibilidad y el grueso del contenido.

Cuadro 3.28: NRQ-4

### Requisitos de eficiencia

Título	Monitorización de peticiones de demasiada duración
Descripción	En aras de mantener el sistema fluyendo, el motor de chequeos deberá ser capaz de identificar y abortar los chequeos que, por causas propias o ajenas al sistema, estén tardando más de lo previsto e incidiendo negativamente en la eficiencia del sistema.

Cuadro 3.29: NRQ-5

### 3.3. Alternativas de solución

En esta sección, se presentan diferentes alternativas tecnológicas que permiten satisfacer los requerimientos del sistema, presentando las alternativas elegidas y detallando las razones de esta decisión.

#### 3.3.1. Frameworks para el desarrollo de la plataforma web

Dada la mayor familiaridad del desarrollador con el lenguaje, solo se barajaron frameworks de desarrollo basados en el lenguaje Python.

- **Django**[13]: popular framework para el desarrollo rápido de aplicaciones web. Consta de una arquitectura similar al popular patrón MVC (Model View Controller), y cuenta con una basta comunidad de usuarios. Los proyectos Django se organizan en *apps*, que permiten modularizar la funcionalidad. Integra un ORM (Object-relational mapping), un sistema de plantillas y una sección de administración adaptable a cualquier sitio web.
- **Flask**[17]: se trata de un minimalista framework para Python que provee las funcionalidades más básicas, dejando al desarrollador elegir el resto de componentes. En los últimos meses, Flask ha ganado gran popularidad frente a Django al dar más libertad al programador.
- **Bottle**[6]: similar a Flask, Bottle es otro micro-framework para Python, siendo tan minimalista que está encapsulado en un solo fichero fuente y no necesita de dependencias externas. Se encuentra en etapas tempranas de su desarrollo.

Se tomó la decisión de optar por **Django** por ser el framework más robusto y con mayor número de utilidades. Su uso por grandes proyectos, como Instagram, afianzó aún más la decisión de usarlo.

#### 3.3.2. Sistemas de gestión de bases de datos

- **MySQL**[29], uno de los sistemas de bases de datos más populares. Se usa en multitud de proyectos de gran envergadura y su utilidad y fiabilidad están más que demostradas. Cuenta con una arquitectura cliente-servidor, y suele necesitar bastantes recursos.
- **SQLite**[37]: potente pero ligero, SQLite se aleja de la arquitectura cliente-servidor y se integra directamente en la aplicación cliente pasando a ser parte integral del mismo. Esto reduce la latencia en las comunicaciones de datos. Es el SGBD que Django usa por defecto.

En este aspecto, se ha tomado la decisión de utilizar SQLite por varias razones. En primer lugar, es la opción que usa Django por defecto. En segundo lugar, los datos

se guardan en un fichero, por lo que resulta trivial mover el sistema de un servidor a otro, lo cual es un gran ahorro de tiempo durante el desarrollo. Finalmente, SQLite provee el rendimiento necesario para las etapas tempranas del proyecto.

De cualquier modo, en un futuro no se descarta pasar a un SGBD basado en la arquitectura cliente-servidor, como MySQL.

### 3.3.3. Sistemas para la ejecución asíncrona de tareas

- **Cron**<sup>[11]</sup> es un demonio<sup>1</sup> encargado de lanzar procesos de forma periódica utilizado en sistemas operativos basados en Unix. Cada usuario en un sistema cuenta con un fichero *crontab*, en el que puede definir qué tareas ejecutar y con qué periodicidad deben ejecutarse.
- **Celery**<sup>[8]</sup> es una cola de tareas asíncronas desarrollado en Python. Una instalación de celery cuenta con una serie de *workers* que leen tareas de una cola de mensajes (basada en el estándar AMQP (Advanced Message Queuing Protocol)) y las ejecutan de forma síncrona o asíncrona, opcionalmente guardando su resultado en alguna clase de estructura.

De las opciones presentadas finalmente se decidió usar **Celery** por varias razones:

- Facilidad al dar de alta nuevas tareas. En el caso de cron, es necesario gestionar manualmente las tareas a ejecutar y añadirlas también de forma manual, el proceso que lanza esas tareas al fichero *crontab*. En el caso de celery, contamos con unas bibliotecas con las que simplemente damos de alta una tarea y el sistema se encarga de ponerla en cola y ejecutarla en el momento adecuado.
- Monitorización. Una de las mayores dificultades que presenta cron es la monitorización de su ejecución. No existe una certeza absoluta de que se estén lanzando los procesos, al menos que se elabore un sistema de verificación externo. En cambio, celery cuenta con sistemas de monitorización de la ejecución y *logs* en los que se registra el progreso de cada tarea.
- Integración. Al estar escrito en Python, celery se integra perfectamente con cualquier proyecto Django. Sin embargo, cron no ofrece de forma nativa ninguna opción de integración.

### 3.3.4. Plataformas para el desarrollo de la aplicación móvil

Como se especificó en la sección 1.3, uno de los objetivos principales del proyecto era el desarrollo, junto a la plataforma web, de una aplicación móvil para que los usuarios pudiesen recibir notificaciones instantáneas en cualquier momento y lugar, directamente en su móvil. Así pues, resultó necesario hacer un estudio de las plataformas móviles existentes para ver en cuál de ellas se centraría el desarrollo.

---

<sup>1</sup>Entiéndase *demonio* en el sentido de un servicio que se ejecuta en segundo plano.

- **iOS[5]** es un sistema operativo móvil de la empresa **Apple** utilizado en los dispositivos de la familia iPhone, iPod Touch e iPad. A pesar de estar basado en un núcleo BSD libre, iOS tiene una licencia privativa.

El desarrollo para iOS es bastante estricto. Sólo es posible desarrollar para iOS utilizando computadoras Mac con el sistema operativo OS X 10.8 o superior[38]. La programación se lleva a cabo utilizando **Objective-C**, un lenguaje derivado del C cuyo uso a día de hoy prácticamente se limita al desarrollo en iOS. Además, es necesario utilizar el conjunto de herramientas que provee Apple, en particular XCode.

El control de Apple sobre las aplicaciones que se desarrollan es férreo. La *App Store*, aplicación desde la que los usuarios instalan nuevos programas, controla fuertemente las aplicaciones que se envían mediante exhaustivas revisiones, que suelen tardar entorno a una semana. Además, para poder lanzar una app es necesario pagar una cuota anual de 99 dólares.

- **Android[3]** es un sistema operativo móvil basado en el kernel de Linux, diseñado para su uso en móviles y tabletas. Inicialmente su desarrollo fue llevado a cabo por la empresa Android, Inc, pero posteriormente Google compró la empresa y es la que en la actualidad impulsa el desarrollo.

El desarrollo en Android es bastante más laxo que en iOS. El lenguaje de programación utilizado es Java, un lenguaje compilado muy conocido y utilizado en muchos ámbitos, más allá del desarrollo móvil. Es posible desarrollar en cualquier plataforma (GNU/Linux, Windows o Mac), utilizando cualquier entorno de desarrollo que sea capaz de compilar código Java con las bibliotecas de Android.

Las aplicaciones desarrolladas para Android se pueden lanzar en un mercado de aplicaciones llamado *Google Play Store*, aunque también existen mercados de terceros al igual que es posible instalar aplicaciones de forma manual. Google cobra una tarifa única de 25 dólares por desarrollador, válida de por vida. Las revisiones que hace Google de las aplicaciones son menos rígidas que en el caso de Apple, y suelen tardar mucho menos.

- **BlackBerry, Windows Phone y Symbian** son otras paltaformas móviles de mucho menor calado y público. El desarrollo para esas plataformas está mucho menos extendido, tiene menor soporte y menor clientela.

BlackBerry OS es un sistema operativo que nació en 1999, época en la que BlackBerry era la compañía estrella en dispositivos *handheld*, siendo la opción principal para empresarios y gente de negocios que tuviese necesidad de chequear el correo electrónico o notas estando lejos de una computadora. En la actualidad, el número de dispositivos que utilizan BlackBerry ha caído estrepitosamente, principalmente por la limitada compatibilidad del sistema BlackBerry 10 con otros dispositivos.

Windows Phone es la apuesta de Microsoft en el mercado móvil. Lanzado en 2010, la última versión del sistema es Windows Phone 8.1. Actualmente

se encuentra en pleno crecimiento, principalmente gracias al dominio de Nokia en mercados crecientes como Asia, América Latina y África[42]. A pesar de ello, todavía no es una plataforma firme con una base de usuarios a considerar.

Symbian es una plataforma libre que tuvo un gran éxito en la década de 2000, siendo la alternativa más asequible a BlackBerry hasta la aparición de Android en 2010. La última versión del sistema se lanzó en octubre de 2012, y desde entonces no habido más desarrollo.

### Cuotas de mercado

En la tabla 3.30 se detallan las cuotas de mercado de las principales plataformas móviles medidas en enero de 2014 entre usuarios mayores de 13 años de edad [10].

<i>Plataforma</i>	<i>Porcentaje</i>
Android	51.7 %
Apple	41.6 %
BlackBerry	3.1 %
Microsoft	3.2 %
Symbian	0.2 %

Cuadro 3.30: Cuota de mercado de plataformas móviles en enero de 2014

### Decisión

Así, dadas las evidencias en la cuota de mercado, facilidad de desarrollo y costes en licencias, se tomó la decisión de usar **Android** como plataforma móvil principal a la hora de desarrollar la aplicación móvil.

En un futuro, según las circunstancias, se valorará trasladar el desarrollo también a la plataforma iOS, aunque es algo harto improbable dadas las exigencias de infraestructura (es necesario tener un ordenador Mac reciente) y monetarias (el coste de la licencia anual es prohibitivo para una aplicación gratuita).

# Capítulo 4

## Análisis

En este capítulo se dispone el análisis realizado en base a los requisitos presentados en el capítulo 3, haciendo uso del lenguaje de modelado UML. Concretamente, el capítulo describe:

- El **modelo conceptual** de los datos usados en la aplicación, elaborado a partir de los requisitos de información.
- Los **casos de uso**, que describen los pasos que componen los procesos habituales del proyecto.
- El **modelo de comportamiento** del sistema, donde se identifican las operaciones que se llevarán a cabo.
- El **modelo de interfaz de usuario**, en el que se presenta un prototipo de la navegación y de las interfaces.

### 4.1. Modelo conceptual

De acuerdo a lo reflejado en el apartado 3.2.2 se presenta el diagrama 4.1 en el que se identifican los principales tipos de datos, sus atributos y las relaciones entre ellos.

A continuación se detallan cada uno de los tipos de datos reflejados en el modelo conceptual.

#### 4.1.1. Usuario

**Nombre** Nombre elegido por el usuario.

**Correo electrónico** Dirección de correo electrónico.

**Contraseña** Contraseña para el login del usuario.

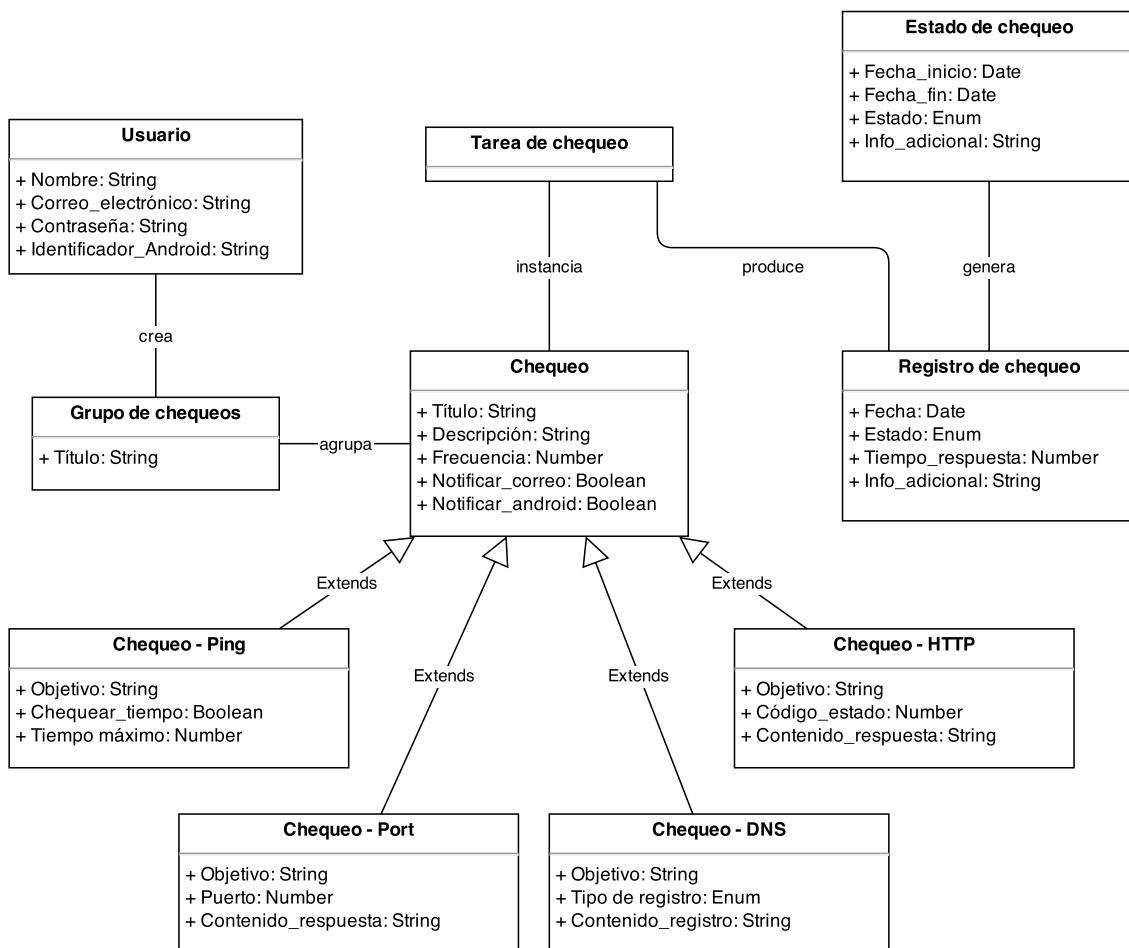


Figura 4.1: Modelo conceptual

**Identificador Android** Identificador único del dispositivo Android del usuario.

#### 4.1.2. Grupo de cheques

**Título** Título para identificar al grupo de cheques.

#### 4.1.3. Chequeo

**Título** Título para identificar al chequeo.

**Descripción** Información adicional en forma textual sobre el chequeo.

**Frecuencia de chequeo** Frecuencia temporal del chequeo, expresada en minutos.

**Notificar por correo** Indica si los cambios de estado del chequeo deben notificarse por correo electrónico.

**Notificar por Android** Indica si los cambios de estado del chequeo deben notificarse por la aplicación Android.

#### 4.1.4. Registro de chequeo

**Fecha** Fecha y hora en la que se obtuvo este registro.

**Estado** Resultado del lanzamiento del chequeo. El estado puede ser *Up*, que indica que el objetivo está funcionando; *Down*, que indica que el objetivo no está funcionando correctamente; y *Error*, que indica que hubo un problema lanzando el chequeo.

**Tiempo de respuesta** (*solo para chequeos tipo Ping*) indica el tiempo de respuesta medio obtenido al lanzar el chequeo.

**Información adicional** Datos extra en caso de que el estado no sea *Up*.

#### 4.1.5. Chequeo - Ping

**Objetivo** Identificador de la máquina a la que enviar el ping. Debe ser un *hostname* o una IP.

**Chequear tiempo?** Indica si además de chequear que la máquina responde al ping, también se debe chequear que el tiempo de respuesta sea menor que el indicado.

**Tiempo máximo** Tiempo de respuesta máximo permitido.

#### 4.1.6. Chequeo - Port

**Objetivo** Identificador de la máquina a chequear.

**Puerto objetivo** Puerto remoto en la máquina que se chequeará.

**Contenido de respuesta (opcional)** Cadena de caracteres que deberá estar presente en la respuesta obtenida de la máquina remota.

#### 4.1.7. Chequeo - DNS

**Objetivo** Dominio sobre el que realizar el chequeo de DNS.

**Tipo de registro** Tipo de registro DNS a verificar. Puede ser A, AAAA, CNAME, MX y TXT.

**Contenido de registro** El contenido que debe tener el registro a consultar.

#### 4.1.8. Chequeo - HTTP

**Objetivo** URL a verificar.

**Código de estado** Código que debe devolver la petición a la URL indicada.

**Cadena de respuesta (opcional)** Cadena que debe encontrarse en la respuesta obtenida.

### 4.2. Modelo de casos de uso

#### 4.2.1. Actores

En este apartado se describen los diversos roles que juegan los usuarios del sistema. Se reflejan todos ellos en el diagrama 4.2.

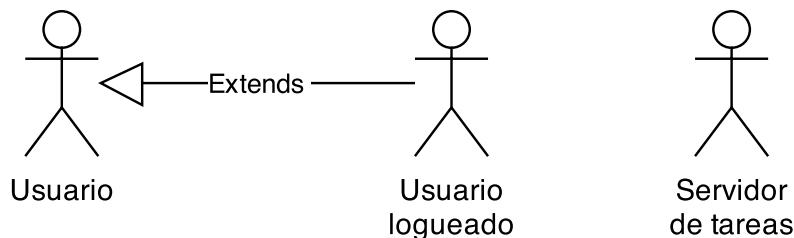


Figura 4.2: Diagrama de actores

### Usuario

	Usuario
Descripción	Usuario libre de la aplicación, que puede haber hecho login previamente o no.

### Usuario logueado

	Usuario logueado
Descripción	Usuario que previamente se ha registrado y ha hecho login en el sistema.

### Servidor de tareas

	Servidor de tareas
Descripción	Servidor de tareas asíncronas que periódicamente lanza tareas que previamente se hayan dado de alta.

#### 4.2.2. Subsistema de gestión de usuarios

Los casos de uso pertenecientes a este subsistema pueden verse en la figura 4.3

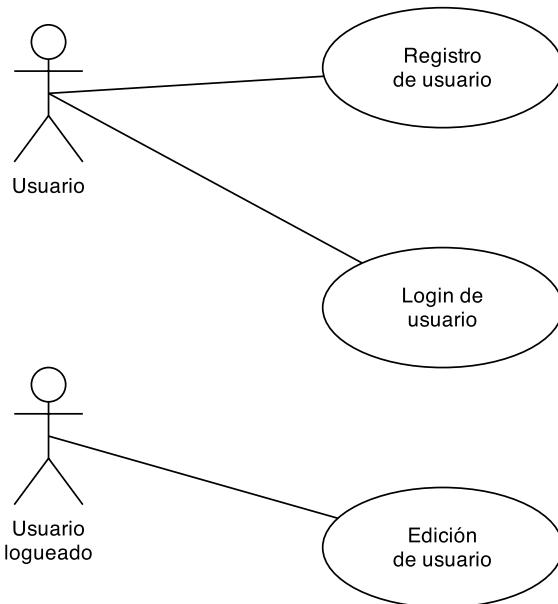


Figura 4.3: Casos de uso del subsistema de gestión de usuarios

### Caso de uso: registro de usuario

**Descripción** Un usuario no logueado se registra en la aplicación para poder dar de alta chequesos.

**Actores** *Usuario*.

#### Escenario principal

1. Un usuario decide registrarse en el sistema y accede al panel de registro.
2. El *usuario* introduce su nombre de usuario, dirección de correo electrónico y contraseña.
3. El *sistema* comprueba que los datos introducidos son correctos.
4. El *sistema* registra al usuario.

#### Flujo alternativo

- 3a** Alguno de los datos introducidos es incompleto o no es correcto. El sistema informa al usuario del error.
- 3b** El nombre de usuario o dirección de correo electrónico ya han sido usados por otro usuario previamente. El sistema informa al usuario del error.

### Caso de uso: login de usuario

**Descripción** Un usuario no logueado pero previamente registrado quiere hacer login en la aplicación.

**Actores** *Usuario*.

#### Escenario principal

1. Un usuario decide hacer login en el sistema.
2. El *usuario* accede al panel de login e introduce su nombre de usuario y contraseña.
3. El *sistema* comprueba que los datos introducidos son correctos.
4. El *sistema* loguea al usuario

#### Flujo alternativo

- 3a** Alguno de los datos introducidos no tiene el formato correcto o está en blanco. El sistema informa al usuario del error.
- 3b** Los datos introducidos no corresponden a ningún usuario registrado. El sistema informa al usuario del error.

### Caso de uso: edición de usuario

**Descripción** Un usuario logueado decide editar alguno de sus datos personales.

**Actores** *Usuario logueado*.

#### Escenario principal

1. El *usuario* accede a su perfil de usuario a través del enlace *Your profile*.
2. El *sistema* muestra los datos actuales.
3. El *usuario* edita los datos que quiera y envía el formulario.
4. El *sistema* comprueba los nuevos datos y guarda los cambios.

#### Flujo alternativo

- 3a Alguno de los nuevos datos no son válidos. El sistema informa del error al usuario.

### 4.2.3. Subsistema de gestión de grupos de cheques

Los casos de uso pertenecientes a este subsistema pueden verse en la figura 4.4

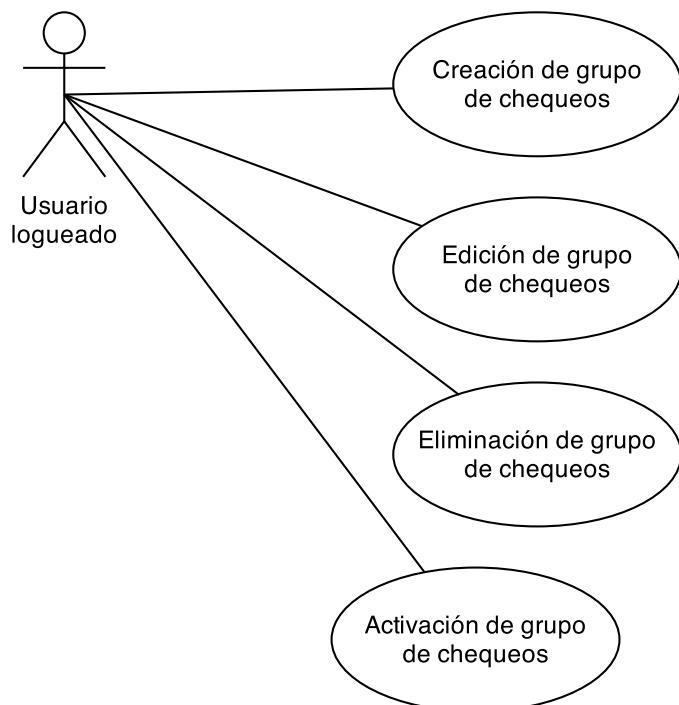


Figura 4.4: Casos de uso del subsistema de gestión de grupos de cheques

### Caso de uso: creación de grupo de cheques

**Descripción** Un usuario logueado decide crear un grupo de cheques.

**Actores** *Usuario logueado.*

#### Escenario principal

1. El *usuario* decide crear un nuevo grupo de cheques.
2. El *usuario* pulsa en el botón de *Añadir grupo*.
3. El *sistema* muestra el formulario para añadir nuevo grupo.
4. El *usuario* escribe el título del grupo.
5. El *sistema* valida el título introducido.
6. El *sistema* crea el nuevo grupo.

#### Flujo alternativo

- 5a El título introducido está en blanco o es inválido. El sistema informa al usuario del error.

### Caso de uso: edición de grupo de cheques

**Descripción** Un usuario logueado decide editar un grupo de cheques.

**Actores** *Usuario logueado.*

#### Escenario principal

1. El *usuario* decide editar un nuevo grupo de cheques.
2. El *usuario* pulsa en el botón de *Editar grupo* del grupo que quiera editar..
3. El *sistema* muestra el formulario para editar el grupo.
4. El *usuario* modifica el título del grupo.
5. El *sistema* valida el título introducido.
6. El *sistema* actualiza los datos del grupo.

#### Flujo alternativo

- 5a El título introducido está en blanco o es inválido. El sistema informa al usuario del error.

### Caso de uso: eliminación de grupo de cheques

**Descripción** Un usuario logueado decide eliminar un grupo de cheques.

**Actores** *Usuario logueado.*

**Escenario principal**

1. El *usuario* decide eliminar uno de sus grupos de cheques.
2. El *usuario* pulsa en el botón de eliminar grupo.
3. El *sistema* muestra el formulario para confirmar la eliminación.
4. El *usuario* confirma la eliminación.
5. El *sistema* elimina los cheques asociados al grupo.
6. El *sistema* elimina el grupo de cheques.

**Flujo alternativo**

- 3a** El *usuario* no confirma la eliminación. El *sistema* muestra la pantalla inicial.

