



ESCUELA SUPERIOR DE INGENIERÍA

Ingeniería Informática

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

José Tomás Tocino García

Cádiz, 16 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, 16 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 SiteUp al completo:

Índice general

Índice general	9
Índice de figuras	13
Índice de cuadros	15
1. Introducción	17
1.1. Contexto social	17
1.2. Motivación	18
1.3. Objetivos	18
1.3.1. Funcionales	19
1.3.2. Transversales	19
1.4. Estado del arte	20
1.5. Alcance	20
1.5.1. Limitaciones del proyecto	21
1.5.2. Licencia	21
1.6. Estructura del documento	22
1.7. Acrónimos	23
2. Planificación	25
2.1. Metodología de desarrollo	25
2.1.1. Primera iteración: aprendizaje preeliminar	25
2.1.2. Segunda iteración: herramientas básicas de chequeo	26
2.1.3. Tercera iteración: inicio de proyecto Django y CRUD básico	26
2.1.4. Cuarta iteración: integración del motor de tareas asíncronas	27
2.1.5. Quinta iteración: desarrollo de la aplicación Android	27
2.2. Planificación temporal	27
2.3. Organización	27

2.4. Recursos inventariables	29
2.5. Costes	29
2.6. Riesgos	30
2.6.1. Compromiso de la plataforma de despliegue	30
2.6.2. Limitación de recursos	31
2.6.3. Rechazo de los servicios vigilados	31
3. Requisitos del sistema	33
3.1. Objetivos del sistema	33
3.2. Catálogo de requisitos	34
3.2.1. Requisitos funcionales	34
3.2.2. Requisitos de información	38
3.2.3. Requisitos no funcionales	41
3.3. Alternativas de solución	42
3.3.1. Frameworks para el desarrollo de la plataforma web	42
3.3.2. Sistemas de gestión de bases de datos	43
3.3.3. Sistemas para la ejecución asíncrona de tareas	44
4. Análisis	45
4.1. Metodología	45
4.2. Modelo conceptual	45
4.2.1. Usuario	45
4.2.2. Grupo de chequeos	47
4.2.3. Chequeo	47
4.2.4. Registro de chequeo	47
4.2.5. Chequeo - Ping	47
4.2.6. Chequeo - Port	48
4.2.7. Chequeo - DNS	48
4.2.8. Chequeo - HTTP	48
4.3. Modelo de casos de uso	48
4.3.1. Actores	48
4.3.2. Subsistema de gestión de usuarios	49
4.3.3. Subsistema de gestión de grupos de chequeos	51
4.3.4. Subsistema de gestión de chequeos	54
4.4. Modelo de comportamiento	58
4.5. Modelo de interfaz de usuario	59
4.5.1. Modelo de navegación	59
4.5.2. Prototipos visuales de la interfaz web	60
5. Diseño	65
5.1. Arquitectura del sistema	65

<i>ÍNDICE GENERAL</i>	11
5.1.1. Arquitectura física	65
5.1.2. Arquitectura lógica	67
5.2. Diseño físico de datos	72
5.2.1. User	72
5.2.2. UserExtra	74
5.2.3. CheckGroup	74
5.2.4. BaseCheck	74
5.2.5. DnsCheck	75
5.2.6. HttpCheck	75
5.2.7. PortCheck	75
5.2.8. PingCheck	76
5.2.9. CheckLog	76
5.2.10. CheckStatus	77
5.2.11. Session	77
5.2.12. MigrationHistory	78
5.2.13. ContentType	78
6. Implementación	79
6.1. Entorno de construcción	79
6.1.1. Aplicación web	79
7. Pruebas	85
7.1. Estrategia	85
7.2. Entorno de pruebas	85
7.3. Roles	86
7.3.1. Desarrollador principal	86
7.3.2. Probadores externos	86
7.4. Niveles de pruebas	87
7.4.1. Pruebas unitarias	87
7.4.2. Pruebas de integración	87
7.4.3. Pruebas de sistema	87
7.4.4. Pruebas de aceptación	88
7.5. Implementación de pruebas	88
8. Conclusiones	89
A. Manual de instalación de la plataforma web	91
A.1. Instalación inicial	91
A.1.1. Descarga de código	91
A.1.2. Entorno virtual y dependencias	92
A.1.3. Creación de la base de datos	92