**Caso de uso: activación de grupo de cheques**

**Descripción** Un usuario logueado decide activar un grupo de cheques.

**Actores** *Usuario logueado*

**Escenario principal**

1. El *usuario* decide activar un grupo de cheques.
2. El *usuario* pulsa en el botón de *Activar grupo*.
3. El *sistema* itera por todos los cheques pertenecientes al grupo, activando cada uno de ellos.
4. El *sistema* informa al usuario de que todos los cheques han sido activados.

**Flujo alternativo**

- 3a** El grupo no cuenta con cheques. El *sistema* no activa ningún chequeo.

**4.2.4. Subsistema de gestión de cheques**

Los casos pertenecientes a este subsistema pueden verse en la figura 4.5.

**Caso de uso: creación de chequeo**

**Descripción** Un usuario logueado decide crear un chequeo dentro de un grupo.

**Actores** *Usuario logueado*

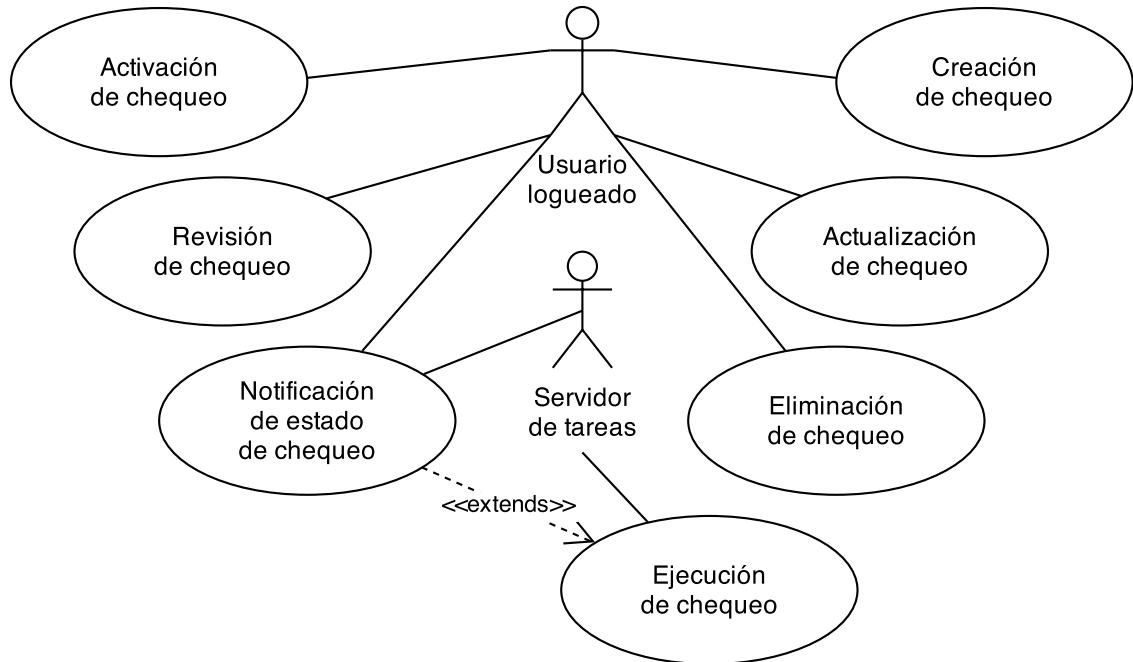


Figura 4.5: Casos de uso del subsistema de gestión de cheques

### Escenario principal

1. El *usuario* decide crear un chequeo dentro de un grupo de chequeos.
2. El *usuario* pulsa el botón de *Añadir chequeo* de un grupo.
3. El *sistema* muestra el formulario de elección del tipo de chequeo.
4. El *usuario* elige el tipo de chequeo a añadir.
5. El *sistema* muestra el formulario con los campos según el tipo de chequeo elegido.
6. El *usuario* introduce los detalles del chequeo y envía el formulario.
7. El *sistema* recibe y comprueba que los datos son correctos.
8. El *sistema* da de alta el nuevo chequeo.

### Flujo alternativo

- 7a Alguno de los datos introducidos no es correcto. El *sistema* informa del error al *usuario* y muestra el formulario de nuevo.

### Caso de uso: actualización de chequeo

**Descripción** Un usuario logueado decide editar un chequeo previamente creado..

**Actores** *Usuario logueado*

### Escenario principal

1. El *usuario* decide editar un chequeo dentro de un grupo de chequeos.
2. El *usuario* pulsa el botón de *Editar chequeo*.
3. El *sistema* muestra el formulario con los campos llenados con la información del chequeo elegido..
4. El *usuario* modifica los detalles del chequeo y envía el formulario.
5. El *sistema* recibe y comprueba que los datos son correctos.
6. El *sistema* actualiza el chequeo.

#### Flujo alternativo

- 5a Alguno de los datos introducidos no es correcto. El *sistema* informa del error al usuario y muestra el formulario de nuevo.

### Caso de uso: eliminación de chequeo

**Descripción** Un usuario logueado decide eliminar un chequeo previamente creado..

**Actores** *Usuario logueado*

#### Escenario principal

1. El *usuario* decide eliminar un chequeo dentro de un grupo de chequeos.
2. El *usuario* pulsa el botón de *Eliminar chequeo*.
3. El *sistema* muestra el formulario de confirmación de eliminación
4. El *usuario* confirma la eliminación del chequeo.
5. El *sistema* elimina el chequeo.

#### Flujo alternativo

- 4a El *usuario* no confirma la eliminación del chequeo. El *sistema* vuelve a la pantalla inicial.

### Caso de uso: activación de chequeo

**Descripción** Un usuario logueado decide activar un chequeo.

**Actores** *Usuario logueado*

#### Escenario principal

1. El *usuario* decide activar un chequeo dentro de un grupo de chequeos.
2. El *usuario* pulsa el botón de *activar chequeo*.
3. El *sistema* activa el chequeo.

### Caso de uso: ejecución de chequeo

**Descripción** El servidor de tareas tiene que lanzar un chequeo periódico.

**Actores** *Servidor de tareas*

#### Escenario principal

1. El *servidor de tareas* pide los chequeos activos.
2. El *sistema* devuelve los chequeos activos.
3. El *servidor de tareas* elige un chequeo activo.
4. El *servidor de tareas* ejecuta el chequeo.
5. El *sistema* devuelve el resultado del chequeo.
6. El *servidor de tareas* comprueba el resultado del chequeo, guardándolo en la base de datos y, si difiere del estado anterior del chequeo, lanzando el Caso de uso: *notificación de estado de chequeo*.

#### Flujo alternativo

- 2a** No hay chequeos activos. El *sistema* devuelve una lista vacía. El *servidor de tareas* permanece en reposo.

### Caso de uso: notificación de estado de chequeo

**Descripción** El servidor de tareas detecta un cambio de estado y lanza notificación.

**Actores** *Servidor de tareas*

#### Escenario principal

1. El *servidor de tareas* detecta un cambio de estado de un chequeo.
2. El *servidor de tareas* comprueba el campo *Notificar\_correo* del chequeo.
3. El *servidor de tareas* lanza una notificación por correo electrónico.
4. El *sistema* envía la notificación al usuario dueño del chequeo.
5. El *servidor de tareas* comprueba el campo *Notificar\_android* del chequeo.
6. El *servidor de tareas* lanza una notificación Android.
7. El *sistema* envía la notificación al usuario dueño del chequeo.

#### Flujo alternativo

- 2a** El chequeo no tiene activa la opción *Notificar\_correo*. El *servidor de tareas* no envía la notificación por correo.

- 5a El chequeo no tiene activa la opción *Notificar\_android*. El *servidor de tareas* no envía la notificación por Android.
- 5b El usuario no tiene dispositivo Android dado de alta. El *servidor de tareas* no envía la notificación por Android.

## 4.3. Modelo de comportamiento

Basándonos en los casos de uso anteriores se crea el modelo de comportamiento, compuesto de diagramas de secuencia, en los que se identificarán las operaciones del sistema, y de lo contratos de estas operaciones.

## TO-DO

## 4.4. Modelo de interfaz de usuario

### 4.4.1. Modelo de navegación

El modelo de navegación de la aplicación web se presenta en la figura 4.6. El modelo de navegación de la aplicación Android se presenta en la figura 4.7

### 4.4.2. Prototipos visuales de la interfaz web

A continuación se presentan los prototipos de las pantallas de la aplicación web. En concreto:

- La figura 4.8 representa la pantalla de bienvenida.
- La figura 4.9 representa la pantalla con el formulario para iniciar sesión.
- La figura 4.10 representa la pantalla para recuperar la contraseña.
- La figura 4.11 representa la pantalla para crear una cuenta de usuario.
- La figura 4.12 representa la pantalla de dashboard, en la que se ve la lista de cheques.
- La figura 4.13 representa la pantalla de detalle de chequeo.
- La figura 4.14 representa la pantalla de creación de nuevo grupo de cheques.
- La figura 4.15 representa la pantalla de creación de nuevo chequeo.

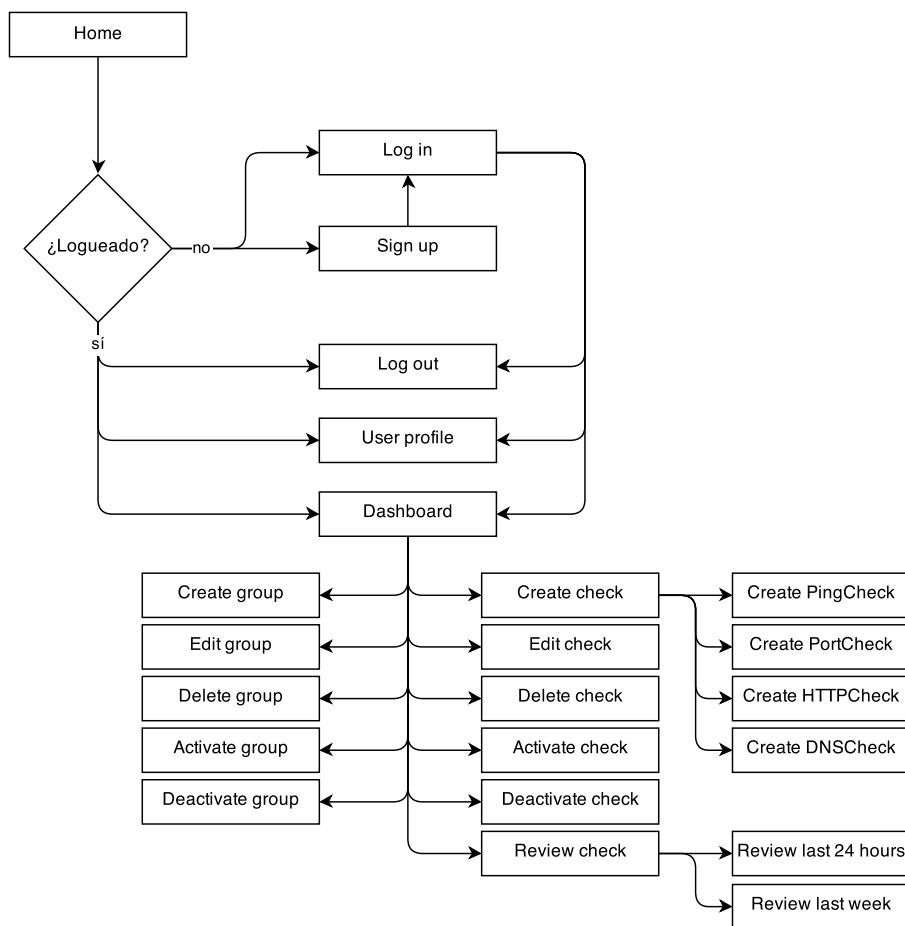


Figura 4.6: Modelo de navegación de la aplicación web

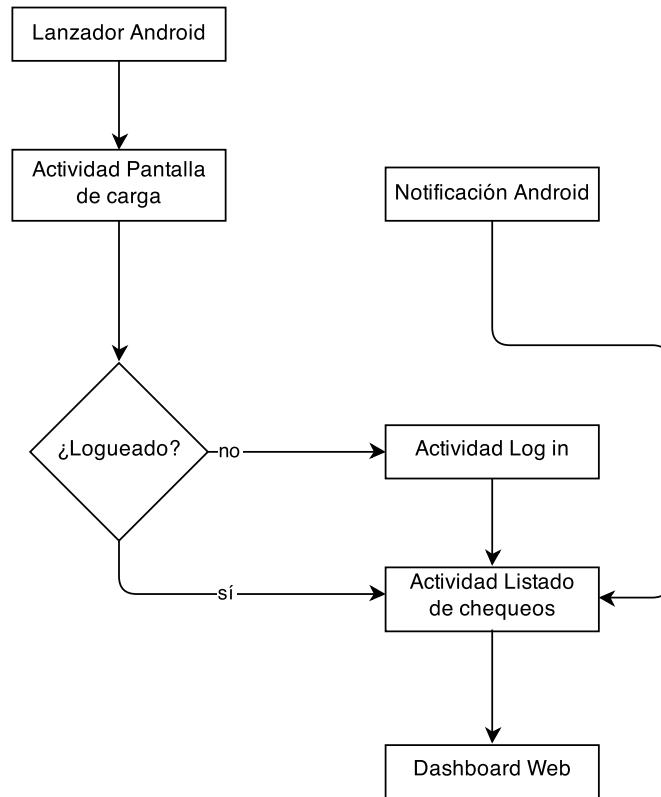


Figura 4.7: Modelo de navegación de la aplicación Android

#### 4.4.3. Prototipos visuales de la aplicación móvil

Seguidamente se presentan los prototipos de las interfaces visuales de la aplicación móvil. En particular:

- La figura 4.16 representa la pantalla de carga de la aplicación Android.
- La figura 4.17 representa la pantalla de login.
- La figura 4.18 representa la pantalla de lista de chequeos.
- La figura 4.19 representa una notificación recibida en el dispositivo.

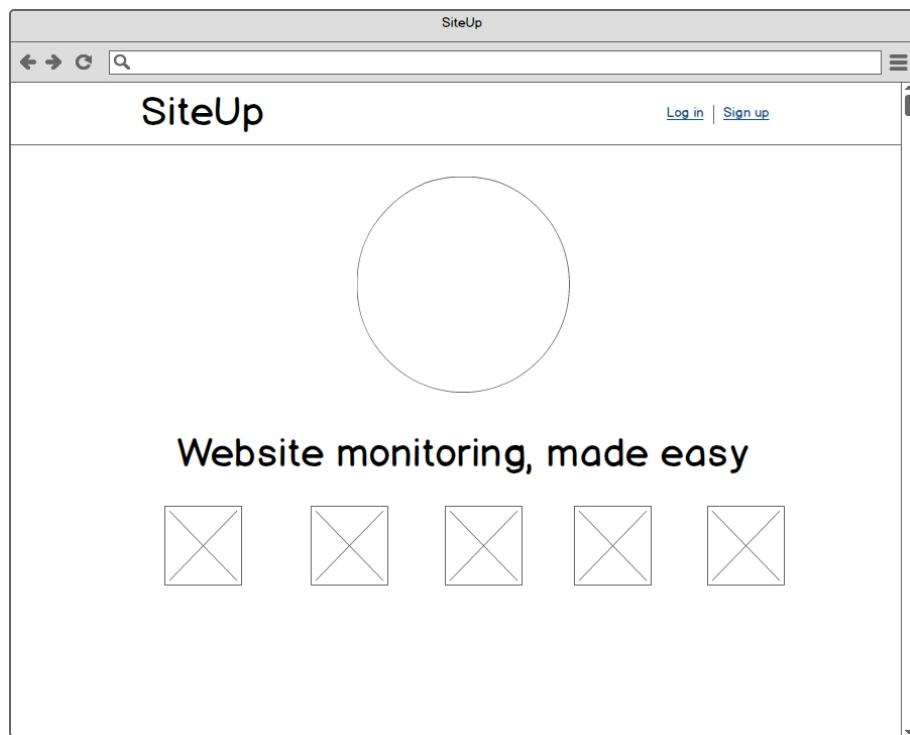


Figura 4.8: Home

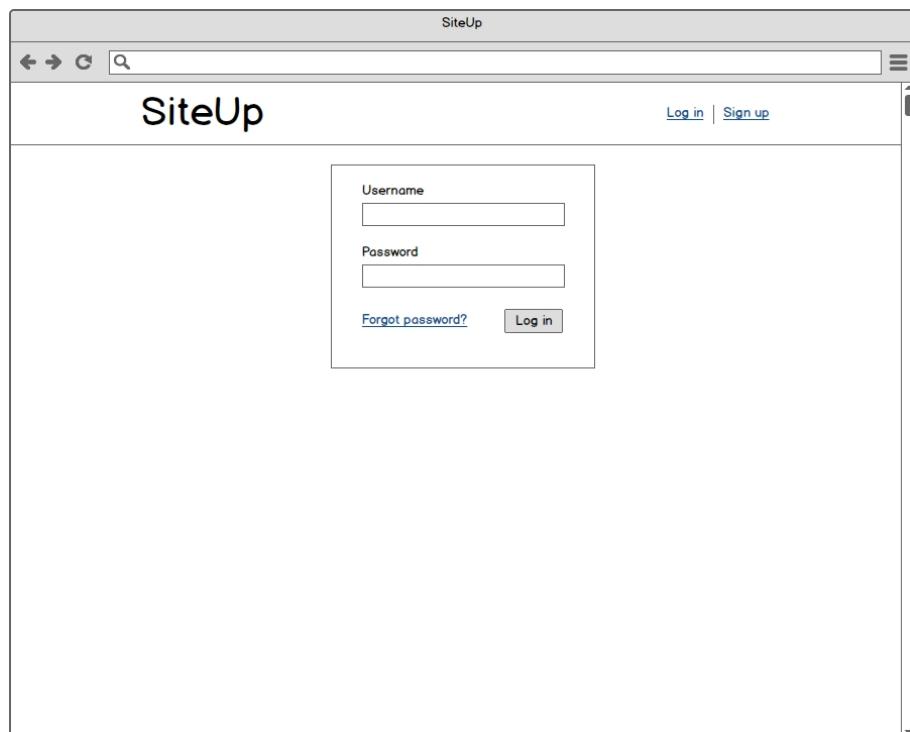


Figura 4.9: Login

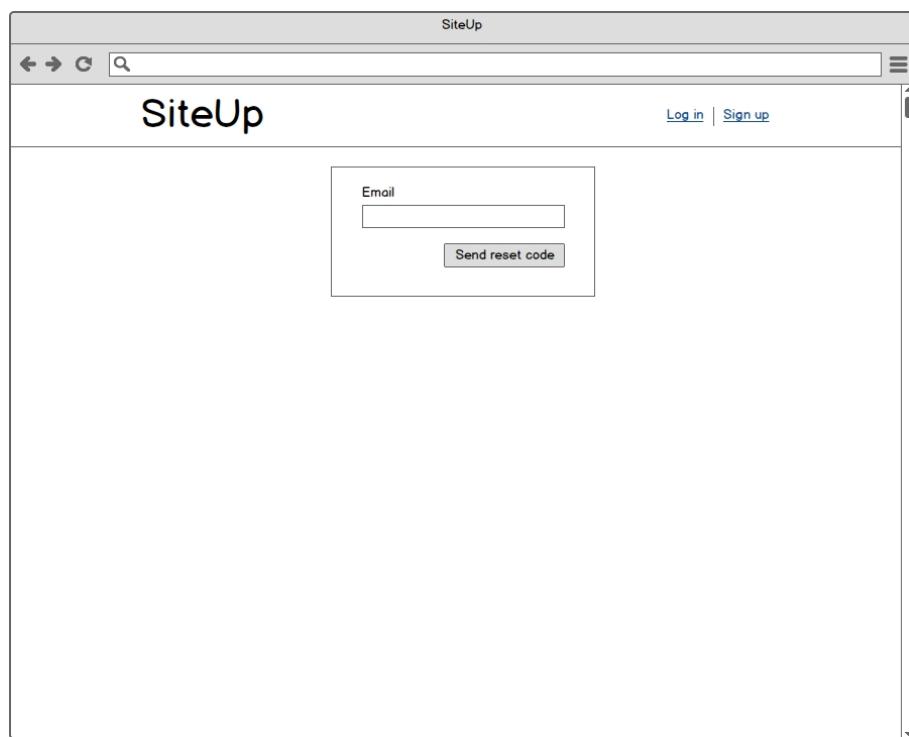


Figura 4.10: Recuperación de contraseña

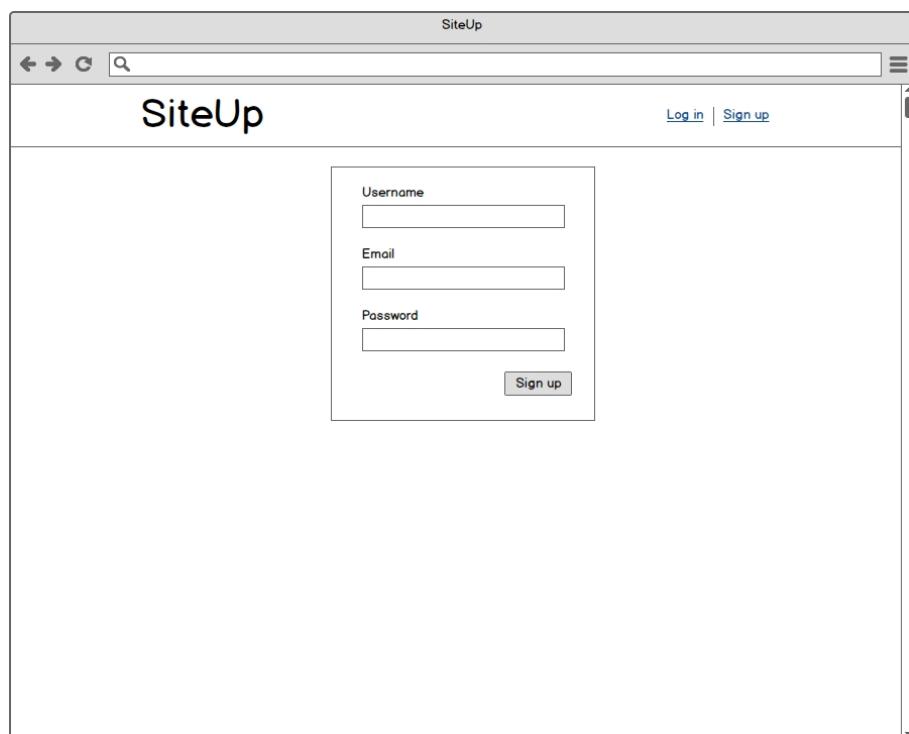


Figura 4.11: Registro de usuario

The screenshot shows the SiteUp dashboard. At the top, there's a header with a search bar and navigation links for 'Dashboard', 'Your profile', and 'Log out'. Below the header, it says 'You have N groups' and has a 'Add group' button. There are two sections, each labeled 'Name of group 1' and 'Name of group 2'. Each section contains two check entries, each with an upward arrow icon, a name ('Name of check'), a description ('Description of check. Lorem ipsum dolor sit amet.'), and four buttons: 'Details', 'Edit', 'Delete', and 'Disable'. To the right of each entry is a small graph icon.

Figura 4.12: Dashboard

This screenshot shows a detailed view of a specific check from the dashboard. It has a header with a search bar and navigation links. The main content area starts with a 'Name of check' section with an upward arrow icon, a name ('Name of check'), a description ('Description of check. Lorem ipsum dolor sit amet.'), and four buttons: 'Details', 'Edit', 'Delete', and 'Disable'. Below this is a chart titled 'Last 24 hours' (with 'Last week' as an option) showing '100% uptime' with a graph icon. At the bottom is a 'Last events' section with a table:

Status	Duration	Start date	End date	Extra
Up	Current	2014/01/01	-	Received Status 200
Down	37 minutes	2013/12/20	2014/01/01	Error in the request

Figura 4.13: Detalle de chequeo

The screenshot shows a web browser window with the title 'SiteUp' at the top. The main content area displays a form titled 'Check group title' with a text input field and a 'Create check group' button below it. At the top right of the page, there are links for 'Dashboard', 'Your profile', and 'Log out'. The overall layout is clean and modern.

Figura 4.14: Creación de grupo de chequeos

The screenshot shows a web browser window with the title 'SiteUp' at the top. The main content area displays a more complex form with fields for 'Title' (text input), 'Description' (text area), 'Target' (text input), and two checkboxes for 'Notify via email' and 'Notify via Android'. Below these is a 'Minutes between checks' input field containing the value '3' with a dropdown arrow, and a 'Create check' button at the bottom. Like Figure 4.14, it includes navigation links at the top right.

Figura 4.15: Formulario genérico de creación de chequeo

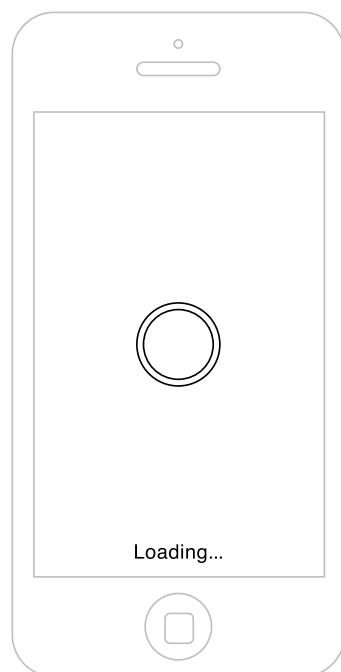


Figura 4.16: Pantalla de carga



Figura 4.17: Pantalla de login

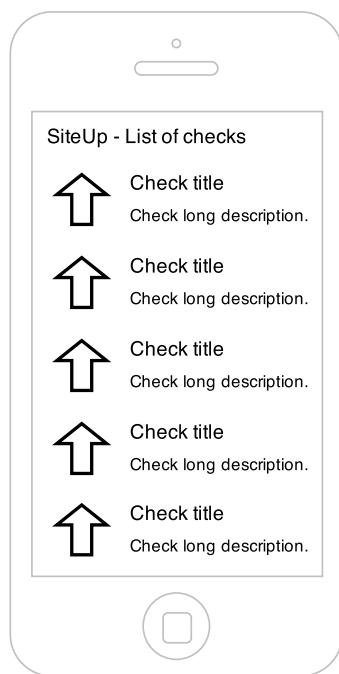


Figura 4.18: Pantalla de listado de chequeos

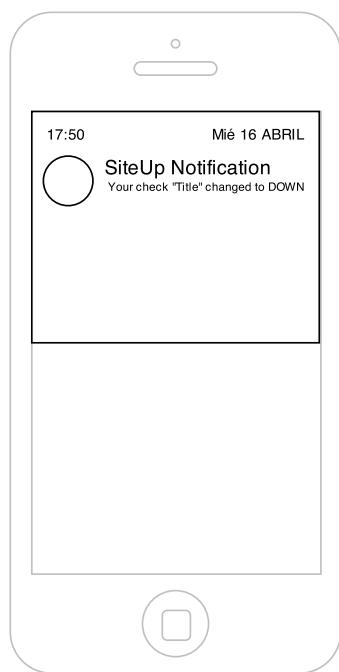


Figura 4.19: Pantalla de notificación



# Capítulo 5

## Diseño

En este capítulo se presentan los detalles de diseño del proyecto, basándonos en el análisis mostrado en los anteriores apartados. Se detallan la arquitectura general y el diseño físico de datos entre otros aspectos.

### 5.1. Arquitectura del sistema

#### 5.1.1. Arquitectura física

En este apartado, describimos los principales componentes hardware que forman la arquitectura física de nuestro sistema, recogiendo por un lado los componentes del servidor de producción y, por otro lado, los componentes del cliente de acceso.

##### Servidor de producción

**Hardware** El hardware mínimo indispensable para la correcta ejecución del motor del proyecto se detalla en la siguiente lista:

- 512MiB de memoria RAM como mínimo.
- 10GiB de disco duro como mínimo.
- Acceso a internet con un canal de subida de al menos 1Mbit/s.

**Software** En cuanto al software necesario para la ejecución del proyecto, se detallan los siguientes elementos, que es necesario instalar para desplegar el sistema:

- Sistema operativo **GNU/Linux**, preferiblemente basado en paquetería Debian.
- Código fuente del proyecto **SiteUp**.

- Servidor de shell remota **SSH**, accesible desde el exterior.
- **Nginx**, servidor web trabajando en modo de proxy inverso.
- **Supervisord**, sistema para el control de procesos que trabajen en modo demonio..
- **RabbitMQ**, cola de tareas sencilla que cumple el estandar AMQP.
- Intérprete de **Python**, versión mínima 2.7.
- Soporte de entornos virtuales **VirtualEnv** para la encapsulación de dependencias.

### Cliente de acceso web

El requisito no funcional NRQ-4, presente en el cuadro 3.2.3 indica que la plataforma web deberá ser accesible desde cualquier clase de dispositivo con acceso a internet. Así pues, la única restricción disponible para los clientes es que cuenten con un navegador modern que provea de acceso web y sea compatible con los últimos estándares web.

### Cliente de acceso Android

Los clientes que quieran acceder mediante la aplicación Android deberán contar con un dispositivo que soporte como mínimo la versión 2.3 del sistema operativo Android. Además, estos dispositivos deberán tener acceso a internet en general, y a los Google Play Services en particular.

#### 5.1.2. Arquitectura lógica

La arquitectura lógica del sistema está formada por los elementos software (servicios, aplicaciones, librerías, frameworks, etc.) que componen el software base, más el software desarrollado para cumplir los requisitos de la aplicación. En esta sección se muestra la organización de los distintos elementos software que componen el proyecto así como la comunicación entre ellos.

#### Plataforma web

En la figura 5.1 se puede ver un esquema de cómo se comunican los diferentes elementos que conforma el sistema web a alto nivel. En detalle, el flujo que sigue en cada capa es el siguiente.

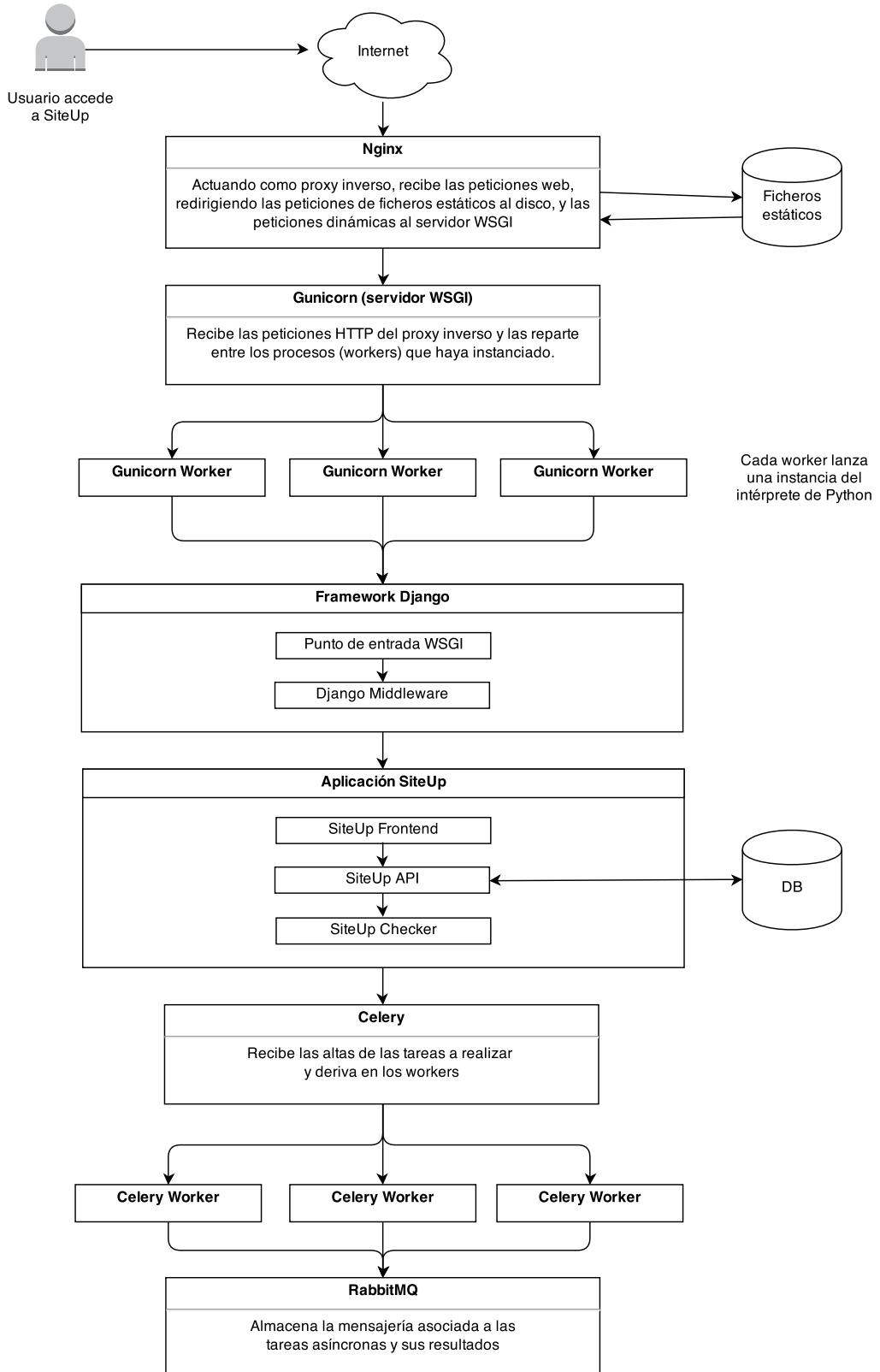


Figura 5.1: Arquitectura lógica del sistema web

**Navegador** Cuando un usuario desea acceder a SiteUp, abre un cliente web (habitualmente un **navegador** web) y teclea la URL. A partir de ahí, el navegador debe obtener el contenido de la web. Para ello (ignorando cuestiones de bajo nivel, como el envío de peticiones ARP o resoluciones de DNS) la acción fundamental es hacer una **petición HTTP** al servidor que se encuentra en la URL indicada. Esto se traduce en abrir una conexión TCP al puerto 80 del servidor remoto y enviar unas cabeceras en las que se indique, entre otras cosas, qué datos se quiere obtener.

**Servidor proxy inverso** En el servidor debe existir algún **servicio** que sea capaz de escuchar y disponer peticiones en el puerto 80 del servidor (el número el puerto puede variar). En nuestro caso el servidor deberá tener instalado **Nginx**, un servidor web/proxy inverso que recibirá las peticiones HTTP del exterior. En estas peticiones, enviadas por el navegador, se detalla qué recurso se necesita enviar.

En el caso de recursos estáticos, como por ejemplo ficheros de imágenes o archivos de hojas de estilo en cascada (CSS (Cascading Style Sheet(s))), es buena práctica que sea el propio servidor el que sirva estos ficheros, dado que no necesitan de ningún procesado dinámico, por lo que no tiene sentido desperdiciar recursos pasando la petición al código Python.

En el caso de recursos dinámicos, esto es, las peticiones web a páginas generadas dinámicamente, Nginx actuará de proxy inverso y pasará la petición al manejador que se haya configurado.

**Servidor de aplicaciones Python** Como se ha comentado, el proxy inverso ha detectado que hay una petición dinámica y según su configuración la ha pasado al servidor de aplicaciones Python. En el caso de nuestro proyecto se tratará del servidor **Gunicorn**. Gunicorn sigue un modelo conocido como *pre-fork worker*, que básicamente consiste en instanciar (*forkear*) varios procesos al lanzar el servidor que se encargarán de procesar las peticiones de forma paralela, según el proceso padre las vaya repartiendo.

En el mundo del desarrollo web en Python se ha establecido un estándar, conocido como WSGI (Web Server Gateway Interface), que regula la forma en la que las aplicaciones web escritas en Python se comunican con un servidor web. Por ello, cuando un worker recibe una petición lo primero que hace es comunicarse con la interfaz WSGI de la aplicación que se vaya a ejecutar, pasándole la petición y un puntero a un *callback* al que informar cuando la respuesta a la petición esté lista.

**Punto de entrada WSGI** Como se ha comentado, el worker del servidor de aplicaciones se comunica con la interfaz WSGI. Todas las aplicaciones desarrolladas en Django cuentan con un módulo `wsgi.py` que actúa como punto de entrada WSGI y que directamente pasa la petición al *middleware* de Django que empieza el procesamiento.

**Middleware Django y aplicación** A partir de este punto, la petición va rebotando entre código propio del framework Django y código de la aplicación. El flujo habitual, en su forma más básica es el siguiente:

1. Los módulos de *middleware* que gestionan las peticiones reciben la petición actual.
2. Django revisa el fichero `urls.py` de la aplicación, que contiene una lista de URLs asociadas a funciones. Django compara la URL de la petición recibida y mira si coincide con alguna de las URLs del proyecto.
3. Si no hay coincidencia, Django devuelve una respuesta negativa, normalmente en forma de código de estado 404.
4. Si hay coincidencia, Django llama a la función asociada a esa URL – conocida como la **vista**. La vista procesa la petición, devolviendo una respuesta.
5. Esa respuesta hace el camino de forma inversa, pasando por todos los pasos hasta el cliente.

Entrando más en detalle, una aplicación de Django se organiza en una arquitectura de módulos pequeños o *aplicaciones*, cada una de las cuales debe tener una serire de responsabilidades reducida y acotada, de forma que sea relativamente sencillo intercambiar las aplicaciones por otras. En el caso del proyecto SiteUp, se cuenta con tres aplicaciones:

- **siteup\_frontend** gestiona las vistas de la aplicación, recibe las peticiones, muestra las páginas y gestiona formularios.
- **siteup\_api** recibe las órdenes de `siteup_frontend`, haciendo los cambios necesarios en la base de datos, dando de alta las tareas y procesando resultados.
- **siteup\_checker** cuenta con el código de bajo nivel para hacer los chequeos de diferentes tipos. Es principalmente una aplicación tipo *helper*.

Cada aplicación en Django cuenta con una arquitectura de tres capas conocida como **Modelo-Vista-Plantilla**, que se asimila al clásico patrón del **Modelo-Vista-Controlador**.

- El **modelo** representa la información en la base de datos, evitando al usuario tener que utilizar código SQL. Para ello, Django cuenta con un ORM que facilita el trabajo con registros de la base de datos.
- Las **vistas** suelen ser funciones asociadas a una URL. La filosofía de Django es que las vistas sirvan solo como *punto de paso* de los datos entre los modelos y las plantillas. Es mala práctica colocar lógica de negocio compleja en las vistas.
- Las **plantillas** son la representación final de los datos, lo que acaba viendo el usuario. Las plantillas suelen estar escritas en HTML aunque pueden también representarse usando JSON, XML, etc.

La organización física de ficheros de un proyecto Django se ajusta perfectamente a la organización lógica hasta ahora descrita, siendo habitual tener un árbol de ficheros similar al siguiente:

- Raíz del proyecto
- siteup – Directorio de proyecto general.
  - settings.py – Configuración general.
  - urls.py – Configuración de URLs.
  - wsgi.py – Punto de acceso WSGI.
- siteup\_api – Directorio para la app (suele haber más de uno).
  - views.py – Definición de vistas (funciones).
  - models.py – Definición de modelos.
  - tests.py – Definición de tests.

Opcionalmente, las aplicaciones de Django pueden contar con ficheros individuales para otros elementos, como formularios, manejadores, migraciones, validadores, utilidades, filtros, procesadores de contexto, etcétera.

**Servidor de tareas asíncronas** Habrá peticiones en SiteUp cuyo objetivo sea dar de alta chequeos. Éstos dan lugar a nuevas tareas que serán ejecutadas asíncronamente. Celery, el servidor de tareas que se utiliza, cuenta con una arquitectura de *workers* similar a Gunicorn, que se reparten las tareas que son dadas de alta por SiteUp. Estas tareas se ejecutan de forma periódica.

Tras su conclusión, Celery se comunica con Django para dejar constancia de los resultados de esas tareas en forma de modelos de la base de datos.

**Broker de mensajes** Por debajo de Celery se encuentra el *broker* de mensajes, que implementa el ya mencionado estándar AMQP. El broker se encarga de almacenar las tareas y sus resultados en forma de mensajes en una cola.

En este proceso es necesario *traducir* las tareas a un formato que sea almacenable y gestionable en una cola. En el caso de Django lo más habitual es utilizar Pickle, el estándar de facto para la serialización de objetos Python.

## Aplicación Android

En la figura 5.2 se puede ver un esquema general de la arquitectura de la aplicación móvil desarrollada para el sistema operativo Android. El flujo se detalla a continuación:

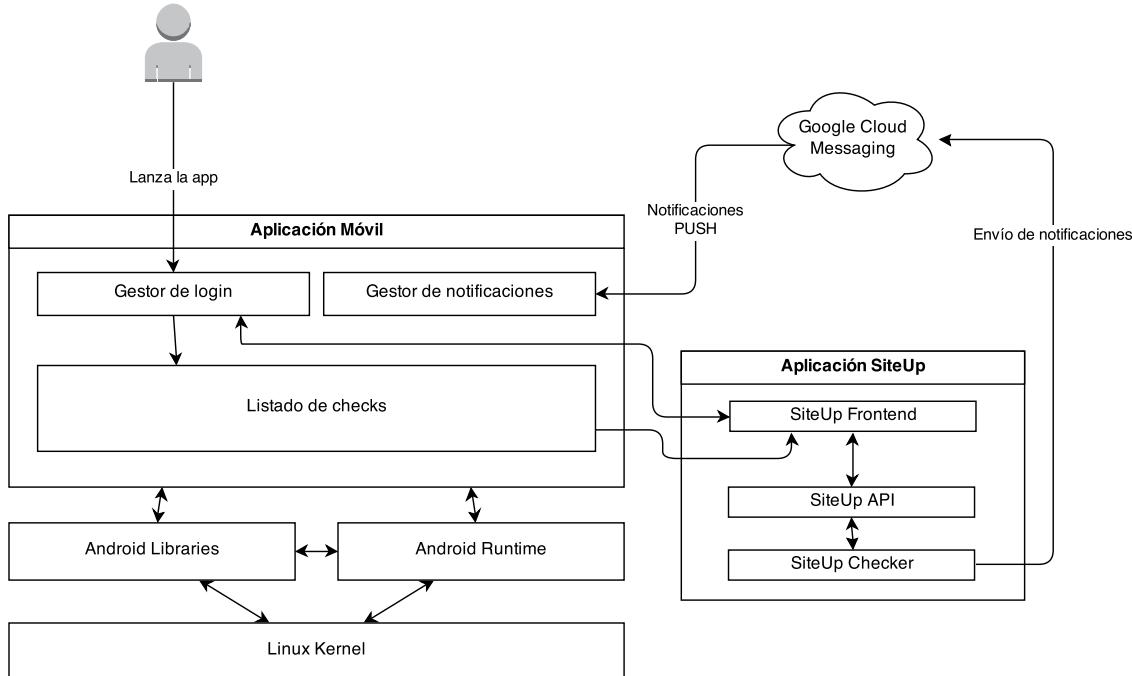


Figura 5.2: Arquitectura lógica de la aplicación Android

**Lanzador** Partimos de la premisa de que el usuario ha instalado la aplicación *SiteUp Client* en su dispositivo. Si el usuario decide lanzar la aplicación, deberá dirigirse al lanzador de aplicaciones de su dispositivo y pulsar sobre el icono de la aplicación, con lo que se iniciará la ejecución.

**Gestor de inicio de sesión** Cuando la aplicación Android se inicia, se lanza una actividad *LoginActivity* que hace varias cosas. En primer lugar, verifica si hay datos de inicio de sesión guardados en el dispositivo. Si no los hay, muestra un formulario para que el usuario introduzca su nombre de usuario y contraseña.

Una vez obtenidos los datos, la aplicación móvil se comunica con la plataforma web para verificar el inicio de sesión. En caso correcto, la plataforma web envía la información de los chequeos a la aplicación móvil.

**Listado de chequeos** En la aplicación móvil se ha dispuesto una actividad *CheckListActivity* que mostrará la lista de chequeos que el usuario ha dado de alta, con información básica de cada uno de ellos. Cuando el usuario pulse en cualquiera de los chequeos, se abrirá un navegador y será redirigido a la plataforma web, particularmente a la página de detalle del chequeo seleccionado.

**Gestor de notificaciones** La aplicación móvil cuenta con un servicio capaz de recibir notificaciones *Push* del servicio *Google Cloud Messaging*. Cuando se recibe un mensaje, se muestra una notificación al usuario, informándole también de

forma audible y a través de vibración. La notificación aparece en el área de notificaciones del sistema operativo. Si el usuario pulsa en la notificación podrá obtener información sobre el chequeo involucrado.

**Google Cloud Messaging** GCM [21] es un servicio para Android que permite enviar datos desde un servidor a los dispositivos Android que tengan instaladas una aplicación en particular. El servicio GCM se encarga de todos los aspectos de la entrega de mensajes. Se trata de un servicio gratuito independientemente del número de mensajes y el tamaño de éstos.

GCM recibirá las notificaciones del *backend* de la plataforma móvil, y se encargará de reenviarlas a los dispositivos Android.

**SiteUp Checker** Dentro de la aplicación móvil, el módulo `siteup_checker` es el encargado de realizar los chequeos, y también de enviar las notificaciones a los dispositivos Android a través del servicio GCM. La plataforma web tiene una clave especial con la que se realizan las peticiones a los servidores de Google, enviando los mensajes.

**Runtime y bibliotecas de Android** Las aplicaciones Android utilizan como base el runtime y las bibliotecas que provee el sistema operativo para acceder a las funciones del dispositivo y demás servicios, como la comunicación por red, acceso a datos del usuario, lectura y escritura de ficheros en el dispositivo, etcétera.

## 5.2. Diseño físico de datos

En la figura 5.3 se muestra el diagrama de modelos que se reflejará en la base de datos, donde se puede apreciar cada una de las tablas que forman el proyecto, así como los campos de las tablas y las relaciones entre ellas.

El diseño de los datos se ha hecho mediante las herramientas de mapeo objeto-relacional que ofrece el framework Django. A continuación se detallan los principales modelos con los que cuenta el sistema.

### 5.2.1. User

Este modelo representa a un usuario registrado en el sistema. Es un modelo pre-determinado de Django y forma parte de la aplicación `django.contrib.auth`.

Sus principales campos son:

- `username` - Nombre de usuario.

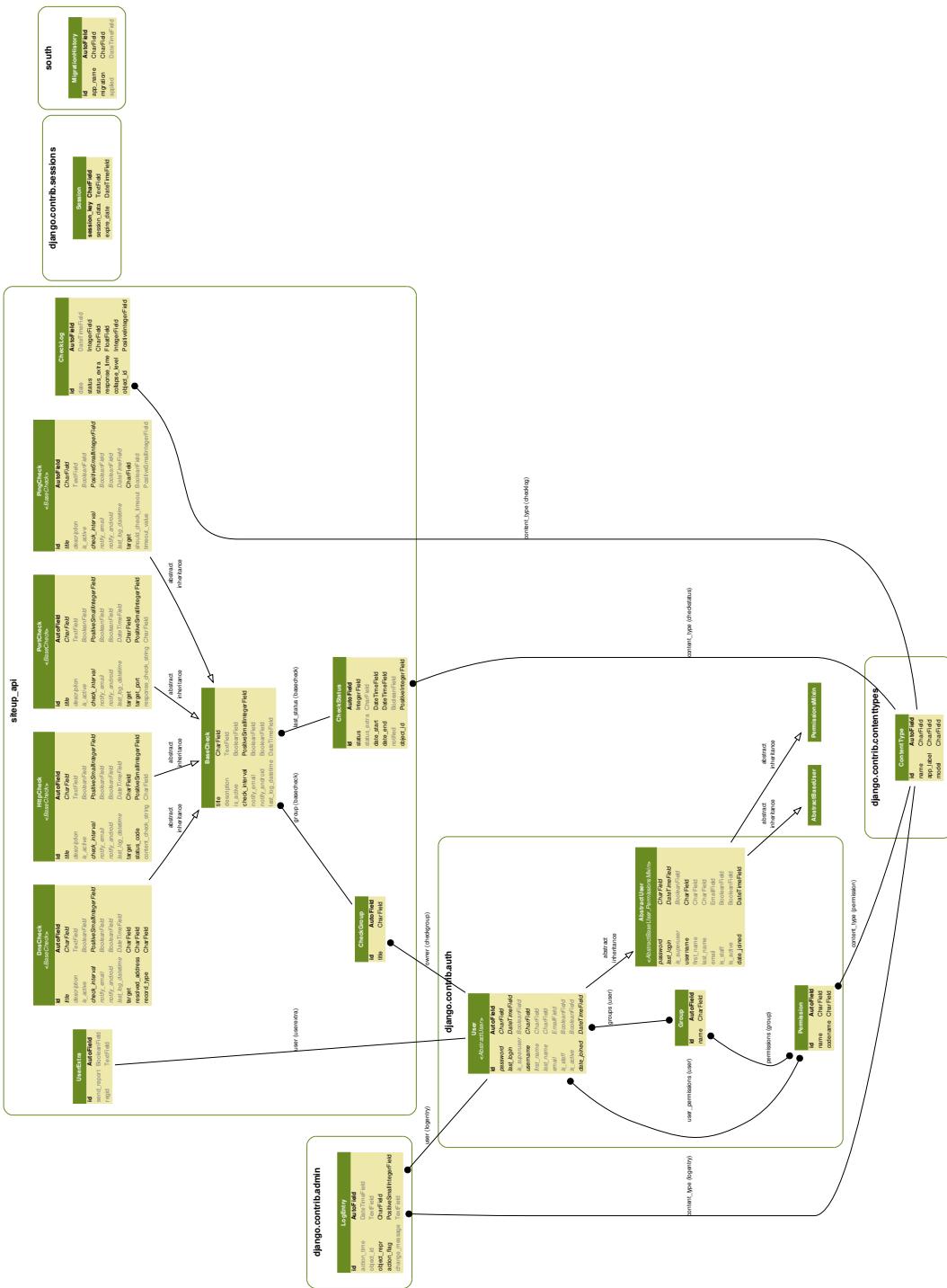


Figura 5.3: Diseño de la base de datos

- password - Contraseña. Se almacena en forma de cifrado irreversible de acuerdo a las exigencias de la actual Ley de Protección de Datos.
- email - Dirección de correo electrónico.