A.1.4. Instalación del servidor	93
A.1.5. Instalación del bróker de mensajes	93
A.1.6. Ejecución de los servicios	93
A.1.7. Configuración del servidor frontal	94
A.1.8. Lanzamiento final	95
A.2. Mejora del rendimiento	95
B. Manual de usuario	97
C. GNU Free Documentation License	99
1. APPLICABILITY AND DEFINITIONS	100
2. VERBATIM COPYING	102
3. COPYING IN QUANTITY	102
4. MODIFICATIONS	103
5. COMBINING DOCUMENTS	105
6. COLLECTIONS OF DOCUMENTS	105
7. AGGREGATION WITH INDEPENDENT WORKS	106
8. TRANSLATION	106
9. TERMINATION	107
10. FUTURE REVISIONS OF THIS LICENSE	107
11. RELICENSING	108
ADDENDUM: How to use this License for your documents	108
Bibliografía y referencias	111

Índice de figuras

2.1. Diagrama de Gantt	28
4.1. Modelo conceptual	46
4.2. Diagrama de actores	49
4.3. Casos de uso del subsistema de gestión de usuarios	52
4.4. Casos de uso del subsistema de gestión de grupos de chequeos	52
4.5. Casos de uso del subsistema de gestión de chequeos	55
4.6. Modelo de navegación de la aplicación web	59
4.7. Modelo de navegación de la aplicación Android	60
4.8. Home	61
4.9. Login	61
4.10. Recuperación de contraseña	62
4.11. Registro de usuario	62
4.12. Dashboard	63
4.13. Detalle de chequeo	63
4.14. Creación de grupo de chequeos	64
4.15. Formulario genérico de creación de chequeo	64
5.1. Arquitectura lógica del sistema web	68
5.2. Diseño de la base de datos	73

Índice de cuadros

3.1. OBJ-1	34
3.2. OBJ-2	34
3.3. RQF-1	34
3.4. RQF-2	35
3.5. RQF-3	35
3.6. RQF-4	35
3.7. RQF-5	35
3.8. RQF-6	36
3.9. RQF-7	36
3.10.RQF-8	36
3.11.RQF-9	36
3.12.RQF-11	37
3.13.RQF-11	37
3.14.RQF-12	37
3.15.RQF-13	37
3.16.RQF-14	38
3.17.IRQ-1	38
3.18.IRQ-2	38
3.19.IRQ-3	39
3.20.IRQ-4	39
3.21.IRQ-5	40
3.22.IRQ-6	40
3.23.IRQ-7	40
3.24.IRQ-8	41
3.25.NRQ-1	41
3.26.NRQ-2	41
3.27.NRQ-3	41
3.28.NRQ-4	42
3.29.NRQ-5	42

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 [4].

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) tenía un dominio personalizado y estaba gestionada a través de Google Apps [5]. 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 *párking*¹, 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.

¹Una página de *párking de dominio* sirve como página temporal mientras un servicio web no es lanzado, de forma que el dominio no esté en blanco.

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.
- 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 el sistema operativo Android para la recepción instantánea de avisos provenientes de la aplicación web.

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** [3], el framework de desarrollo web en el que se basa la aplicación, se distribuye bajo la licencia *BSD*.

- El servidor web **nginx** [7] 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, así como las principales operaciones con sus diagramas de secuencia y contratos.
- **Diseño.** Seguido del análisis, se expone en detalle la etapa de diseño del sistema, con los diagramas de clases.
- **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:

- **??,** donde detallamos qué hemos usado para la elaboración del proyecto.
- **Manual de instalación de la plataforma web** del proyecto en sistemas nuevos.
- **Manual de usuario,** donde se explica cómo usar la aplicación.