### 5.2.2. UserExtra

Este modelo guarda información adicional sobre los usuarios, evitando así tener que modificar el modelo que integra Django. Forma parte de la aplicación siteup\_api.

Sus principales campos son:

- send\_report - Booleano, indica si el usuario recibirá reportes diarios sobre el estado de sus cheques.
- regid - Texto, guarda el identificador único del dispositivo Android obtenido a través de Google Play Services.

### 5.2.3. CheckGroup

Parte de la aplicación siteup\_api. Identifica a un grupo de cheques. Todos los cheques deben pertenecer a un grupo.

El campo principal de este modelo es title, que guarda una cadena con el título del grupo.

### 5.2.4. BaseCheck

Parte de la aplicación siteup\_api. Este modelo **abstracto** guarda campos comunes a todos los tipos de cheques. No tiene representación directa en la base de datos, sino que sirve para evitar duplicidad de código.

Sus principales campos son:

- title - Texto, representa el título del chequeo.
- description - Texto, representa la descripción del chequeo.
- is\_active - Booleano, indica si el chequeo está activo.
- check\_interval - Entero, indica el intervalo entre chequeos en minutos.
- notify\_email - Booleano, indica si los cambios de estado del chequeo se notificarán vía correo electrónico.
- notify\_android - Booleano, indica si los cambios de estado del chequeo se notificarán vía Android utilizando notificaciones push.

- `last_log_datetime` - Fecha, almacena la fecha de la última vez que se lanzó el chequeo.

### 5.2.5. DnsCheck

Parte de la aplicación siteup\_api. Este modelo deriva de BaseCheck y representa un chequeo de registros DNS. Sus principales campos son:

- `target` - Texto, indica el dominio a chequear.
- `record_type` - Enum, indica el tipo de registro a chequear. Puede ser de tipo A, AAAA, CNAME, MX y TXT.
- `resolved_address` - Texto, indica el contenido que debe tener el registro revisado.

### 5.2.6. HttpCheck

Parte de la aplicación siteup\_api. Este modelo deriva de BaseCheck y representa un chequeo a través de peticiones HTTP.

Sus principales campos son:

- `target` - Texto, indica la URL a comprobar.
- `status_code` - Entero, indica el código de estado que se debe recibir.
- `content_check_string` - Cadena, representa una cadena de texto que, opcionalmente, debe estar presente en la respuesta obtenida desde la URL indicada.

### 5.2.7. PortCheck

Parte de la aplicación siteup\_api. Este modelo deriva de BaseCheck y representa un chequeo de puertos remotos.

Sus principales campos son:

- `target` - Cadena, identifica el servidor remoto al que conectarse.
- `target_port` - Entero, indica el puerto remoto al que conectarse.
- `response_check_string` - Cadena, representa una cadena de texto que, opcionalmente, debe estar presente en la respuesta obtenida desde el servidor.

### 5.2.8. PingCheck

Parte de la aplicación siteup\_api. Este modelo deriva de BaseCheck y representa un chequeo mediante el envío de paquetes ICMP.

Sus principales campos son:

- target - Cadena, identifica al servidor remoto que hay que chequear.
- should\_check\_timeout - Booleano, indica si hay que comprobar el tiempo de respuesta de los paquetes ping.
- timeout\_value - Entero, si el anterior campo evalúa a *Verdadero*, este campo guarda el tiempo de respuesta máximo permitido.

### 5.2.9. CheckLog

Parte de la aplicación siteup\_api. Es un registro (*log*) que representa el resultado de ejecutar un chequeo.

Sus principales campos son:

- date - Fecha, representa el momento en el que se obtuvo el resultado.
- status - Entero, representa el estado del chequeo que se ha obtenido. Los posibles valores son:
  - 0: el chequeo ha terminado correctamente y el estado es correcto (*Up*).
  - 1: el chequeo ha terminado correctamente y el estado es negativo (*Down*).
  - 2: el chequeo no ha podido concluirse. Equivalente en estadísticas al estado negativo. (*Error*).
- response\_time - Entero, solo válido para los chequeos de tipo Ping. Guarda el tiempo de respuesta obtenido.
- collapse\_level - Entero. Indica el nivel de *colapsado* del registro. Dado que el número de registros es muy alto (en una hora pueden llegar a generarse más de 3600 registros por chequeo), es necesario resumir los registros más antiguos. Este campo indica si el *CheckLog* ya ha sido resumido o no.

### 5.2.10. CheckStatus

Parte de la aplicación siteup\_api. Representa un cambio de estado de un chequeo. Cuando un chequeo es ejecutado, se genera una instancia de *CheckLog* y se comprueba el estado de éste con el estado de la última comprobación. Si se verifica que el estado ha cambiado con respecto a los últimos estados, se genera una instancia de *CheckStatus* y se lanzan las notificaciones pertinentes.

Los campos principales de este modelo son:

- status - Entero, indica el estado. El código numérico responde al mismo formato que se sigue en el modelo *CheckLog*.
- status\_extra - Cadena, guarda información adicional sobre el estado.
- date\_start - Fecha, indica el momento en el que el chequeo cambia a este estado.
- date\_end - Fecha, indica el momento en el que el chequeo deja de estar en este estado. Si es el estado actual de un chequeo, se mantiene en blanco.
- notified - Booleano, indica si este cambio de estado ha sido ya notificado.

### 5.2.11. Session

Parte de la aplicación `django.contrib.sessions`. Representa una sesión de usuario en la aplicación web. Sus campos principales son:

- session\_key - Cadena, guarda el identificador de la sesión.
- session\_data - Texto, guarda los datos asociados a la sesión.
- expire\_date - Fecha, indica hasta qué fecha son válidos los datos de la sesión.

### 5.2.12. MigrationHistory

Parte de la aplicación `south`. Representa una migración de la base de datos. Los campos principales son:

- app\_name - Cadena, nombre de la app sobre la que se hace la migración.
- migration - Cadena, identificador de la migración.
- applied - Fecha, indica cuándo se aplicó la migración.

### 5.2.13. ContentType

Parte de la aplicación `django.contrib.contenttypes`. Es el bloque principal del *Content Types Framework*, un framework integrado en Django que permite trabajar de forma genérica con los modelos, ofreciendo funcionalidades como la posibilidad de crear claves foráneas genéricas (es decir, que puedan apuntar a cualquier modelo).

Una instancia de *ContentType* representa un modelo particular de una aplicación en concreto. Los campos que contiene son:

- name - Cadena, nombre del modelo en formato legible.
- model - Cadena, nombre original del modelo.
- app\_label - Cadena, nombre de la aplicación a la que pertenece el módulo.

## 5.3. Diseño de la imagen corporativa

La **imagen corporativa** del proyecto comprende el diseño del **logotipo**, la elección de la paleta de **colores** y las **tipografías**, que se usan en el diseño de la web, emails y cualquier otro elemento visual.

Para su elaboración se han seguido ciertas premisas o *guías de branding*, de forma que se consiguiese el look más adecuado:

- **Diseño minimalista.** En SiteUp, la información es lo más importante, así que no tiene sentido utilizar un diseño muy sobrecargado.
- **Colores monocromáticos.** Siguiendo en la línea del punto anterior, esta *regla* facilita que haya una consistencia en las interfaces de la web y la aplicación móvil sin tener que recurrir a complejas combinaciones de color.
- **Tipografías libres.** Era importante no utilizar tipografías que requiriesen una licencia comercial para su uso.

### 5.3.1. Diseño del logotipo

Partiendo del nombre del proyecto se pensaron varios bocetos para el logotipo. La idea principal era jugar con las dos palabras que componen el nombre: *Site* y *Up*. Para la primera palabra, la metáfora de usar una ventana de un navegador o algo que representase un servidor era demasiado compleja, así que se decidió simplemente usar la letra S.

Por otro lado, para la segunda palabra estaba claro desde el principio que se usaría alguna metáfora de tipo flecha para representar la dirección *arriba*. Se probaron distintos tipos de flechas y al final se decidió por una doble flecha de tipo galón, con suficiente separación para que se viese en bajas resoluciones. El logotipo resultante se puede ver en la figura 5.4



Figura 5.4: Logotipo de SiteUp

### 5.3.2. Elección de la gama cromática

Teniendo en mente la idea apuntada anteriormente de que la gama de colores debería ser preferiblemente monocromática, se empezó a buscar un tono que funcionase con la temática del sitio. Se hizo una búsqueda de sitios web de servicios de soporte, y en la mayoría de ellos la gama cromática era siempre muy neutra, con tonos poco saturados y casi siempre fríos.



Figura 5.5: Detalle de la paleta de colores de SiteUp

Así, tras varias pruebas, se decidió optar por una gama de azules poco saturados, rozando el grisáceo. En la figura 5.5 se detalla la paleta de colores principal.

### 5.3.3. Elección de la tipografía

Siguiendo las directrices apuntadas previamente, se buscaba una tipografía sencilla y minimalista. Se probaron algunas alternativas comerciales, empezando, cómo

no, por Helvética y pasando por Myriad Pro, Próxima Nova y algunas otras. Al final nos decantamos por utilizar **Open Sans** [32], una tipografía sans-serif limpia y moderna, de corte humanista, y desarrollada por Steve Matteson para Google.

Se trata de una tipografía limpia, optimizada para la legibilidad en toda clase de soportes, y por supuesto libre, bajo una licencia Apache 2.0.



Figura 5.6: Muestra de Open Sans en diferentes pesos

## 5.4. Diseño de la interfaz de usuario de la plataforma web

En esta sección se detallarán las interfaces visuales de la plataforma web del proyecto, indicando qué acciones realizan cada uno de los componentes que las conforman.

### 5.4.1. Pantalla de inicio

En la figura 5.7 se ve el diseño de página de inicio, o *home*, de la plataforma web. Es la pantalla que aparece al acceder a la base del proyecto. Se componen de una barra superior de navegación en la que aparece el logotipo a la izquierda, y las opciones de navegación a la derecha. Al pulsar el logotipo, se redirige a la pantalla de inicio.

Si el usuario no está logueado, las opciones de navegación disponibles son:

- **Log in:** lleva al usuario a la pantalla de inicio de sesión.
- **Sign up:** lleva al usuario a la pantalla de registro de usuario.

Si el usuario sí está logueado, las opciones que aparecen son diferentes:

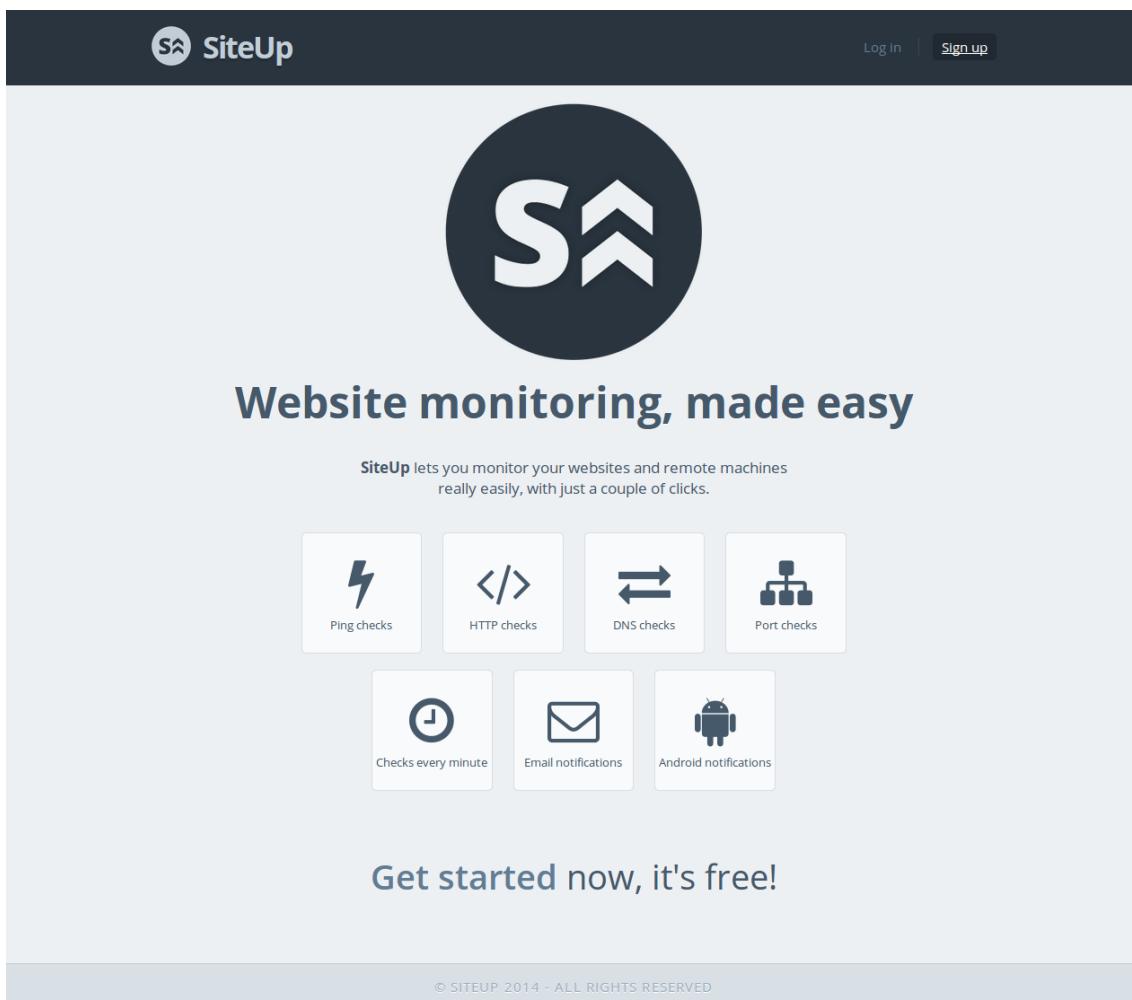


Figura 5.7: Pantalla de inicio

- **Dashboard:** lleva al usuario al listado de cheques.
- **Your profile:** lleva al usuario al formulario de datos personales.
- **Log out:** cierra la sesión del usuario.

La barra de navegación superior tiene el mismo comportamiento en todas las secciones de la web.

#### 5.4.2. Pantalla de inicio de sesión

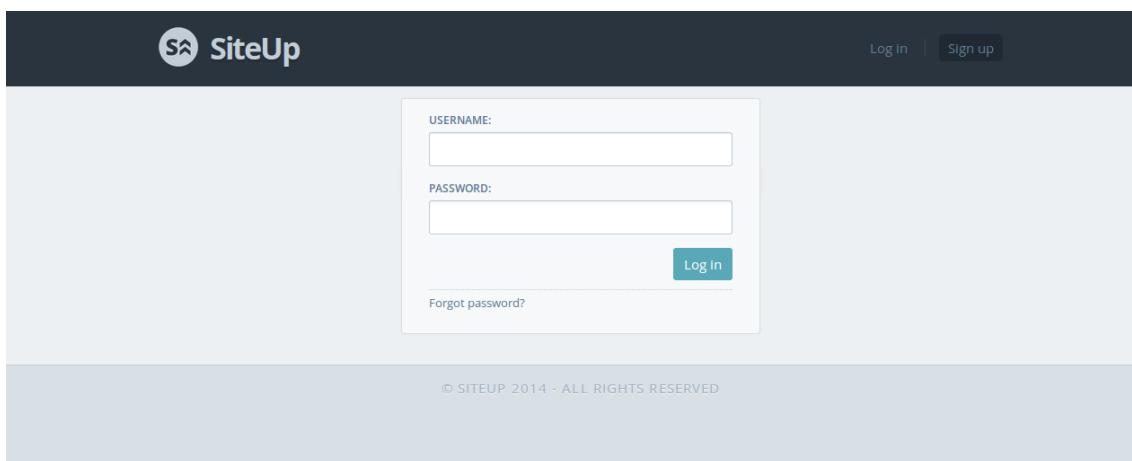


Figura 5.8: Pantalla de inicio de sesión

En la figura 5.8 se ve el diseño de la pantalla de inicio de sesión. Se presenta un formulario en el que el usuario deberá escribir su nombre de usuario y su contraseña, y tras ello pulsar el botón de *Login*. En caso de que los datos introducidos sean correctos, el sistema iniciará sesión y redirigirá al usuario a la lista de cheques. En caso contrario, volverá a aparecer el formulario con los errores que se hayan producido.

Opcionalmente, si el usuario no recuerda su contraseña puede pulsar en el botón de *Forgot password?*, lo que le llevará a la pantalla de recuperación de contraseña.

#### 5.4.3. Pantalla de recuperación de contraseña

En esta pantalla, visible en la figura 5.9, el usuario podrá introducir su correo electrónico para recuperar su contraseña. Tras llenar el campo de texto, deberá pulsar el botón. Si el email introducido es correcto, el sistema enviará un correo electrónico con instrucciones para reiniciar la contraseña, y en pantalla se mostrará un mensaje indicando que se ha iniciado el proceso de reinicio.

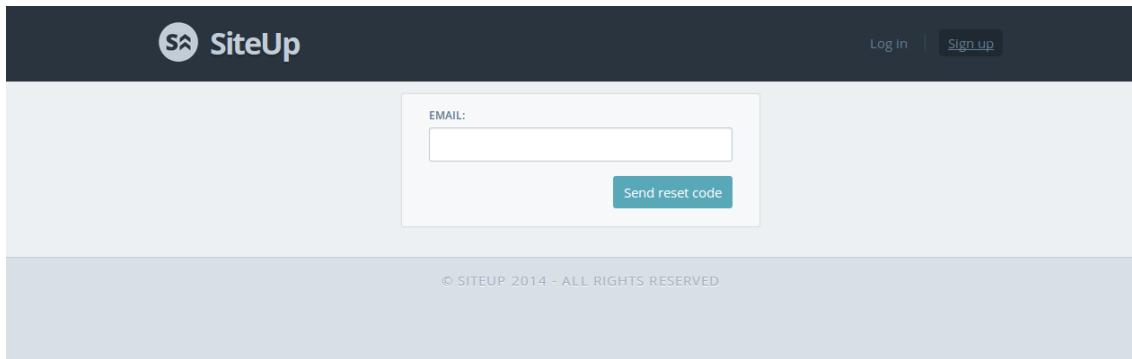


Figura 5.9: Pantalla de recuperación de contraseña

#### 5.4.4. Pantalla de registro de usuario

En la figura 5.10 se puede ver la pantalla de registro de usuario.

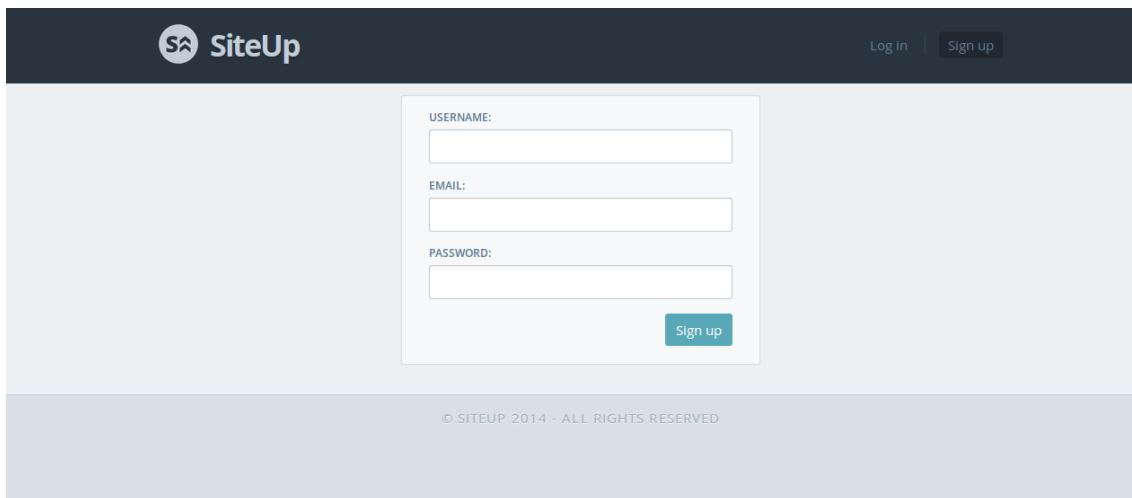


Figura 5.10: Pantalla de registro de usuario

Dispone de un formulario, en el que el usuario deberá introducir sus datos personales: nombre de usuario, dirección de correo electrónico y contraseña. Una vez llenados, deberá pulsar el botón. Si los datos son correctos, el sistema creará la cuenta de usuario. Si no, el sistema volverá a mostrar el formulario, resaltando los errores en los datos.

#### 5.4.5. Pantalla de lista de cheques

En la figura 5.11 se puede ver la pantalla principal de SiteUp, la lista de cheques o **dashboard**. En esta pantalla aparecen los grupos de cheques que tiene dados de alta un usuario, así como los cheques que pertenecen a cada grupo. Desde esta pantalla se pueden hacer un gran número de operaciones:

- Pulsando el botón *Add group* aparecerá la pantalla de creación de grupo.

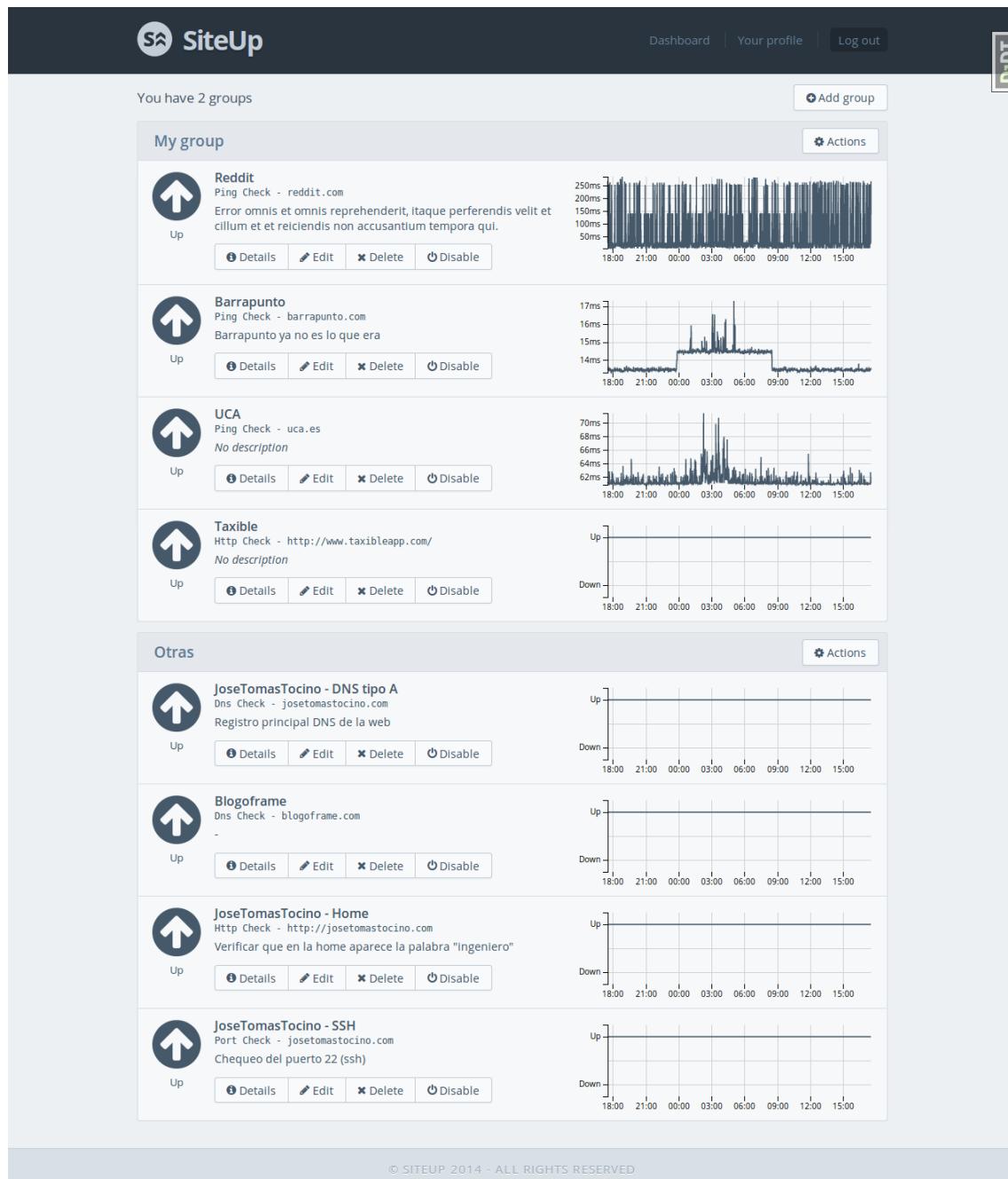


Figura 5.11: Pantalla de lista de chequeos

- Cada grupo tiene un menú desplegable que aparece al pulsar o colocar el ratón sobre el botón *Actions*. Desde ahí, es posible:
  - Añadir un chequeo al grupo pulsando el botón *Add check*.
  - Activar todos los chequeos del grupo pulsando el botón *Enable all checks*.
  - Desactivar todos los chequeos del grupo pulsando el botón *Disable all checks*.
  - Editar los detalles del grupo pulsando el botón *Edit*.
  - Borrar el grupo y los chequeos que contiene pulsando el botón *Delete*.
- Dentro de cada grupo aparece cada uno de los chequeos que contiene, con un ícono indicando el estado del chequeo, el título, descripción, una gráfica de su estado en las últimas 24 horas y una serie de botones de acción, que permiten:
  - Ver los detalles del chequeo pulsando el botón *Details*.
  - Editar las propiedades del chequeo con el botón *Edit*.
  - Borrar el chequeo con el botón *Delete*.
  - Activar o desactivar el chequeo pulsando *Enable* o *Disable*.

#### 5.4.6. Pantalla de creación de grupo

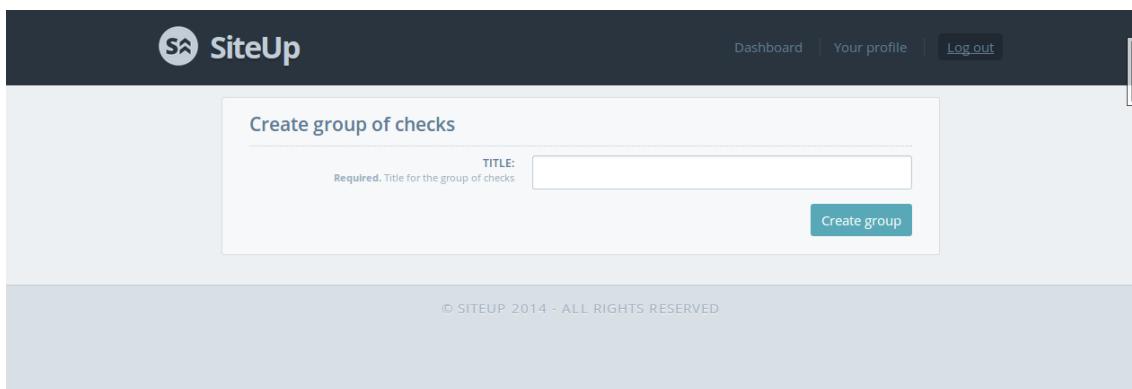


Figura 5.12: Pantalla de creación de grupo de chequeos

La figura 5.12 representa la pantalla para crear un grupo de chequeos. Muestra un formulario en el que hay que introducir los detalles del grupo, concretamente el título que lo describe.

Cuando el usuario introduzca el título y pulse el botón, el sistema revisará que los datos sean correctos. En caso afirmativo, el sistema creará el nuevo grupo. En caso negativo, se volverá a mostrar el formulario, indicando los fallos ocurridos.

#### 5.4.7. Pantallas de creación de chequeo

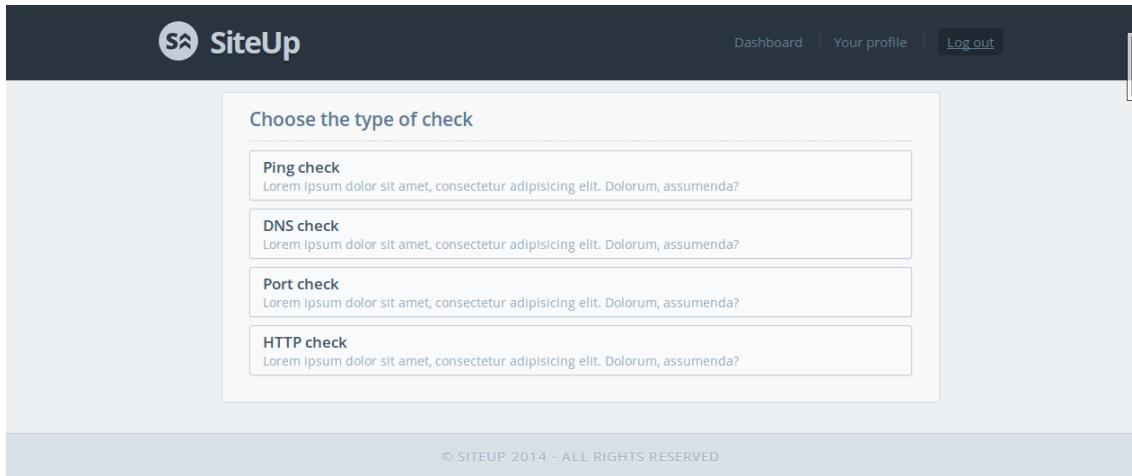


Figura 5.13: Pantalla de elección de tipo de chequeo

La creación de un chequeo se divide en dos pantallas principalmente. El primer paso es elegir el tipo de chequeo a crear, mediante los botones que aparecen en la pantalla de la figura 5.13.

Una vez elegido el tipo de chequeo a crear, aparecerá un formulario para introducir los detalles del chequeo, que variará según el tipo de chequeo elegido. En la figura 5.14 se puede ver el formulario que aparecerá a la hora de crear un chequeo de tipo DNS.

Cuando el usuario rellene el formulario y pulse el botón, se enviarán los datos y el sistema comprobará que la información es correcta. En tal caso, se creará el nuevo chequeo y se redirigirá al usuario al *dashboard*. En caso contrario, se volverá a mostrar el formulario, indicando los errores encontrados.

#### 5.4.8. Pantalla genérica de borrado

En la figura 5.15 se representa la pantalla de confirmación de borrado que aparece cuando se intenta borrar un chequeo o un grupo de chequeos. La pantalla es sencilla: pregunta al usuario si desea realmente borrar el elemento, ofreciendo dos botones, uno para confirmar el borrado y otro para cancelar la operación. En ambos casos, tras la interacción el sistema redirige al usuario al *dashboard*.

#### 5.4.9. Pantalla del perfil de usuario

En la figura 5.16 se muestra la pantalla para editar el perfil de usuario, a la que se llega pulsando el botón *Your profile* situado en la barra superior de navegación.

The screenshot shows a web application titled "SiteUp" with a dark header bar. The header includes the SiteUp logo, navigation links for "Dashboard", "Your profile", and "Log out", and a small user icon. The main content area has a white background and features a form titled "Create new Dns check". The form consists of several input fields with validation messages:

- TITLE:** Required. Short title for the check. (Input field)
- DESCRIPTION:** Larger description, you can include notes. (Input field)
- TARGET:** Required. Should be a hostname. (Input field)
- RECORD TYPE:** Required. Type of dns resource record. (Dropdown menu set to "A")
- RESOLVED ADDRESS:** Required. Should be a valid value for the selected record type. (Input field)
- CHECK INTERVAL:** Required. In minutes. How often the check should be triggered. (Input field containing "1")
- NOTIFY EMAIL:** (checkbox checked)
- NOTIFY ANDROID:** (checkbox checked)

At the bottom right of the form is a blue "Create check" button.

Figura 5.14: Pantalla de introducción de datos del chequeo

The screenshot shows a web application titled "SiteUp" with a dark header bar. The header includes the SiteUp logo, navigation links for "Dashboard", "Your profile", and "Log out", and a small user icon. The main content area has a white background and features a modal dialog box titled "Confirm deletion". The dialog contains the following text and buttons:

Are you sure you want to delete the check?  
Title: Reddit

[Go back](#) [Delete](#)

At the bottom right of the dialog is a blue "Delete" button.

Figura 5.15: Pantalla genérica de borrado

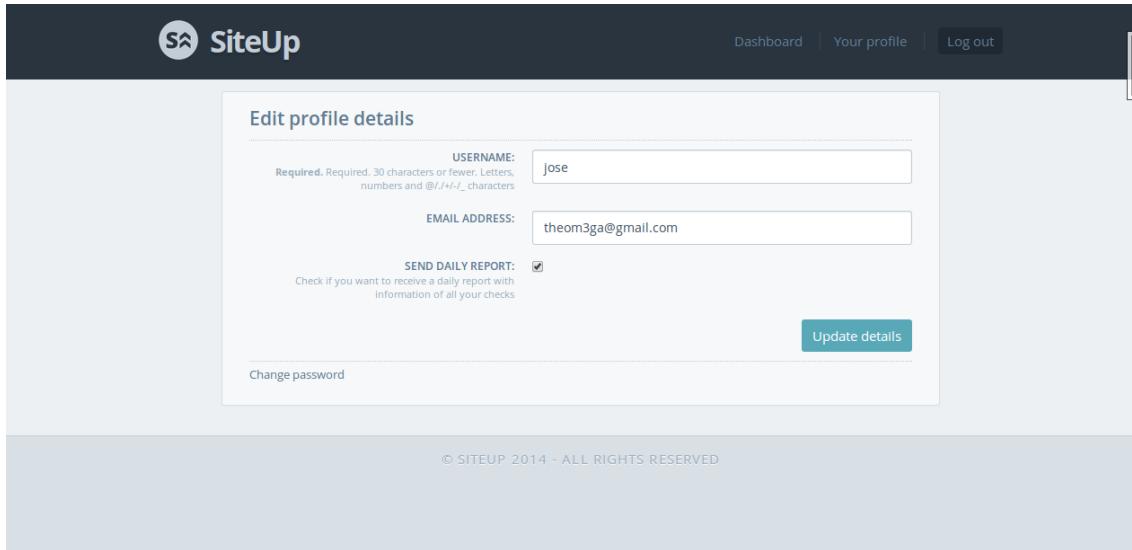


Figura 5.16: Pantalla del perfil de usuario

Desde esta pantalla el usuario puede modificar sus datos de usuario: nombre de usuario, correo electrónico y si desea recibir un reporte diario sobre el estado de sus chequeos.

También cuenta con la opción de cambiar la contraseña pulsando el botón *Change password*.

## 5.5. Diseño de la interfaz de usuario de la aplicación móvil

En esta sección se detallarán las interfaces visuales de la aplicación móvil para sistemas operativos Android, indicando la navegabilidad e interacción entre las pantallas.

### 5.5.1. Aspecto desde el lanzador

La figura 5.17 muestra el aspecto del ícono de la aplicación desde el lanzador de aplicaciones de Android.

Como se comentó en la sección 5.3.1, el logotipo fue diseñado pensando en su legibilidad en bajas resoluciones, por lo que el ícono de la aplicación es fácilmente identificable desde el lanzador.

Al pulsar el ícono de la aplicación, se lanza la pantalla de carga.

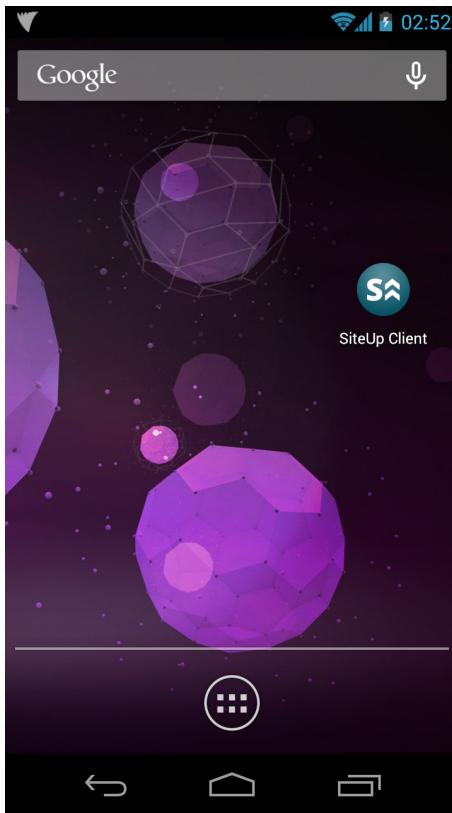


Figura 5.17: Icono de SiteUp en el lanzador

### 5.5.2. Pantalla de carga

En cuanto se pulsa el lanzador aparece la pantalla de carga de la aplicación, visible en la figura 5.18. Esta pantalla permanece visible poco tiempo ya que el tiempo de carga es rápido.

Tras el proceso de carga, si la aplicación Android detecta que el usuario ha iniciado sesión previamente, se carga la pantalla con el listado de chequeos. En caso de que no haya datos de una sesión anterior, se carga la pantalla de login.

### 5.5.3. Pantalla de inicio de sesión

Tras cargar la aplicación, si no se detectan datos de inicio de sesión previos se muestra la pantalla de inicio de sesión, visible en la figura 5.19, en la que el usuario debe introducir su nombre de usuario y contraseña para iniciar sesión.

### 5.5.4. Pantalla de listado de chequeos

La figura 5.20 muestra la pantalla con el listado de chequeos del usuario. Desde aquí es posible cerrar la sesión de usuario, o pulsar en alguno de los chequeos para

Guardando captura...

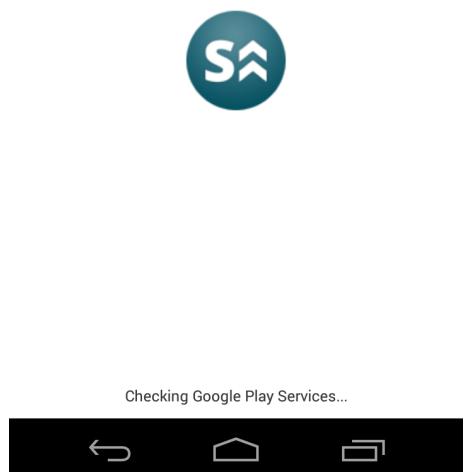


Figura 5.18: Pantalla de carga de la aplicación móvil

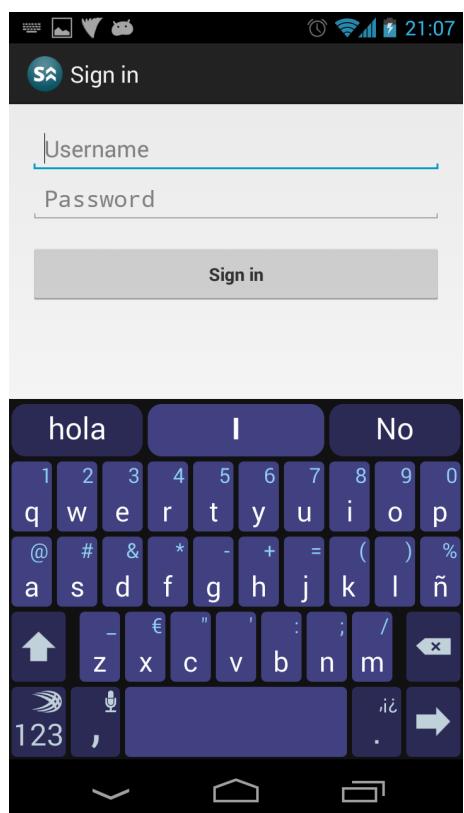


Figura 5.19: Pantalla de inicio de sesión de la aplicación móvil

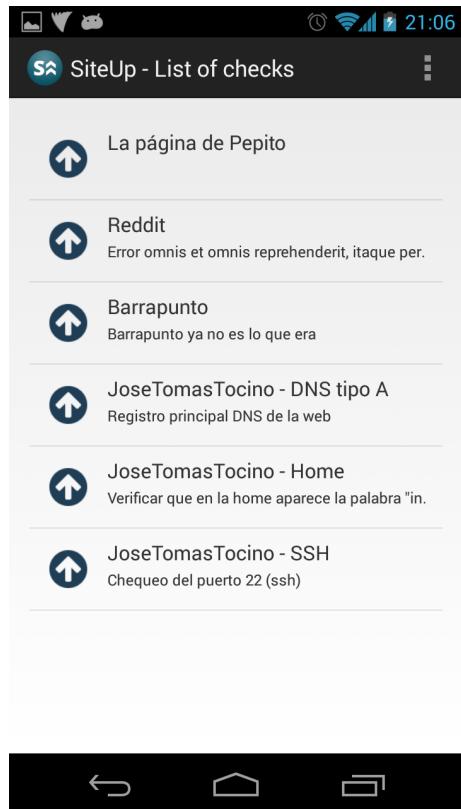


Figura 5.20: Pantalla de chequeos de la aplicación móvil

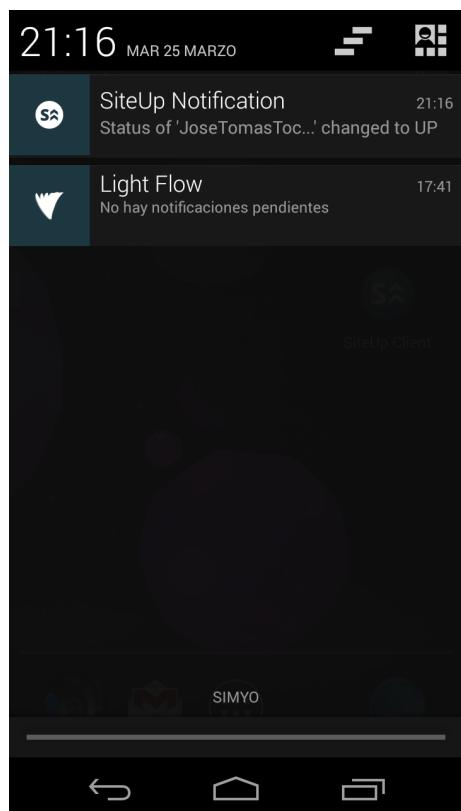


Figura 5.21: Aspecto de las notificaciones

ver sus detalles, lo cual abrirá un navegador y mostrará la plataforma web.

### **5.5.5. Notificación**

La figura 5.21 muestra el aspecto que tienen las notificaciones que la aplicación es capaz de recibir. Desde estas notificaciones es posible dirigirse a la web para ver los detalles del chequeo involucrado.

# Capítulo 6

## Implementación

Como complemento a la lectura de este capítulo se recomienda tener una copia local del repositorio del proyecto, disponible para su libre descarga desde la forja oficial [27].

### 6.1. Entorno de construcción

En esta sección se detalla el marco tecnológico utilizado para la construcción del sistema, basándonos en lo presentado en la sección 3.3, *Alternativas de solución*.

#### 6.1.1. Aplicación web

El entorno tecnológico de la aplicación web ha resultado ser muy variado y rico en tecnologías en total vigencia en la actualidad.

#### Herramientas de diseño y desarrollo

Como editor principal se ha utilizado **Sublime Text 2** [39], un editor freeware escrito en Python especialmente dotado para el desarrollo de aplicaciones web. Cuenta con un gran número de características que lo han hecho destacar en los últimos años, como la habilidad de seleccionar y editar a la vez varios fragmentos de texto, el rápido motor de *búsqueda difusa* para encontrar ficheros y cadenas o su capacidad de extensión a través de paquetes escritos por terceros y fácilmente instalables.

Para ciertas tareas, como la escritura de la presente memoria, se ha hecho uso del veterano editor **Emacs** [19], que sigue siendo la mejor opción a la hora de editar documentos de **LATEX**.

Se ha utilizado **draw.io** [14] para la generación de diagramas de casos de uso, secuencia, diagramas de flujo y similares. Se trata de una herramienta web, actualmente integrable en Google Drive, con un enorme número de elementos de dibujo y la capacidad de exportar en un gran número de formatos, entre ellos en formato vectorial PDF (Portable Document Format). Para el diseño visual de las interfaces de usuario se ha utilizado **Adobe Photoshop** [1].

## Gestión de dependencias

Para facilitar la gestión de las dependencias del proyecto se han utilizado **VirtualEnv** [40] y **VirtualEnvWrapper** [41]. Estas herramientas permiten generar *entornos virtuales* para cada proyecto, en los que se instalan las dependencias necesarias. Estos entornos se activan y desactivan, de forma que las bibliotecas instaladas en el entorno virtual de un proyecto no son accesibles desde el entorno del otro proyecto. Esto evita la polución del nivel general de bibliotecas y facilita el control estricto de las versiones de las dependencias.

Junto a `virtualenv`, la herramienta **pip** [33] permite guardar en un fichero anexo la lista de dependencias de un proyecto, de forma que sea fácil reinstalarlas todas si hubiese que repetir la instalación en otro sistema.

## Control de versiones

Todo el código fuente del proyecto se encuentra alojado en un repositorio público en **GitHub** [27], haciendo uso de los planes gratuitos. GitHub es una forja que utiliza el sistema de control de versiones **Git** [18]. Además del alojamiento de código, GitHub provee numerosas funcionalidades adicionales, tanto a nivel social (permitiendo a las forjas tener *followers*, por ejemplo) como a nivel funcional (ofreciendo sistemas de tickets, estadísticas, etcétera).

El uso de un control de versiones es fundamental por varios motivos. En primer lugar, sirve como sistema de copia de seguridad. En segundo lugar, permite deshacer cambios en el código que no funcionen bien, siendo siempre posible *volver atrás*. Por último, sirve como *cuaderno de bitácora* improvisado, ya que se guarda el historial de *commits* que el desarrollador va enviando junto a los mensajes, siendo posible ver en una línea temporal el progreso del trabajo.

## Lenguaje de programación

Como se ha comentado en numerosas ocasiones en la presente memoria, el lenguaje de programación elegido para el desarrollo del proyecto es **Python** [34], un lenguaje interpretado de alto nivel desarrollado por Guido Van Rossum en 1991. Python soporta múltiples paradigmas de programación, desde la orientación a objetos hasta la programación funcional, pasando por el clásico estilo imperativo. Su

principal uso ha sido como lenguaje de scripting, pero también tiene su hueco en contextos más amplios como lenguaje principal.

Su facilidad de aprendizaje, lo simple de su sintaxis (basada en la indentación para marcar los bloques) y su extensibilidad (sobre todo gracias al uso de métodos mágicos y metaprogramación) han hecho que el lenguaje sea muy popular y su uso en los últimos años se haya expandido enormemente.

## Framework de desarrollo

El framework de desarrollo elegido es **Django** [13], un proyecto de código abierto nacido en 2005. La meta fundamental de Django es facilitar la creación de sitios web complejos, haciendo especial hincapié en mantener un ritmo de desarrollo rápido y evitar la duplicidad de código. Así, gran parte del potencial de Django es la gran cantidad de funcionalidad que ya trae integrada, ya sea en el propio núcleo del framework o a través de *apps* oficiales que se incluyen en la distribución oficial.

Los elementos más destacables de Django son:

- Un sistema de mapeo objeto-relacional, que facilita enormemente el trabajo con elementos de bases de datos a través de instancias de modelos.
- Un sistema de procesamiento de peticiones a través de vistas.
- Un despachador de URLs basado en expresiones regulares.
- Soporte para plantillas.
- Un sistema de autenticación extensible.
- Una interfaz de administración que se adapta dinámicamente a cualquier proyecto.
- Un sistema de comentarios.
- Soporte integrado contra numerosos tipos de ataques vía web, como inyección SQL o cross-site scripting.

## Nivel de persistencia

**SQLite** es un sistema de gestión de bases de datos relacional de dominio público, contenido en una pequeña biblioteca. A diferencia de los sistemas de gestión de bases de datos cliente-servidor, el motor de SQLite no es un proceso independiente con el que el programa principal se comunica. En lugar de eso, la biblioteca SQLite se enlaza con el programa pasando a ser parte integral del mismo. El programa utiliza la funcionalidad de SQLite a través de llamadas simples a subrutinas y funciones. Esto reduce la latencia en el acceso a la base de datos, debido a que las llamadas a funciones son más eficientes que la comunicación entre procesos.

El conjunto de la base de datos (definiciones, tablas, índices, y los propios datos), son guardados como un sólo fichero estándar en la máquina host. Este diseño simple se logra bloqueando todo el fichero de base de datos al principio de cada transacción. Esto no impide que varios procesos o hilos pueden acceder a la misma base de datos sin problemas.

En general, SQLite no suele ser la mejor opción en sistemas de producción muy extensos, pero para proyectos de menor envergadura su rendimiento es más que suficiente.

## Gestión de tareas

Para la gestión de tareas de forma asíncrona se ha utilizado **Celery**, una cola de tareas asíncrona enfocada a operaciones en tiempo real. Celery está escrito en Python, pero se integra fácilmente con cualquier otro sistema que implemente su interfaz. Las unidades de ejecución básicas, las *tareas*, se ejecutan de forma concurrente en uno o varios *workers*. Las tareas pueden ejecutarse de forma asíncrona, en segundo plano, o de forma síncrona, haciendo que el flujo de ejecución espere hasta que el resultado de la tarea esté disponible.