1.7. Acrónimos

PSFL Python Software Foundation License

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

HTTP Hyper Text Transfer Protocol

ICMP Internet Control Message Protocol

MIT Massachusetts Institute of Technology

MVC Model View Controller

PDF Portable Document Format

URL Uniform Resource Locator

Capítulo 2

Planificación

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. En esta sección se detallan las etapas y otros detalles relativos a la planificación del proyecto.

2.1. Metodología de desarrollo

Para la realización del proyecto se ha utilizado un modelo de desarrollo iterativo incremental. En la redacción del presente documento se presentarán la fase de investigación preliminar y las etapas de análisis, diseño, implementación y pruebas del proyecto en su estado final. A continuación se detallan cada una de las iteraciones por las que ha ido pasando el proyecto.

2.1.1. Primera iteración: aprendizaje preeliminar

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 barajaron las diferentes posibilidades de implementación del sistema, así como las posibles herramientas y bibliotecas de terceros que pudieran ser de ayuda.

En particular, en esta etapa se decidió el uso del framework web **Django**, las tecnologías de desarrollo *front-end* ¹ (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: 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

¹El desarrollo web *front-end* hace referencia a la parte de una web con la que el usuario interactúa directamente en el navegador.

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.

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).

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

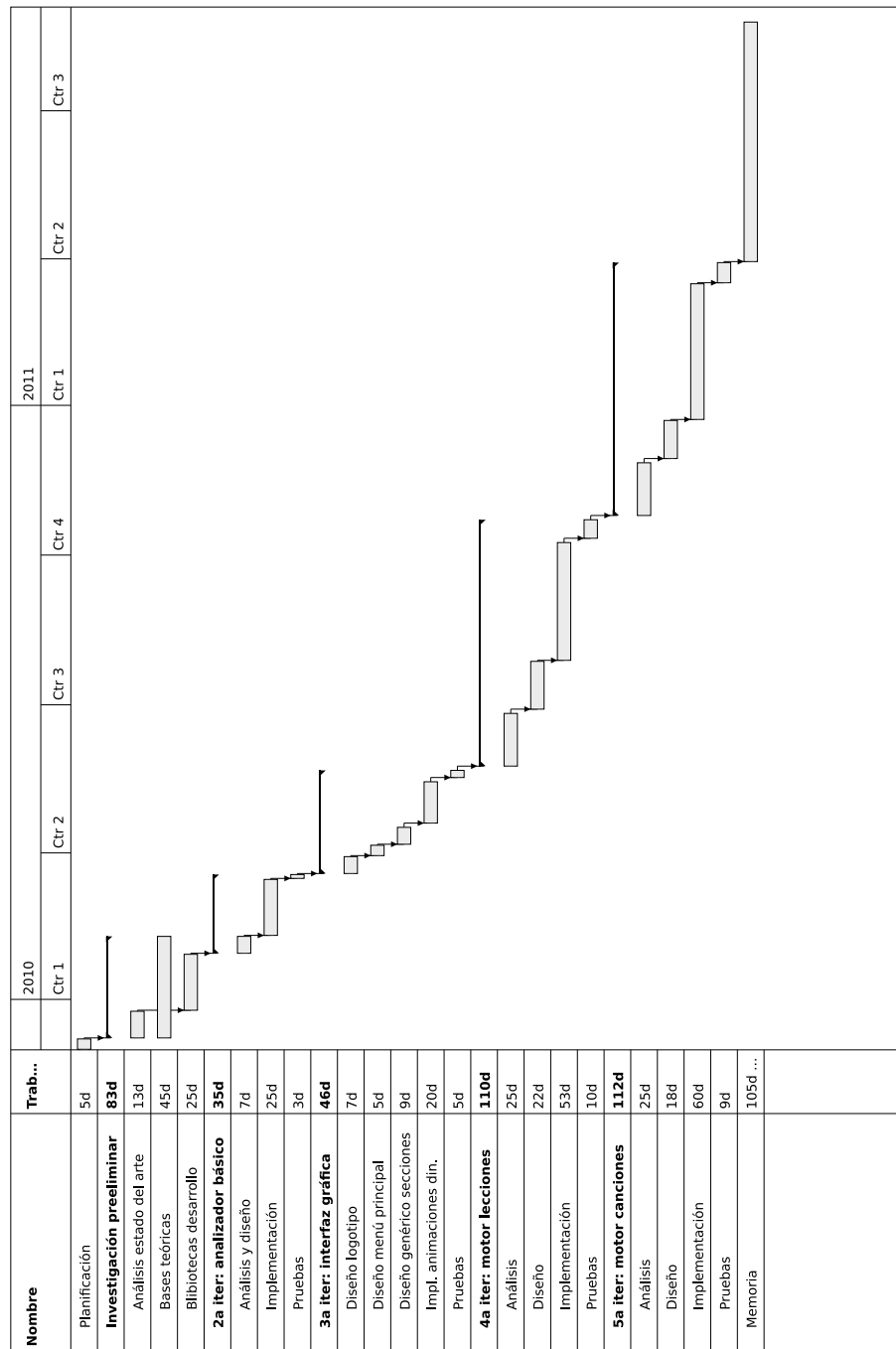


Figura 2.1: Diagrama de Gantt

proyecto. El desarrollo del proyecto ha sido revisado y guiado de forma continua por el tutor Ivan 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
 - 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

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 [6], 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 [8].

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 chequeos 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 chequeos.

Cuadro 3.4: RQF-2

Gestión de grupos de chequeos

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

Cuadro 3.5: RQF-3

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

Cuadro 3.7: RQF-5

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

Cuadro 3.8: RQF-6

Gestión de chequeos

Título	Creación de chequeos
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 chequeos
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 chequeos
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 chequeos