Celery se comunica mediante mensajes, normalmente utilizando un **broker** que media entre los clientes y los workers. Para iniciar una tarea, el cliente pone un mensaje en la cola del broker. Entonces, el broker entrega el mensaje al worker de Celery. La opción recomendada, y la que se ha utilizado en SiteUp, es usar **RabbitMQ** como broker de mensajes.

RabbitMQ, y la mayoría de brókers de mensajes, implementan el protocolo AMQP, un estándar abierto para enviar mensajes entre aplicaciones y organizaciones, siempre que éstas sean capaces de interactuar usando las interfaces del protocolo.

## Herramientas de frontend

Para el desarrollo del frontend de la web, esto es, lo que el usuario acaba viendo y utilizando en el navegador, se han utilizado varias tecnologías, herramientas y bibliotecas.

En primer lugar, la base de la web está escrita en **HTML5** [24], un lenguaje de marcado usado para estructurar y presentar contenido en la web. Es la quinta revisión del estándar HTML (Hyper Text Markup Language), y su objetivo principal es unificar la sintaxis tan dispersa que había generado el uso de HTML4 y XHTML. Además, introduce una serie de elementos (etiquetas) con contenido semántico, tales como `<section>` y `<article>`. En la actualidad la especificación de HTML5 no está cerrada y siguen añadiéndose cambios. Es por ello que la compatibilidad con los navegadores no es total, aunque sí muy extensa.

Una buena práctica en el desarrollo web es desacoplar lo máximo posible el contenido de un documento (definido con HTML) del diseño del mismo, que habitual-

mente se define usando **CSS** [7]. En particular, en SiteUp se ha utilizado CSS3, la última versión del estándar. CSS permite la definición de **estilos** que se aplican a **elementos** de una web, mediante **reglas**. Así, las reglas se componen de un selector, que define los elementos a los que se le aplicará el estilo, y un bloque de declaración, en el que se describen las propiedades que se le aplicarán al elemento. Hay un gran número de reglas para la definición de selectores, y una extensa lista de propiedades definibles en CSS.

A pesar de todo, CSS cuenta con una serie de limitaciones de base. Por ejemplo, no cuenta con mecanismos sencillos para favorecer la modularidad del código. Por ello han surgido una serie de *metalenguajes* que añaden nuevas características para facilitar el desarrollo pero que, al final, acaban generando código CSS compatible. Uno de estos metalenguajes es **Sass!** (**Sass!**) [35]. Entre las funcionalidades que Sass añade se encuentran:

- Definición de variables.
- Anidamiento de reglas.
- Uso de *mixins*.
- Herencia de selectores.

Además, Sass cuenta con bibliotecas de terceros, como **Compass** [9], un framework de desarrollo que añade a Sass numerosas utilidades adicionales.

Con HTML5 y CSS3 es posible crear perfectamente sitios web con animaciones y funcionalidad básica, pero en SiteUp era necesario tener cierta interactividad a la hora de, por ejemplo, mostrar gráficas. Para ello resulta necesario utilizar un lenguaje de scripting para web, y ese lenguaje es **JavaScript** [26]. Se trata de un lenguaje dinámico utilizado principalmente para añadir dinamismo a los sitios web. Se lanzó en 1995 y desde entonces ha ido mejorando con el tiempo, sobre todo con la adición de numerosas API (Application Programming Interface) en los navegadores. El problema que ha tenido JavaScript es el irregular soporte de los diferentes navegadores, sobre todo las versiones de Internet Explorer. Así, era habitual utilizar código distinto para realizar el mismo procedimiento según fuese el navegador.

Afortunadamente existen bibliotecas de desarrollo que actúan como capa de abstracción y permiten al desarrollador olvidarse de las diferencias de compatibilidad entre navegadores. La biblioteca más conocida para uso general es **jQuery** [28]. Creada en 2006 por John Resig, se estima que más del 50 % de **todos** los sitios web a nivel mundial usan jQuery. Su popularidad se debe sobre todo a su facilidad de uso, su ligereza y la potencia que ofrece. Por suerte la evolución de los navegadores está permitiendo que el soporte nativo de JavaScript sea cada vez mejor, siendo progresivamente menos necesario el uso de bibliotecas *unificadoras* como jQuery, aunque su utilidad a la hora de trabajar, por ejemplo, con efectos o con cargas asíncronas de datos (**AJAX!** (**AJAX!**)) sigue siendo vigente.

Para el renderizado y muestra de las gráficas, en SiteUp se ha usado **D3.js** [12], una biblioteca para la manipulación de documentos basada en datos. D3 permite

generar interesantes visualizaciones de multitud de tipos, teniendo además capacidad para añadir interactividad, animaciones y propiedades dinámicas. Las gráficas generadas por D3 normalmente utilizan SVG (Structured Vector Graphics), por lo que el rendimiento es muy alto.

### Bibliotecas de terceros para backend

En el proyecto se han utilizado un gran número de bibliotecas auxiliares para facilitar el desarrollo y ampliar la funcionalidad de forma sencilla.

**South** South [36] es una biblioteca para Django que permite generar **migraciones** de datos y estructurales de la base de datos. El término *migración* hace referencia a una modificación estructural en la que se definen de forma lógica las diferencias entre el estado anterior y el posterior, de forma que sea posible trasladar o *transformar* los datos de un estado a otro, incluso dando la posibilidad de deshacer los cambios.

Es una función de gran utilidad, y muchos otros frameworks, como *Ruby on Rails*, incorporan este mecanismo de forma nativa. South se ha hecho tan popular y común que su código ha sido integrado en Django de forma nativa a partir de la versión 1.7.

**Fabric** Fabric [15] se compone de una herramienta de línea de comandos y una biblioteca que facilitan la comunicación con servidores remotos a través de SSH, facilitando el despliegue de aplicaciones y las tareas de administración.

Provee un conjunto básico de operaciones para ejecutar, ya sea forma local o remota, operaciones de toda clase: desde comandos de shell hasta subida y descarga de ficheros, pasando por funciones auxiliares.

En SiteUp, Fabric se ha utilizado para facilitar el despliegue de la plataforma web en el servidor remoto. Para ello, se desarrolló un fichero `fabfile.py`, que se encuentra en la raíz del proyecto web, que define las operaciones a realizar. Cuando se ejecuta el siguiente comando:

```
$ fab deploy
```

El sistema realiza las siguientes operaciones en el servidor remoto:

- Activa el entorno virtual (`virtualenv`).
- Descarga los últimos cambios del código desde la forja [27].
- Instala los nuevos requisitos definidos mediante el fichero de requisitos del entorno virtual (véase la sección 6.1.1 *Gestión de dependencias*).
- Dispone los ficheros estáticos del proyecto en un directorio especial, para que el servidor web pueda servirlos (véase la sección 5.1.2, *Plataforma web*).

- Aplica cualquier cambio que se haya realizado a la estructura de la web mediante migraciones.
- Reinicia los servicios.

La reducción de tiempo es considerable y la facilidad de uso es total.

**django-braces** Se trata de una extensión de Django que provee un conjunto de *mixins* que agregan funcionalidad a las vistas de forma sencilla.

Un **mixin** es una clase que ofrece cierta funcionalidad para ser heredada por una subclase, pero que no está pensada para funcionar por sí misma de forma autónoma. Así, la herencia que se produce al heredar de un mixin es básicamente un mecanismo para integrar funcionalidad adicional, en lugar de una herencia en el sentido clásico.

Entre la funcionalidad que provee django-braces se encuentran, por ejemplo, mixins para controlar el acceso a zonas para usuarios registrados o para devolver fácilmente respuestas JSON.

**django-extensions** Se trata de una colección de extensiones para Django de muy diversos tipos: comandos de administración, extensiones para la zona de administración, etcétera.

Algunas de las utilidades más interesantes que provee son las siguientes:

- El comando `shell_plus` provee una shell que carga automáticamente todos los modelos de la aplicación, de forma que sea fácil hacer consultas rápidas a la base de datos.
- El comando `graph_models` genera automáticamente un diagrama visual con el modelo Entidad-Relación de la base de datos.

**django-debug-toolbar** Proporciona una barra con extensa información de depuración: tiempos de cpu y renderizado, consultas SQL ejecutadas, plantillas utilizadas, variables de entorno y de petición, ficheros estáticos, señales dadas de alta, mensajes de registro...

Esta extensión es uno de los mecanismos de depuración más utilizados en los proyectos Django de todo el mundo. Además, automáticamente se desactiva en entornos de producción, por lo que no supone un problema.

### 6.1.2. Aplicación móvil

El entorno tecnológico de la aplicación móvil es el estándar para el desarrollo de aplicaciones Android hoy en día, que consta de los siguientes elementos.

### 6.1.3. Herramientas de desarrollo

Para el desarrollo de la aplicación móvil se ha optado por utilizar **Android Studio** [4], un entorno de desarrollo integrado basado en el popular software IntelliJ IDEA y perfectamente personalizado con las bibliotecas y herramientas necesarias para el desarrollo de aplicaciones Android.

Está preparado y gestionado por Google, y aunque está en una etapa temprana de desarrollo, su uso es totalmente viable y los resultados son correctos. Incluye elementos muy útiles, como la integración con los servicios de **Google Cloud Messaging**, que se han utilizado en el proyecto (véase la sección 5.1.2, *Aplicación Android*).

### 6.1.4. Servicio de notificaciones

Para el envío y la recepción de notificaciones se ha utilizado el servicio **Google Cloud Messaging for Android**. Se trata de un servicio gratuito que permite enviar datos desde un servidor a dispositivos Android que previamente se hayan registrado a través de una aplicación en particular.

El flujo de funcionamiento es el siguiente:

1. El usuario instala la aplicación, por ejemplo *SiteUp Client*.
2. La aplicación conecta con los servidores de GCM y pide el **código de registro** del dispositivo, que lo identifica unequívocamente.
3. La aplicación guarda ese código en el servidor de la plataforma web.
4. En caso de necesitar enviar una notificación, la plataforma web hace una petición al servidor de GCM, enviando el código de registro del dispositivo al que enviar la notificación y, lógicamente, el contenido de la notificación.
5. El servidor de GCM se encarga de transmitir la notificación al dispositivo Android.
6. El dispositivo recibe la notificación, siendo la aplicación móvil la encargada de actuar en consecuencia, mostrando un mensaje o realizando cualquier otra operación.

La facilidad con la que se pueden enviar mensajes de esta manera, de forma instantánea, es asombrosa y pone de manifiesto las facilidades que la plataforma Android pone a los desarrolladores.

## 6.2. Organización del código fuente

En esta sección se detalla la organización del código fuente del proyecto, describiendo la utilidad de los ficheros y directorios.

El código fuente presente en la forja [27] cuenta con los siguientes elementos en el directorio raíz:

- android-client: código fuente de la aplicación Android.
- logs: directorio con los ficheros de registro.
- memoria: directorio con el código fuente de la presente memoria.
- web: código fuente de la plataforma web.
- LICENSE: fichero con el texto de la licencia.
- INSTALLING-web.md: fichero con instrucciones de instalación de la plataforma web.
- README.md: fichero de presentación para la forja.

### 6.2.1. Plataforma web

El código de la plataforma web se encuentra íntegramente contenido en el directorio web del proyecto, que contiene lo siguiente:

- db: directorio donde se guarda la base de datos y otros ficheros.
  - celerybeat.db: datos del servicio Beat de Celery.
  - django-db.sqlite3: base de datos principal SQLite.
- siteup: directorio con ficheros a nivel de proyecto.
  - settings: directorio de configuraciones.
    - base.py: configuración general.
    - local.py: configuración para entorno local de desarrollo.
    - production.py: configuración para entorno de producción
  - celery.py: configuración de Celery.
  - urls.py: configuración de rutas de acceso web y URLs.
  - wsgi.py: configuración del punto de acceso para el servidor WSGI.
- siteup\_api: directorio de la app que gestiona la API.
  - migrations: directorio de migraciones generadas por South.
  - tests: directorio con los tests de la aplicación.
  - admin.py: configuración de la app para el panel de administración.
  - managers.py: gestores de modelos personalizados.
  - models.py: definiciones de los modelos de la base de datos.
  - utils.py: funciones auxiliares.

- validators.py: validadores personalizados para formularios y campos de modelos.
- views.py: vistas de acceso.
- siteup\_checker: directorio de la app que gestiona las tareas de chequeo a bajo nivel.
  - deployment: ficheros auxiliares para las tareas de despliegue.
  - management: tareas personalizadas, a ser lanzadas mediante la línea de comandos.
  - monitoring: procedimientos para chequeos:
    - ping.py: chequeo mediante ping.
    - http.py: chequeo mediante peticiones HTTP.
    - pport.py: chequeo de puertos.
    - ddns.py: chequeo de registros DNS.
  - templates: plantillas para los correos de notificación.
  - tests: tests para el código de la aplicación.
  - admin.py: configuración de la app para el panel de administración.
  - models.py: modelos de la aplicación.
  - tasks.py: definiciones de las tareas de Celery.
  - tasks\_notification.py: definiciones de las tareas de Celery relacionadas con la emisión de notificaciones.
- siteup\_frontend: directorio de la app que gestiona el frontend de la web.
  - static\_src: código fuente de los ficheros de los assets del front-end.
    - js: ficheros JavaScript.
    - scss: ficheros SASS.
  - static: código generado a partir de los fuentes en static\_src. El flujo de trabajo del front-end se detalla en la sección 6.3.2.
  - templates: plantillas HTML para las páginas de la web.
  - templatetags: funciones personalizadas para usar en el código de las plantillas.
  - context\_processors.py: ficheros de procesamiento del contexto personalizados.
  - forms.py: formularios adaptados para las secciones de la web, con código de validación.
  - tests.py: procedimientos de test para la aplicación.

- `views.py`: definición de las vistas (funciones que responden a las peticiones) de la web.
- `static`: directorio donde se agrupan los ficheros estáticos de la web.
- `Gruntfile.js`: fichero para el gestor de tareas Grunt.
- `package.json`: fichero de dependencias de node.js.
- `fabfile.py`: fichero de tareas de Fabric.
- `manage.py`: utilidad de línea de comandos de Django.
- `requirements.txt`: fichero de dependencias para el entorno virtual.

### 6.2.2. Aplicación móvil

Los ficheros de la aplicación móvil se encuentran contenidos en el directorio `android-client`. El contenido es el siguiente:

- `app`: directorio con el código fuente.
  - `build`: ficheros binarios compilados a partir del código fuente, entre los que se encuentra el fichero de compilación con extensión `.apk` que se instala en el dispositivo.
  - `libs`: bibliotecas de terceros.
  - `src`: código fuente. La ruta completa hasta los ficheros es `src/main/java/com/josetomastocino/siteupclient/app/`, siguiendo las prácticas habituales de los proyectos Java.
    - `CheckInList.java`: representa un chequeo en la lista de chequeos.
    - `CheckListActivity.java`: actividad que muestra la lista de chequeos.
    - `CheckListAdapter.java`: adaptador de array para la lista de chequeos.
    - `GcmBroadcastReceiver.java`: receptor de notificaciones GCM. Notifica al servicio `GcmIntentService`.
    - `GcmIntentService.java`: manejador de notificaciones GCM.
    - `LoginActivity.java`: actividad que gestiona el inicio de sesión.
    - `SplashScreenActivity.java`: actividad que gestiona la carga y el registro del dispositivo en Google Play Services.
  - `gradle*`: ficheros de configuración para el motor de compilación Gradle.

## 6.3. Detalles de implementación

En esta sección se describen algunos detalles en la implementación del proyecto que han resultado de interés suficiente para ser dignos de mención, ya sea por los conflictos que dieron, por su complejidad o por otras razones subjetivas.

### 6.3.1. Uso de herencia de modelos

Uno de los primeros dilemas que surgieron en el proyecto fue el usar o no herencia a la hora de definir los modelos de los cheques. La motivación principal era que los cheques, en general, tienen un gran número de parámetros y funciones comunes, por lo que resultaría interesante encontrar alguna forma de agrupar esos elementos comunes para evitar duplicidad de código.

#### Primera iteración: duplicación total

El primer enfoque que se intentó fue, directamente, repetir todos los campos y métodos en todos los cheques. Dado que el proyecto cuenta con cuatro tipos de cheques, los campos y métodos comunes se repetían cuatro veces, el código ocupaba mucho más espacio y era mucho más propenso a errores e inconsistencia.

#### Segunda iteración: herencia multitable

El siguiente paso fue utilizar herencia directa (*o multitabla*) de modelos. Pongamos un ejemplo:

```
class Lugar(models.Model):
    nombre = models.CharField(max_length=50)
    direccion = models.CharField(max_length=50)

class Restaurante(Lugar):
    acepta_reservas = models.BooleanField()
```

En este caso, el modelo *Restaurante* es, a su vez, un tipo de *Lugar*. Además, es posible que existan *Lugares* de forma independiente. Esto es así porque Django crea dos tablas, una para cada modelo, y se genera una relación para los casos en los que un *Lugar* sea a la vez un *Restaurante*.

El problema en el caso de SiteUp es que la clase base sería un *Chequeo*, y no tendría sentido que existiese un chequeo sin tipo, por lo que este enfoque no era válido. Además, la herencia multitable está desaconsejada en la mayoría de los casos.

### Tercera iteración: herencia abstracta

La herencia abstracta es similar a la herencia multitabla, pero con la diferencia de que para el *modelo base* no se genera una tabla, sino que los campos se guardan en la tabla de los modelos hijos. Modificando el caso anterior:

```
class Lugar(models.Model):
    nombre = models.CharField(max_length=50)
    direccion = models.CharField(max_length=50)

    class Meta:
        abstract = True

class Restaurante(Lugar):
    acepta_reservas = models.BooleanField()

class Colegio(Lugar):
    es_concertado = models.BooleanField()
```

En este caso, la tabla creada para el modelo *Restaurante* contendrá columnas para los campos *nombre*, *dirección* y *acepta\_reservas*, y la tabla para el modelo *Colegio* contendrá los campos *nombre*, *dirección* y *es\_concertado*. Además, cualquier método que se le añadiese al modelo base estaría disponible para los hijos.

Finalmente, esta fue el enfoque que se decidió usar. La única pega que aparece es que no hay manera sencilla de recuperar todos los objetos que hereden de un modelo – en el ejemplo, todos los objetos de base *Lugar* – como sí la había en el caso de la herencia multitabla, pero es una pega menor, subsanable de forma sencilla utilizando métodos mágicos como `__subclasses__`:

```
lugares = sum((list(x.objects.all()) for x in Lugar.__subclasses__()), [])
```

### 6.3.2. Workflow de front-end

Para la implementación del front-end se han seguido las últimas prácticas en el desarrollo web front-end a nivel mundial. En particular, se ha utilizado un *toolchain*<sup>1</sup> basado en Grunt [22], un sistema de ejecución de tareas desarrollado con node.js [31] pensado para automatizar tareas repetitivas o monótonas, como minificación, compilación, testeo, *linting*<sup>2</sup>, etc.

En la figura 6.1 se puede ver una diagrama del flujo de desarrollo de las hojas de estilo y del código JavaScript de la parte frontend. El desarrollador trabaja dentro de la carpeta `siteup_frontend/static_src`, que contiene los ficheros fuente de las hojas de estilo y los scripts de JavaScript.

---

<sup>1</sup>Toolchain: conjunto de herramientas que funcionan de forma encadenada y sucesiva.

<sup>2</sup>Linting: revisión de código para comprobar que sigue ciertas prácticas establecidas a nivel de lenguaje, organización, etc.

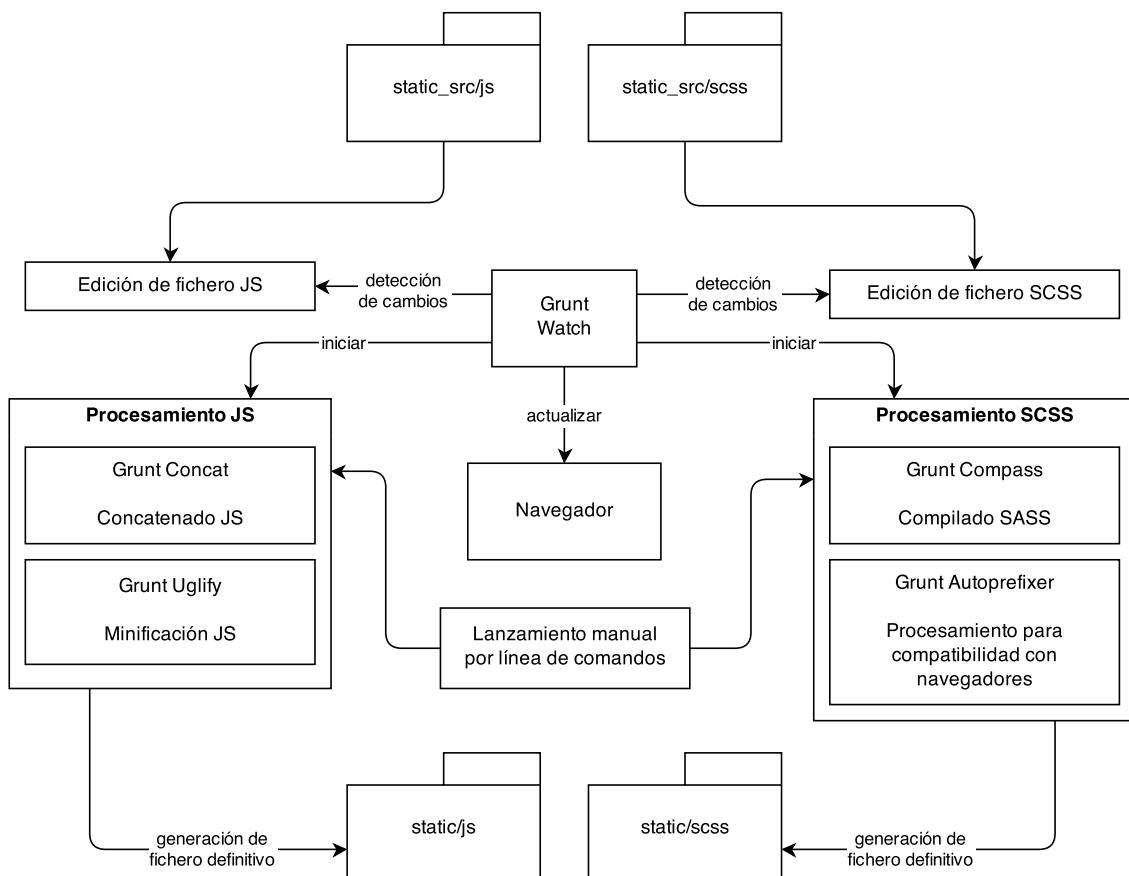


Figura 6.1: Diagrama de flujo de trabajo del front-end

Se detallan a continuación los pasos que se siguen en el flujo de trabajo.

### Instalación de paquetes de desarrollo

Para seguir este flujo de trabajo es necesario instalar ciertas dependencias en el sistema. En particular, es necesario instalar el entorno node.js, según las instrucciones de su distribución particular. Hecho esto, lo siguiente es ejecutar el siguiente comando:

```
$ npm install
```

Esto generará un directorio `node_modules` con las dependencias de node instaladas.

### Inicio del flujo de trabajo

El siguiente paso es lanzar uno de los plugins de grunt, **grunt-contrib-watch**, que se encarga de vigilar los cambios sobre los ficheros. Cuando detecta cambios, hace dos cosas:

- Primero, lanza las tareas de procesamiento que se detallan a continuación.
- Segundo, terminadas las tareas, lanza una señal para actualizar el navegador. Así se evita al usuario tener que ir al navegador y manualmente actualizar la web. Este proceso se conoce como **livereload**.

Este proceso se lanza utilizando el siguiente comando:

```
$ grunt watch
Running "watch" task
Waiting...
```

También es posible lanzar el procesamiento de forma manual ejecutando el siguiente comando:

```
$ grunt
```

El primer paso del procesamiento es siempre **limpiar** los ficheros generados previamente. Esta acción la lleva a cabo el plugin **grunt-contrib-clean**.

### Procesamiento JS

El procesamiento de los ficheros JavaScript consta de varios pasos. El primero de ellos es la **concatenación** de todos los ficheros de código. Es una buena práctica tener el menor número de ficheros anexos en una web para reducir al mínimo el número de peticiones al servidor. Para esto se utiliza el plugin **grunt-contrib-concat**, que navega por el directorio `siteup_frontend/static_src/js` y concatena todos los ficheros JavaScript que encuentre en un solo fichero.

El siguiente paso es la **minificación** del fichero generado. Básicamente lo que se busca es reducir el tamaño final del fichero mediante varios procesos, como la eliminación de espacios en blanco y la reducción de los nombres de las variables. Esto lo lleva a cabo el plugin **grunt-contrib-uglify**

Hecho esto, se copia el fichero generado a la carpeta de salida `siteup_frontend/static/js`. Una muestra de la salida generada por Grunt al realizar el procesamiento es la siguiente:

```
>> File "siteup_frontend/static_src/js/main.js" changed.

Running "clean:js" (clean) task
Cleaning siteup_frontend/static/js/main.js...OK

Running "concat:build" (concat) task
File "siteup_frontend/static/js/main.js" created.

Done, without errors.
Completed in 0.335s at Sat Apr 19 2014 19:00:22 GMT+0200 (CEST) - Waiting...
```

## Procesamiento SCSS

Como se mencionó previamente en la sección 5.1.1, para el desarrollo de las hojas de estilo se utilizó Sass, un superconjunto de CSS que añade numerosas funcionalidades que habitualmente resultan tediosas de hacer directamente en CSS. El único contra que tiene el uso de Sass es que los ficheros no pueden ser interpretados directamente por los navegadores, sino que es necesario compilarlos a CSS tradicional.

Así, el primer paso del procesamiento es la compilación de los ficheros Sass. Esto lo lleva a cabo el plugin **grunt-contrib-compass**, que lee los ficheros situados en `siteup_frontend/static_src/scss` y realiza la generación.

El siguiente paso es el procesamiento del fichero CSS generado para maximizar su compatibilidad con todos los navegadores. Esto se hace leyendo las propiedades CSS utilizadas y añadiendo prefijos en aquellos casos que sea necesario. El plugin encargado de hacer este trabajo es **grunt-autoprefixer**. Como ejemplo, si tenemos el siguiente código original:

```
a {
  transition: transform 1s
}
```

Tras el procesamiento se convertirá a:

```
a {
  -webkit-transition: -webkit-transform 1s;
  transition: -ms-transform 1s;
```

```
transition: transform 1s
}
```

Se ve que el plugin añade los prefijos necesarios para los navegadores basados en WebKit y para Internet Explorer (-webkit y -ms respectivamente).

Resulta interesante comentar que este plugin es configurable, de forma que podemos decidir a qué navegadores vamos a dar soporte. Por defecto añade los prefijos necesarios para soportar las dos últimas versiones de los navegadores más populares.

Tras este proceso, el fichero generado se copia a la carpeta `siteup_frontend/static/css`.

## 6.4. Gestión de la base de datos

El uso del mapeo objeto-relacional (ORM) que integra Django ayuda a que el desarrollador no tenga que interactuar con la base de datos utilizando lenguajes de bajo nivel como **SQL!** (**SQL!**), sino que se trabaja con una capa de abstracción mayor. Es más, la creación de las estructuras de la base de datos (tablas, índices, restricciones de integridad, etc.) también la lleva a cabo el ORM, en particular a través de scripts de sincronización y migraciones.

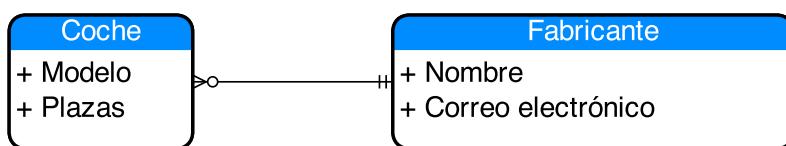


Figura 6.2: Diagrama ER de ejemplo 1

Pongamos por ejemplo un sencillo esquema entidad-relación como el que aparece en la figura 6.2, en el que se definen dos entidades con una relación de tipo 1:N. Haciendo uso del ORM de Django, este esquema se implementaría de la siguiente manera:

```
from django.db import models

class Fabricante(models.Model):
    nombre = models.CharField(max_length=100)
    correo_electronico = models.EmailField(max_length=254)

class Coche(models.Model):
    modelo = models.CharField(max_length=100)
    plazas = models.PositiveSmallIntegerField()

    fabricante = models.ForeignKey(Fabricante)
```

Una vez definidos los modelos, la base de datos se generará utilizando el siguiente comando de Django:

```
$ python manage.py syncdb
```

El comando `syncdb` sincroniza la base de datos con los modelos, creando las tablas necesarias. Desde este momento, el ORM de Django ofrecerá una serie de comandos para hacer consultas de manera sencilla. Por ejemplo, para obtener todos los fabricantes:

```
fabricantes = Fabricante.objects.all()
```

Para obtener el fabricante llamado *Seat*:

```
fabricante = Fabricante.objects.get(nombre="Seat")
```

Es posible embeder condiciones algo más avanzadas. Por ejemplo, para obtener los coches con más de 3 plazas:

```
coches_grandes = Coche.objects.filter(plazas__gt=3)
```

Django detecta las relaciones entre modelos y provee mecanismos para navegar esas relaciones, por ejemplo.

```
fabricante = Fabricante.objects.get(nombre="BMW")
coches_de_bmw = fabricante.coche_set.all()
```

```
leon = Coche.objects.get(modelo="Leon")
fabricante_del_leon = Coche.fabricante
```

Django automáticamente detecta las relaciones y genera las consultas necesarias para unir los datos de las tablas involucradas. Django también cumple con las restricciones de integridad que se impongan en la base de datos. En nuestro ejemplo, si borramos un fabricante, todos sus coches son borrados también de forma automática.

```
>>> fabricante = Fabricante.objects.get(nombre="BMW")
>>> fabricante.delete()
>>> coche_318 = Coche.objects.get(modelo="318i")
Traceback (most recent call last):
...
DoesNotExist: Coche matching query does not exist.
```

Este comportamiento puede configurarse para que no se produzca el borrado *en cascada*, al igual que muchísimos otros parámetros que también son configurables.

Django también ofrece métodos para comprobar que los datos de un modelo son correctos antes de guardarlos en la base de datos. Por ejemplo, si intentamos introducir una dirección de correo inválida, el sistema nos alertará:

```
>>> fabricante = Fabricante()
>>> fabricante.nombre = "Fab"
>>> fabricante.correo_electronico = "invalido"
>>> fabricante.full_clean()
Traceback (most recent call last):
```

```
...
ValidationErrors: {'correo_electronico': [u'Enter a valid email address.']}
```

### 6.4.1. Cambios en la estructura

El problema del proceso planteado ahora es que tiene una limitación: no contempla cambios estructurales en los modelos una vez que la base de datos esté creada. Para ello es necesario utilizar **South**, tal y como se comentó en la sección 6.1.1.

South compara el estado de la base de datos con los nuevos modelos, detecta los cambios y realiza las operaciones necesarias. Si, por ejemplo, añadiésemos un atributo *fecha\_fundacion* al modelo *Fabricante*, éste quedaría así:

```
class Fabricante(models.Model):
    nombre = models.CharField(max_length=100)
    correo_electronico = models.EmailField(max_length=254)
    fecha_fundacion = models.DateField()
```

Ahora, para evitar perder los datos que ya tuviésemos en la base, utilizaríamos South para crear una **migración**:

```
$ python manage.py schemamigration aplicacion --auto
+ Added field fecha_fundacion on main.Fabricante
Created 0002_auto__add_field_fabricante_fecha_fundacion.py.

$ python manage.py migrate aplicacion
Running migrations for main:
- Migrating forwards to 0002_auto__add_field_fabricante_fecha_fundacion.
> main:0002_auto__add_field_fabricante_fecha_fundacion
- Loading initial data for main.
Installed 0 object(s) from 0 fixture(s)
```

En caso de que borrásemos el campo o quisiésemos hacer más cambios, estas migraciones pueden deshacerse o modificarse sin mayor problema.



# **Capítulo 7**

## **Pruebas**

En este capítulo se detallan las baterías de pruebas a las que se ha sometido **SiteUp**, ya sean de carácter manual o automatizadas mediante software específico.

### **7.1. Estrategia**

La estrategia de pruebas seguida en SiteUp es híbrida. Por un lado, se cuenta con una batería de pruebas automatizada, basadas en el motor de pruebas que incluye el framework de desarrollo, que verifican las funcionalidades más críticas del sistema.

Por otro lado, se cuenta con pruebas manuales realizadas de forma periódica. A corto plazo y de forma frecuente se revisaban las funcionalidades más inmediatas de las aplicaciones. A largo plazo, se han dado de alta numerosos chequeos y se han verificado a lo largo del tiempo.

### **7.2. Entorno de pruebas**

Para las pruebas automáticas, el entorno de pruebas es una copia virtual del entorno de producción real, creado automáticamente por el software de testeo, de forma que no se modifiquen los valiosos datos de la base de datos de producción. En cada ejecución de los tests se crea una base de datos y un entorno nuevos, vacíos, en los que se llevan a cabo las pruebas.

Para las pruebas manuales, dado que se realizan directamente sobre el sistema en producción, el entorno coincide con lo dispuesto en la sección 5.1.1.

## 7.3. Roles

Se presentan dos roles principales necesarios para la ejecución de las pruebas.

### 7.3.1. Desarrollador principal

En primer lugar, el desarrollador del producto es el principal probador del software. Por un lado, es el encargado del desarrollo y ejecución de las baterías de pruebas automatizadas, teniendo la obligación de interpretar los resultados y actuando en consecuencia.

Por otro lado, también ha de probar el producto manualmente como un usuario más, sobre todo a tenor de lo expuesto en la sección 1.2, en la que se refleja que la motivación principal del proyecto es una necesidad personal del desarrollador.

### 7.3.2. Probadores externos

Además del desarrollador principal fue conveniente contar con la ayuda de varios probadores externos que dieran su punto de vista sobre el software, como usuarios. Así, realizaron pruebas de carácter manual a corto plazo, opinando sobre la interfaz de usuario, la usabilidad y la responsividad del proyecto, así como a largo plazo, dando de alta diversos chequeos y probando su funcionalidad.

En todas las etapas, el desarrollador obtenía feedback de los probadores, integrando en la medida de lo posible los consejos y apreciaciones que se obtenían.

## 7.4. Niveles de pruebas

### 7.4.1. Pruebas unitarias

Las **pruebas unitarias** tuvieron por objetivo localizar errores en los elementos software antes de ser integrados con el resto de elementos del sistema.

En SiteUp, el grueso de las pruebas unitarias se centró principalmente en el módulo de chequeo, organizado dentro de la aplicación `siteup_checker`. Los diversos casos de test comprobaron de forma individual que los procedimientos que llevan a cabo los chequeos devuelvan el resultado correcto, asegurando la ausencia de falsos positivos y resultados erróneos.

Por ejemplo, para la verificación del procedimiento de **chequeo HTTP** se hizo uso del sitio web **HttpBin** [25], una web especialmente pensada para facilitar el testeo de conexiones HTTP.

### 7.4.2. Pruebas de integración

Las **pruebas de integración** tuvieron por objetivo localizar errores en módulos completos, analizando la interacción entre varios artefactos software.

Como ejemplo, tras realizar las pruebas unitarias en los módulos de chequeo, se pasó a hacer pruebas de integración entre los procedimientos de chequeo y las instancias de los modelos de chequeo dadas de alta por los usuarios, de forma que se asegurase que la creación de un chequeo por parte de un usuario era reflejada en la base de datos y daba lugar a la correcta ejecución del procedimiento de chequeo.

### 7.4.3. Pruebas de sistema

Las pruebas de sistema, que buscan asegurar que el sistema cumple con todos los requisitos establecidos, se desarrollaron de forma continua a medida que iba avanzando el proyecto.

En cada iteración, las pruebas se llevaron a cabo utilizando un entorno virtual y el servidor de pruebas que integra el propio framework Django, y se utilizó software de *throttling* de red para simular condiciones de red adversas que se asemejasen a las condiciones reales.

### 7.4.4. Pruebas de aceptación

Para verificar que el producto estaba listo para el paso a producción se hizo un despliegue en un servidor real, ubicado en Alemania, y se dio de alta una decena de chequeos de todos los tipos. El servidor ha permanecido online desde mediados del mes de febrero hasta la fecha.

En este tiempo, además de recibir las actualizaciones del software a medida que el desarrollo avanzaba, el proyecto ha detectado de forma eficiente las caídas en el servicio de los chequeos dados de alta, notificando de manera correcta tanto por correo electrónico como a través de la aplicación de Android.

En particular se puede poner de ejemplo la web oficial de la Universidad de Cádiz, que se ha utilizado como objetivo de los chequeos. En el tiempo que ha estado el chequeo activo se han detectado caídas en la web unas 15 veces, la última el 16 de abril de 2014, día en el que durante la mañana la web ha pasado a estar inaccesible más de 7 veces durante períodos de entre 10 y 30 minutos.

## 7.5. Implementación de pruebas

Como se ha comentado, la implementación de las pruebas automáticas se ha hecho utilizando las capacidades de testing que provee el framework Django. Una vez instalado el sistema es posible lanzar las pruebas utilizando el siguiente comando:

```
$ python manage.py test  
Creating test database for alias 'default'...  
  
(output omitted)
```

---

```
Ran 33 tests in 28.651s
```

```
OK  
Destroying test database for alias 'default'...
```

Tras la ejecución de los tests se muestra una lista de aquellos que han fallado, ya sea por no haberse cumplido los assertos definidos o por otro error no controlado.

# Capítulo 8

## Conclusiones

Durante el transcurso del desarrollo de SiteUp, y sobre todo al término del mismo, se han obtenido unas conclusiones y unos resultados, tanto de forma personal como para con la comunidad, que intentaremos reflejar en este capítulo.

### 8.1. Objetivos cumplidos

Al término del desarrollo del proyecto, el proyecto ha completado todos los objetivos a cumplir, presentes en el planteamiento inicial y detallados en la sección 1.3. En particular:

- Se ha creado un conjunto de herramientas de chequeo de servicios de internet, que se encuentra disponible en el módulo `siteup_checker/monitoring` del proyecto.
- Se ha creado una aplicación online de acceso público con la que los usuarios pueden gestionar sus chequeos de manera sencilla, que en la fecha de escritura de la presente memoria es accesible en la url <http://siteup.josetomastocino.com>.
- Se ha establecido un sistema de notificaciones con el que mantener informados a los usuarios tanto por correo electrónico como mediante una aplicación móvil para el sistema operativo Android desarrollada a tal efecto.

La compleción total de los objetivos funcionales también pone de manifiesto que se han alcanzado satisfactoriamente los objetivos transversales y personales dispuestos en el inicio. Se han afianzado fuertemente los conocimientos de desarrollo web en general y con Django en particular hasta un nivel de competencia suficiente para servir como baza en la búsqueda laboral. Además, SiteUp ha servido satisfactoriamente como primera incursión en el mundo del desarrollo de aplicaciones Android.

## 8.2. Conclusiones personales

SiteUp es uno de los pocos proyectos personales que han surgido para suplir una **necesidad** real y específica y que, al término del desarrollo, han conseguido su objetivo. La idea inicial, expuesta en la sección 1.2, ha servido como eje motor del proyecto en todo momento. Si se diese la situación presentada en el inicio, el proyecto SiteUp sería capaz de detectar el problema e informar en consecuencia.

### 8.2.1. Lecciones aprendidas

Si bien es cierto que el proyecto no es de una complejidad técnica muy alta, el **cúmulo de tecnologías** diferentes utilizadas en cada uno de los niveles de abstracción es sobrecogedor. Adquirir una **competencia** básica en todas estas tecnologías es una tarea de una envergadura importante, y la correcta ejecución del sistema es comparable al mecanismo de un reloj, en el que todas las piezas deben funcionar de manera coordinada.

Desde la configuración de los numerosos servicios usados en el entorno de producción, al despliegue de la aplicación, pasando por el desarrollo de los módulos de chequeo, hasta llegar a la implementación del código de la capa de presentación, sin olvidar la implementación de la aplicación Android. En todos los niveles, en unos en mayor medida que en otros, se ha alcanzado una **soltura** suficiente para el desarrollo y posterior lanzamiento de un producto completo.

Al tratarse de un proyecto de ejecución continua en el que constantemente se están haciendo chequeos y enviando notificaciones, el flujo de *feedback* entre los usuarios y el desarrollador (que en las etapas iniciales son la misma persona) es mucho más **dinámico** que en cualquier otro proyecto en el que la ejecución fuese momentánea, como un juego o una aplicación de escritorio. Este planteamiento ha permitido que en SiteUp se hayan detectado rápidamente conflictos y problemas en el sitio, como por ejemplo que algunos usuarios prefieren no recibir resúmenes diarios de sus chequeos, motivando la inclusión de una opción en sus perfiles para desactivar esta funcionalidad.

Finalmente, tras la conclusión del proyecto, se ha conseguido un **punto de vista** diferente de las alternativas presentadas en la sección 1.4, *Estado del arte*.

Con un conocimiento más formado de lo que hay detrás (salvando las distancias) de estos proyectos resulta más sencillo **evaluar** en qué aciertan y en qué fallan, qué les falta y qué les sobra, y sobre todo qué se puede aprender de ellos para su integración en SiteUp. En particular, me ha sorprendido que no sea común que esta clase de servicios proporcionen una aplicación móvil nativa que sirva, al menos, para la recepción de notificaciones, tal y como se hace en SiteUp. La inversión es despreciable si la comparamos con la utilidad que provee.

## 8.3. Trabajo futuro

Durante la etapa de desarrollo y, sobre todo, durante las sesiones de pruebas han surgido muchos frentes de trabajo futuro, unos de mayor relevancia que otros, pero importantes al fin y al cabo y que aquí se presentan.

### 8.3.1. SiteUp Probes

**SiteUp Probes** es un proyecto que busca crear *sondas* que ejecuten de manera independiente los chequeos dados de alta en una instalación de SiteUp, enviando los resultados de vuelta al servidor central. La idea es que estas sondas esté geográficamente distribuidas, de forma que los resultados de los chequeos no dependan tanto de la situación del nodo central, sino que se calculen como una media de los resultados obtenidos por distintas sondas.

La motivación de esta ampliación viene dada por lo descrito en la sección 2.6.1, *Compromiso de la plataforma de despliegue*. El uso de un solo punto a la hora de lanzar los chequeos puede comprometer la veracidad y corrección de los resultados.

Se trata de un escalón importante y complejo en el trabajo de ampliación del proyecto. A nivel de implementación implicaría habilitar alguna clase de canal de comunicación entre el nodo central de SiteUp y las sondas para la comunicación de los chequeos a ejecutar y de los resultados de éstos. A nivel logístico conllevaría la búsqueda de nuevos sistemas en los que instalar las sondas, más allá del servidor de producción que está alquilado para el proyecto.

### 8.3.2. Monetización

Para asegurar la supervivencia de cualquier proyecto es necesario contar alguna clase de **modelo de negocio** que sufrague los gastos que se generan. En el caso de SiteUp, la **monetización** se puede conseguir mediante la creación de niveles de cuentas de usuarios, que gocen de distintos beneficios. Por ejemplo:

- Las cuentas gratuitas solo podrán crear un chequeo, y el tiempo entre chequeos será como mínimo de 5 minutos.
- Las cuentas básicas costarán 5€ al mes, podrán crear hasta 5 chequeos y el tiempo mínimo entre chequeos será de 2 minutos.
- Las cuentas premium costarán 10€ al mes, podrán crear hasta 30 chequeos y el tiempo mínimo entre chequeos será de 1 minuto.

La implementación de este modelo de monetización no es compleja, aunque la gestión de pasarelas de pago y cobros recurrentes es una de los procedimientos más problemáticos a nivel de programación, sobre todo a la hora de verificar la correcta ejecución del sistema.

### 8.3.3. Ampliación de los tipos de chequeos

Los cuatro tipos de chequeos actualmente presentes en SiteUp cubren un gran abanico de frentes a la hora de vigilar el estado de un servicio de Internet, pero existen (y seguirán surgiendo) nuevos puntos que convendría comprobar. Como ejemplos de posibles chequeos a añadir en un futuro se podrían considerar:

- Chequeo de servidores de bases de datos mediante el envío de consultas de verificación.
- Chequeo de transacciones. Esto es, chequeos compuestos de varios pasos.
- Chequeos de APIs con campos adicionales, como autenticación, cabeceras especiales o parámetros GET/POST.

El trabajo en este punto pasaría, básicamente, por encontrar una manera de modularizar satisfactoriamente los chequeos, de forma que fuese trivial añadir más tipos de chequeos sin necesidad de modificar código en todas las partes de la aplicación.

### 8.3.4. Ampliación de tipos de notificación

Actualmente, SiteUp envía notificaciones mediante correo electrónico y mediante la aplicación Android. Como posibles ampliaciones se podrían considerar el uso de mensajes cortos (SMS) y el desarrollo de una app para iOS capaz de recibir notificaciones PUSH.

# Apéndice A

## Manual de instalación de la plataforma web

A continuación se presentan las instrucciones de instalación de la plataforma web en un servidor de producción.

Las instrucciones de instalación que se presentan a continuación suponen un sistema con unos requisitos de hardware mínimos especificados en la sección 5.1.1, *Servidor de producción*. Se supone un servidor con una distribución basada en paquetería Debian con acceso root.

La versión más actualizada de las instrucciones de instalación de la plataforma web se encuentran en el fichero `INSTALLING-web.md` alojado en el directorio raíz de la forja de código [27].

### A.1. Instalación inicial

#### A.1.1. Descarga de código

Para instalar la plataforma web es necesario clonar el repositorio [27] con el código de la aplicación. Primero, creamos una ubicación donde alojar el código desde la terminal:

```
$ mkdir /srv/siteup -P  
$ cd /srv/siteup
```

Una vez ahí, clonamos el repositorio desde Github:

```
$ git clone https://github.com/JoseTomasTocino/pfc-ii.git .
```

### A.1.2. Entorno virtual y dependencias

Como se ha comentado previamente, el proyecto utiliza **virtualenv** y **virtualenvwrapper** para una gestión más limpia de las dependencias. La instalación de estos dos elementos es sencilla, en primer lugar se instalan los dos paquetes:

```
$ pip install virtualenv
$ pip install virtualenvwrapper
```

En segundo lugar, añadimos el código de *bootstrapping* de `virtualenvwrapper` a nuestro perfil de terminal, habitualmente `.bashrc`:

```
$ cat >> ~/.bashrc

if [ -f /usr/local/bin/virtualenvwrapper.sh ]
then
    source /usr/local/bin/virtualenvwrapper.sh
fi

EOF
```

Hecho esto, será necesario reiniciar la terminal, tras lo cual podremos crear el entorno virtual e instalar las dependencias:

```
$ mkvirtualenv siteup
$ sudo apt-get install python-dev
$ pip install -r web/requirements.txt
```

### A.1.3. Creación de la base de datos

El siguiente paso es preparar la base de datos. El siguiente comando genera la estructura básica de la base de datos para las tablas y aplica las migraciones existentes:

```
$ python web/manage.py syncdb
$ python web/migrate siteup_api
```

Tras esto, la base de datos se habrá generado y se encontrará en el fichero `web/db/django-db.sqlite3`.

### A.1.4. Instalación del servidor

En estas instrucciones vamos a usar **Nginx** como servidor frontal, aunque también es posible utilizar Apache. Para instalar nginx, en sistemas GNU/Linux basados en paquetería Debian, el procedimiento es muy sencillo:

```
$ sudo apt-get install nginx -y
```

### A.1.5. Instalación del bróker de mensajes

El bróker de mensajes elegido es **RabbitMQ**, aunque en principio valdría cualquier otro bróker que fuese compatible con el estándar AMQP. La instalación es sencilla:

```
$ sudo apt-get install rabbitmq-server -y
```

La configuración por defecto es suficiente para las necesidades del proyecto.

### A.1.6. Ejecución de los servicios

Para el control de los servicios que ejecutan el proyecto es necesario utilizar alguna clase de software de monitorización de procesos. La opción recomendada es **supervisor**. Su instalación es sencilla:

```
$ sudo apt-get install supervisor -y
```

Tras esto, es necesario generar el fichero de configuración de supervisor. SiteUp integra una tarea que automáticamente genera este fichero por nosotros. Solo tendremos que indicar un par de variables, que nos pedirá por teclado. En particular:

- GMAIL\_PASS, es la clave de la cuenta de correo utilizada para enviar notificaciones. Se define en el fichero web/siteup/settings/base.py.
- GCM\_API\_KEY, es la clave de la API del servicio Google Cloud Messaging.

Para generar el fichero de configuración utilizaremos el siguiente comando:

```
$ python web/manage.py supervisor_conf
```

Se generará un fichero supervisor-siteup.conf con la configuración. Es buena idea revisar el fichero para comprobar que todas las rutas son correctas. Tras ello, solo queda mover el fichero al lugar correcto y reiniciar el demonio de supervisor.

```
$ sudo mv web/supervisor-siteup.conf /etc/supervisord/conf.d
```

En entornos de producción, normalmente no se usará supervisor, ni gunicorn ni nginx, sino que utilizaremos el servidor integrado de Django. En esas circunstancias será necesario definir las variables mencionadas previamente de otra manera. La manera más habitual es añadirlas al **script de activación** del entorno virtual, de la siguiente manera:

```
$ cdvirtualenv  
$ cd bin  
$ cat >> postactivate  
export GMAIL_PASS=yourpass  
export GCM_API_KEY=yourkey
```

EOF

### A.1.7. Configuración del servidor frontal

El siguiente paso es configurar el servidor nginx para que reciba las peticiones apropiadas. Vamos a suponer que SiteUp va a recibir las peticiones en el dominio siteup.uca.es. Creamos el fichero /etc/nginx/sites-available/siteup y añadimos el siguiente contenido:

```
server {
    listen 80;
    server_name siteup.uca.es;

    # Static files
    location /static/ {
        root /srv/siteup/web/siteup_frontend;
        try_files $uri $uri/ =404;
    }

    location / {
        proxy_set_header X-Real-IP      $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Host          $http_host;
        proxy_redirect off;

        if (!-f $request_filename) {
            proxy_pass      http://127.0.0.1:8000;
            break;
        }
    }
}
```

Tras eso, activamos el fichero de configuración de la siguiente manera:

```
$ cd /etc/nginx/sites-enabled
$ ln -s ../sites-available/siteup
```

Básicamente lo que hace ese fichero de configuración es revisar las peticiones que llegan y, si apuntan a la ruta /static, nginx se encarga de servir esos ficheros estáticos de front-end. Si no, pasa la petición al servidor gunicorn que estará escuchando en la dirección local 127.0.0.1:8000.

### A.1.8. Lanzamiento final

Tras toda esta configuración, los últimos pasos que quedan son activar los servicios y servidores. Primero, lanzamos el controlador supervisor, que a su vez lanzará el servidor dinámico gunicorn y el servidor de tareas celery:

```
$ sudo service supervisord reload
```

Seguidamente, reiniciamos el servidor frontal para que pueda empezar a recibir peticiones:

```
$ sudo service nginx restart
```

Con esto, el proyecto estará funcionando y debería ser accesible a través del navegador.

## A.2. Mejora del rendimiento

La configuración por defecto de **RabbitMQ** hace que consuma mucha memoria y puede llegar a resultar un problema en sistemas con recursos limitados, como los VPS de bajo coste.

Para arreglar esto, una opción es modificar el fichero de configuración, situado en `/etc/rabbitmq/rabbitmq.config` para limitar la memoria máxima usada por Rabbit a un 15 % del total. El contenido a añadir al fichero es el siguiente:

```
[  
    {rabbit, [{vm_memory_high_watermark, 0.15}]}  
].
```

Por otro lado, si el número de chequeos que se preveen tener es bajo, es posible reducir el número de workers que utiliza **Celery** modificando el fichero `supervisor-siteup.conf` que se generó en los pasos anteriores, cambiando el valor del parámetro `-c 16` a un número menor.



# Apéndice B

## Manual de usuario

En este capítulo se explicará el funcionamiento de la aplicación desde el punto de vista del usuario final, detallando las opciones de cada una de las secciones.

Para poder acceder a la aplicación, tal y como se explicó en el apéndice *Manual de instalación de la plataforma web*, será necesario utilizar el comando:

```
./oflute
```

### B.1. Ejecución inicial

Al lanzar la aplicación, se abrirá una ventana y aparecerán, consecutivamente, la pantalla de créditos del autor y la pantalla de presentación del juego. Pasados unos momentos éstas desaparecerán, aunque tiene la opción de omitir cada una de las pantallas pulsando la tecla escape.

Una vez desaparezcan ambas pantallas, se iniciará una animación para mostrar el menú principal. Puede cancelar la animación y mostrar rápidamente el menú principal pulsando la tecla escape.

Cuando las animaciones concluyan, el menú principal ya será completamente funcional. Podremos acceder a las diferentes opciones, ya sea haciendo click con el ratón en los botones de pantalla, o pulsando alguna de las teclas de acceso directo. Las secciones y teclas asignadas son las siguientes:

- **Analizador de notas** (*tecla 1*) Abre el analizador de notas.
- **Canciones** (*tecla 2*) Pasa al menú de selección de canciones.
- **Lecciones** (*tecla 3*) Abre el menú de elección de lecciones.
- **Calibrar micrófono** (*tecla 4*) Lanza el sistema de calibración del micrófono.
- **Salir** (*tecla 5*) Cierra la aplicación

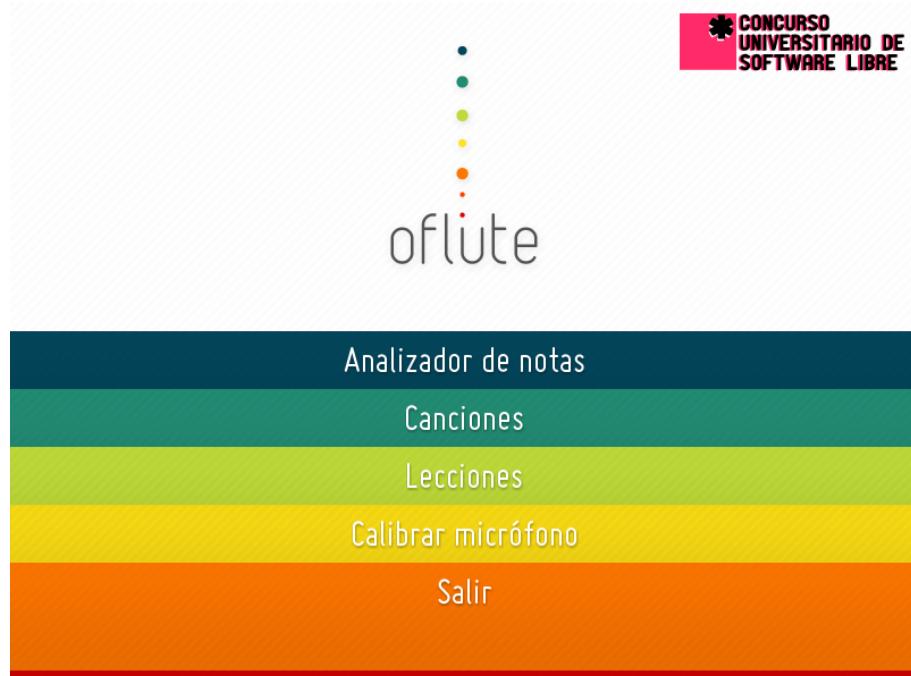


Figura B.1: Pantalla de créditos del autor



Figura B.2: Pantalla de presentación del juego

Lo recomendado, en la primera ejecución de **oFlute**, es lanzar la sección *Calibrar micrófono*, ya sea pulsando el botón con el ratón o mediante la tecla 4. Con esto nos cercioramos de que el programa es capaz de distinguir el sonido del micrófono del ruido de fondo. Una vez hecho, ya estará todo listo para acceder al resto de secciones.



## B.2. Sección – calibrar micrófono

Pulsando el botón correspondiente, o mediante la tecla 4, pasaremos a la sección de calibración del micrófono. En primer lugar, aparecerá el siguiente mensaje:



Como se indica en la Figura B.4, para comenzar la calibración del micrófono, el usuario deberá pulsar la tecla espacio y guardar silencio en el micrófono durante dos segundos. En ese momento, aparecerá el siguiente mensaje:

Una vez pasen los dos segundos y concluya la calibración, se mostrará el siguiente

## Calibrando. Guarde silencio.

Figura B.5: Mensaje al inicio del calibrado

mensaje:

Calibración completa.  
Pulse escape para volver al menú.

Figura B.6: Mensaje al final del calibrado

Cuando concluya el proceso, la aplicación habrá guardado el valor umbral de ruido del micrófono, lo que facilitará el análisis del sonido de la flauta.

Hay casos en los que la calibración será defectuosa, y un mensaje informará de ello. En tal caso, no habrá más que comprobar la configuración del micrófono y repetir el proceso de calibrado.

### B.3. Sección – analizador de notas

Al pulsar el botón correspondiente a la sección del analizador de notas, mediante una animación aparecerán los elementos de esta parte de la aplicación (figura ?? en página ??).

Una vez terminen de mostrarse todos los elementos, el usuario podrá empezar a utilizar la funcionalidad de la sección. En concreto, el analizador de notas nos permite ver reflejada en el pentagrama la nota que toquemos con la flauta.

Su uso, pues, es muy sencillo. Simplemente tendremos que tocar nuestra flauta de forma que el micrófono sea capaz de captar su sonido. Si lo hacemos correctamente, el analizador mostrará en cada momento la nota que está tocando sobre el pentagrama.

Una vez hayamos acabado de utilizar esta sección de la aplicación, podremos pulsar en el botón *Volver*, o en la tecla escape del teclado, lo que nos llevará de vuelta al menú principal.

### B.4. Sección – lecciones

Si seleccionamos la opción *Lecciones* en el menú principal, o alternativamente presionamos la tecla 3 del teclado, la aplicación mostrará una animación para ocultar

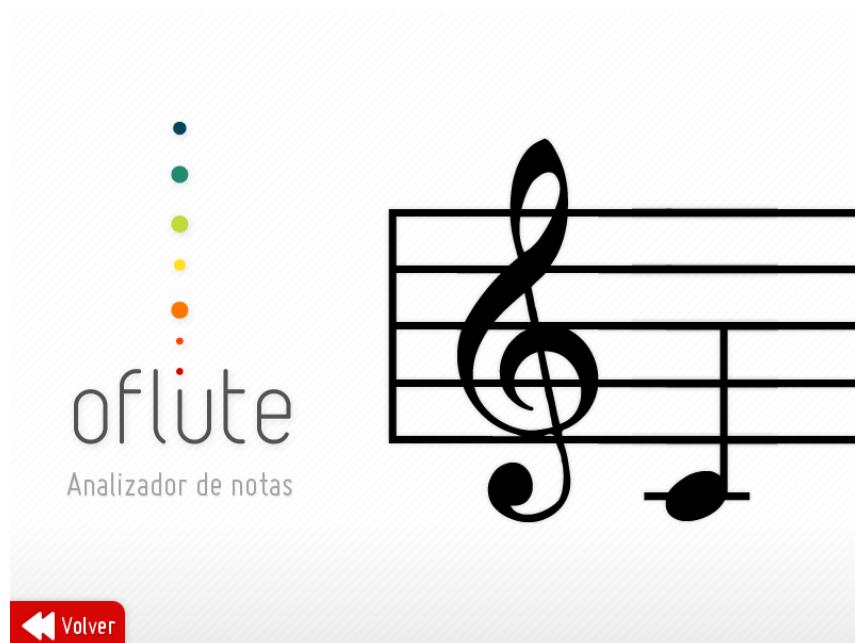


Figura B.7: Pantalla del analizador de notas

el menú principal y nos mostrará el menú de selección de lecciones.



Figura B.8: Pantalla del menú de selección de lecciones

En cualquier momento podemos hacer click en el botón *Volver al menú*, o pulsar la tecla escape del teclado, para ir a la sección anterior.

En el menú de selección de lecciones podremos encontrar los siguientes elementos:

- **Título** de la lección
- **Descripción** de la lección
- Botón **Comenzar lección**.
- Botón **Anterior lección**.
- Botón **Siguiente lección**.
- Botón **Volver al menú**.

Mediante los botones de *anterior* y *siguiente lección* podremos navegar entre las diferentes lecciones cargadas en el sistema. Haciendo uso de los paneles de *título* y *descripción* nos informaremos sobre la lección elegida. Una vez que tengamos claro la lección que queremos tomar, pulsaremos en el botón *comenzar lección*.

Una vez que decidamos comenzar la lección, el sistema ocultará los elementos del menú de selección de lecciones mediante animaciones y, posteriormente, cargará y mostrará los elementos de la lección elegida.

**Lección 1:**  
**Las notas: Do grave**

Para tocar un Do Grave, es decir, el do que está más abajo en la partitura, debes cerrar todos los orificios de la flauta.

♩

◀ Volver al menú

Figura B.9: Pantalla de lección de ejemplo

Las lecciones se muestran en forma de conjunto de elementos multimedia – imágenes y texto – de fácil comprensión, que el usuario podrá leer y estudiar de forma independiente. En futuras versiones de la aplicación se podrán utilizar lecciones de varias etapas, así como integrar sonidos y otro tipo de multimedia.

## B.5. Sección – canciones

El usuario podrá acceder a esta sección desde el menú principal, pulsando en la botón *Canciones*, o con la tecla 2 del teclado.

Al acceder, desaparecerá el menú principal y se mostrará el menú de selección de canciones.



Figura B.10: Pantalla del menú de selección de canciones

Esta pantalla consta de los siguientes botones:

- **Anterior canción**, representado con una flecha hacia arriba.
- **Siguiente canción**, representado con una flecha hacia abajo.
- **Comenzar canción**, simbolizado con la palabra *OK*.
- **Volver**.

Mediante los botones *anterior* y *siguiente canción*, el usuario podrá elegir el tema a interpretar con la flauta. Una vez esté resaltada la canción correcta, el usuario deberá pulsar el botón *Comenzar canción – OK* para que dé comienzo la interpretación.

También es posible pulsar el botón *volver* para ir de nuevo al menú principal.

### B.5.1. Interpretación de la canción

Al lanzar la canción, aparecerá la pantalla de interpretación de canción, que contiene numerosos elementos importantes para el jugador.

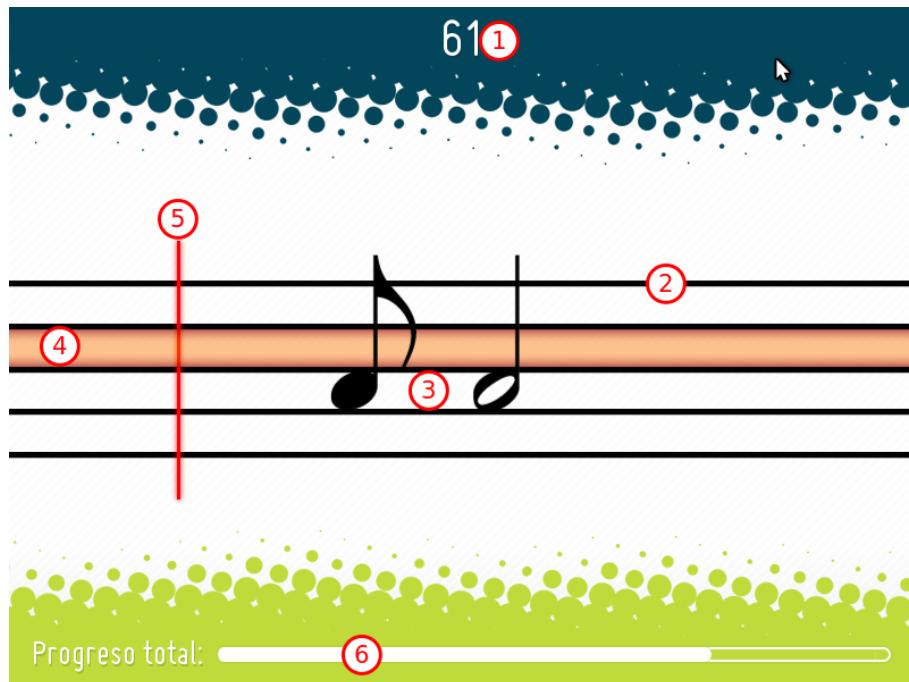


Figura B.11: Pantalla de interpretación de canción

1. **Marcador** de la puntuación del usuario.
2. **Pentagrama**. Sobre él aparecerán las notas.
3. **Notas**. Éstas irán apareciendo por el lado derecho del pentagrama, y moviéndose hacia la izquierda de forma ordenada y rítmica.
4. **Resaltado de nota**. Resalta la posición en el pentagrama de la nota que está tocando el usuario con la flauta en cada momento. Esto ayuda visualmente a saber si estamos interpretando la nota correcta o no.
5. **Barra de interpretación**. Esta barra indica cuándo debe el usuario empezar a tocar la nota.
6. **Barra de progreso**. Indica el progreso de la interpretación de la canción. Cuando la barra esté completa, la canción concluirá.

La forma de juego es sencilla. Como se ha indicado, las notas aparecerán sobre el pentagrama por el lado derecho, e irán desplazándose hacia la izquierda. Una vez que una nota llegue a la *barra de interpretación*, el jugador deberá tocar con la flauta la nota correcta, de forma constante hasta que llegue el turno de la siguiente nota.

Para ayudar al jugador, la barra de *resaltado de nota* indicará en cada momento qué nota se está tocando. Así, si en un instante es necesario tocar un *Sol* pero la

barra de resaltado está por encima, el usuario sabrá que debe cerrar más orificios de la flauta.

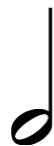
Las posibles **notas** que pueden aparecer son *Do, Re, Mi, Fa, Sol, La, Si, Do grave* y *Re grave*. Las posibles figuras que podrán aparecer son las siguientes:

**Redonda** – Dura cuatro tiempos.



Figura B.12: Figura musical – Redonda

**Blanca** – Dura dos tiempos.



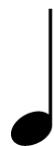
(a) Posición normal



(b) Posición invertida

Figura B.13: Figura musical – Blanca

**Negra** – Dura un tiempo.



(a) Posición normal



(b) Posición invertida

Figura B.14: Figura musical – Negra

**Corchea** – Dura la mitad que una negra.



(a) Posición normal



(b) Posición invertida

Figura B.15: Figura musical – Corchea

De manera excepcional, es posible alargar el tiempo de una figura mediante el uso del **puntillo**, que hará que su duración aumente la mitad de su tiempo original. El puntillo se representa mediante un pequeño círculo a la derecha de la base de la nota.

Por otro lado, las siguientes son las figuras que representan los **silencios**. Funcionan igual que las notas normales, solo que el jugador, en lugar de tocar una nota, deberá mantenerse en silencio desde que el silencio toque la barra de interpretación hasta que llegue la siguiente figura.

**Silencio de redonda** – Dura cuatro tiempos, y se coloca unido, por debajo, a la cuarta línea del pentagrama.



Figura B.16: Silencio de redonda

**Silencio de blanca** – Dura dos tiempos, y se coloca posado sobre la tercera línea del pentagrama.



Figura B.17: Silencio de blanca

**Silencio de negra** – Dura lo mismo que una negra.



Figura B.18: Silencio de negra

**Silencio de corchea** – Dura la mitad que una negra.



Figura B.19: Silencio de corchea

El final de la interpretación vendrá indicado por la doble barra final. Una vez que ésta llegue a la zona de interpretación, se dará por concluida la canción.



Figura B.20: Doble barra de final de pentagrama

### B.5.2. Resultados de la interpretación

Tras la interpretación de la pieza, aparecerá la zona de puntuaciones.

En esta pantalla, además del **título** y **subtítulo** de la canción, se mostrará el **porcentaje de aciertos** que ha tenido el jugador en su interpretación. Además, aparecerá una frase de apoyo que variará según el éxito de la partida.

Una vez llegados a este punto, el jugador puede volver a la pantalla anterior pulsando la tecla escape.



Figura B.21: Pantalla de resultados tras la interpretación



# Apéndice C

## GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc.

<http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

### Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document “free” in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

### 1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited

in duration, to use that work under the conditions stated herein. The “**Document**”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as “**you**”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A “**Modified Version**” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A “**Secondary Section**” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “**Invariant Sections**” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The “**Cover Texts**” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “**Transparent**” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “**Opaque**”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated

HTML, PostScript or PDF produced by some word processors for output purposes only.

The “**Title Page**” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

The “**publisher**” means any person or entity that distributes copies of the Document to the public.

A section “**Entitled XYZ**” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “**Acknowledgements**”, “**Dedications**”, “**Endorsements**”, or “**History**”.) To “**Preserve the Title**” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

## **2. VERBATIM COPYING**

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

## **3. COPYING IN QUANTITY**

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document’s license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as

long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

## 4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the

public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover

Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

## 5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled “History” in the various original documents, forming one section Entitled “History”; likewise combine any sections Entitled “Acknowledgements”, and any sections Entitled “Dedications”. You must delete all sections Entitled “Endorsements”.

## 6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

## 7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works

permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

## **8. TRANSLATION**

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

## **9. TERMINATION**

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your

rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

## 10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy’s public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

## 11. RELICENSING

“Massive Multiauthor Collaboration Site” (or “MMC Site”) means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A “Massive Multiauthor Collaboration” (or “MMC”) contained in the site means any set of copyrightable works thus published on the MMC site.

“CC-BY-SA” means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

“Incorporate” means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is “eligible for relicensing” if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

## ADDENDUM: How to use this License for your

## documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright © YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled “GNU Free Documentation License”.

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the “with ... Texts.” line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.



# Bibliografía y referencias

- [1] *Adobe Photoshop*. <http://www.adobe.com/es/products/photoshop.html>.
- [2] *Amazon just lost \$4.8M after going down for 40 minutes*. <http://www.geekwire.com/2013/amazon-lost-5m-40-minutes/>.
- [3] *Android Operating System*. <http://www.android.com/>.
- [4] *Android Studio IDE*. <http://developer.android.com/sdk/installing/studio.html>.
- [5] *Apple iOS 7*. <http://www.apple.com/es/ios/>.
- [6] *Bottle: Python Web Framework*. <http://bottlepy.org/>.
- [7] *Cascading Style Sheets Snapshot 2010*. <http://www.w3.org/TR/CSS/>.
- [8] *Celery: Distributed Task Queue*. <http://www.celeryproject.org/>.
- [9] *Compass, an open-source CSS Authoring Framework*. <http://compass-style.org/>.
- [10] *ComScore Reports January 2014, US, Smartphone Subscriber Market Share*. [http://www.comscore.com/Insights/Press\\_Releases/2014/3/comScore\\_Reports\\_January\\_2014\\_US\\_Smartphone\\_Subscriber\\_Market\\_Share](http://www.comscore.com/Insights/Press_Releases/2014/3/comScore_Reports_January_2014_US_Smartphone_Subscriber_Market_Share).
- [11] *Cron, task scheduler*. [http://es.wikipedia.org/wiki/Cron\\_\(Unix\)](http://es.wikipedia.org/wiki/Cron_(Unix)).
- [12] *D3.js – Data-Driven Documents*. <http://d3js.org/>.
- [13] *Django, the web framework for perfectionists with deadlines*. <http://www.djangoproject.com>.
- [14] *draw.io, a free online diagram drawing application*. <http://draw.io>.
- [15] *Fabric*. <http://www.fabfile.org/>.
- [16] *Facebook to Acquire Whatsapp for \$19B*. <http://newsroom.fb.com/news/2014/02/facebook-to-acquire-whatsapp/>.
- [17] *Flask, a microframework for Python*. <http://flask.pocoo.org/>.
- [18] *Git, a distributed version control system*. <http://git-scm.com/>.
- [19] *GNU Emacs*. <http://www.gnu.org/software/emacs/>.

- [20] *Google Apps for Business.* <http://www.google.com/enterprise/apps/business/>.
- [21] *Google Cloud Messaging for Android.* <http://developer.android.com/google/gcm/index.html>.
- [22] *Grunt: The JavaScript Task Runner.* <http://gruntjs.com/>.
- [23] *Hetzner vServer VQ7.* [http://www.hetzner.de/en/hosting/produkte\\_vserver/vq7](http://www.hetzner.de/en/hosting/produkte_vserver/vq7).
- [24] *HTML 5 Nightly Draft.* <http://www.w3.org/html/wg/drafts/html/master/Overview.html>.
- [25] *HttpBin, HTTP Client Testing Service.* <http://httpbin.org>.
- [26] *JavaScript.* <http://en.wikipedia.org/wiki/JavaScript>.
- [27] *JoseTomasTocino's GitHub - Forja de SiteUp.* <https://github.com/JoseTomasTocino/pfc-ii>.
- [28] *jQuery – write less, do more.* <http://jquery.com/>.
- [29] *MySQL, The world's most popular open source database.* <http://mysql.com>.
- [30] *Nginx web server.* <http://nginx.org>.
- [31] *node.js.* <http://nodejs.org/>.
- [32] *Open Sans.* <http://opensans.com/>.
- [33] *pip, a tool for installing Python packages.* <http://www.pip-installer.org/>.
- [34] *Python, the official website.* <https://www.python.org/>.
- [35] *Sass: Syntactically Awesome Style Sheets.* <http://sass-lang.com/>.
- [36] *South.* <http://south.aeracode.org/>.
- [37] *SQLite.* <https://sqlite.org/>.
- [38] *Start Developing iOS applications.* <https://developer.apple.com/library/iOS/referencelibrary/GettingStarted/RoadMapiOS/index.html>.
- [39] *Sublime Text: The text editor you'll fall in love with.* <http://sublimetext.com>.
- [40] *VirtualEnv, a tool to create isolated Python environments.* <http://www.virtualenv.org/>.
- [41] *VirtualEnvWrapper, a set of extensions to VirtualEnv.* <http://virtualenvwrapper.readthedocs.org/>.
- [42] *Why IDC predicts Windows Phone will surpass iOS by 2016.* <http://www.wired.com/2012/06/why-idc-predicts-windows-phone-will-surpass-ios-by-2016/>.
- [43] *Wikipedia, Denial-of-service attack.* [http://en.wikipedia.org/wiki/Denial-of-service\\_attack](http://en.wikipedia.org/wiki/Denial-of-service_attack).