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 chequeos
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 chequeos 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 chequeos 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"> ■ Nombre de usuario ■ Dirección de correo electrónico ■ Contraseña ■ (<i>Opcional</i>) Identificador de dispositivo Android

Cuadro 3.17: IRQ-1

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

Cuadro 3.18: IRQ-2

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

Cuadro 3.19: IRQ-3

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

Cuadro 3.20: IRQ-4

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

Cuadro 3.21: IRQ-5

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

Cuadro 3.22: IRQ-6

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

Cuadro 3.23: IRQ-7

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

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:** 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 vasta comunidad de usuarios.

Los proyectos Django se organizan en *apps*, que permiten modularizar la funcionalidad. Integra un **ORM!** (**ORM!**), un sistema de plantillas y una sección de administración adaptable a cualquier sitio web.

- **Flask:** 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:** 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**, 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:** 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** es un demonio ¹ 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** 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!** (**AMQP!**)) 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 añadirm también de forma manual, el proceso que lance 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.

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

Capítulo 4

Análisis

4.1. Metodología

SiteUp ha seguido una metodología de desarrollo ágil en la que, mediante fases de desarrollo rápidas y ligeras, se intenta evitar los formales caminos de las metodologías tradicionales, enfocándose en las personas y obteniendo así resultados en etapas más tempranas del desarrollo.

En las sucesivas secciones y capítulos se detallará el proceso de análisis y posteriores fases del proyecto en la última iteración del proyecto. Así se consigue una documentación más concisa y cercana al producto final.

4.2. Modelo conceptual

De acuerdo a lo reflejado en el apartado 3.2.2 se presenta el diagrama 4.1 en el que se identifican las clases, sus atributos y las relaciones entre ellas.

4.2.1. Usuario

Nombre Nombre elegido por el usuario.

Correo electrónico Dirección de correo electrónico.

Contraseña Contraseña para el login del usuario.

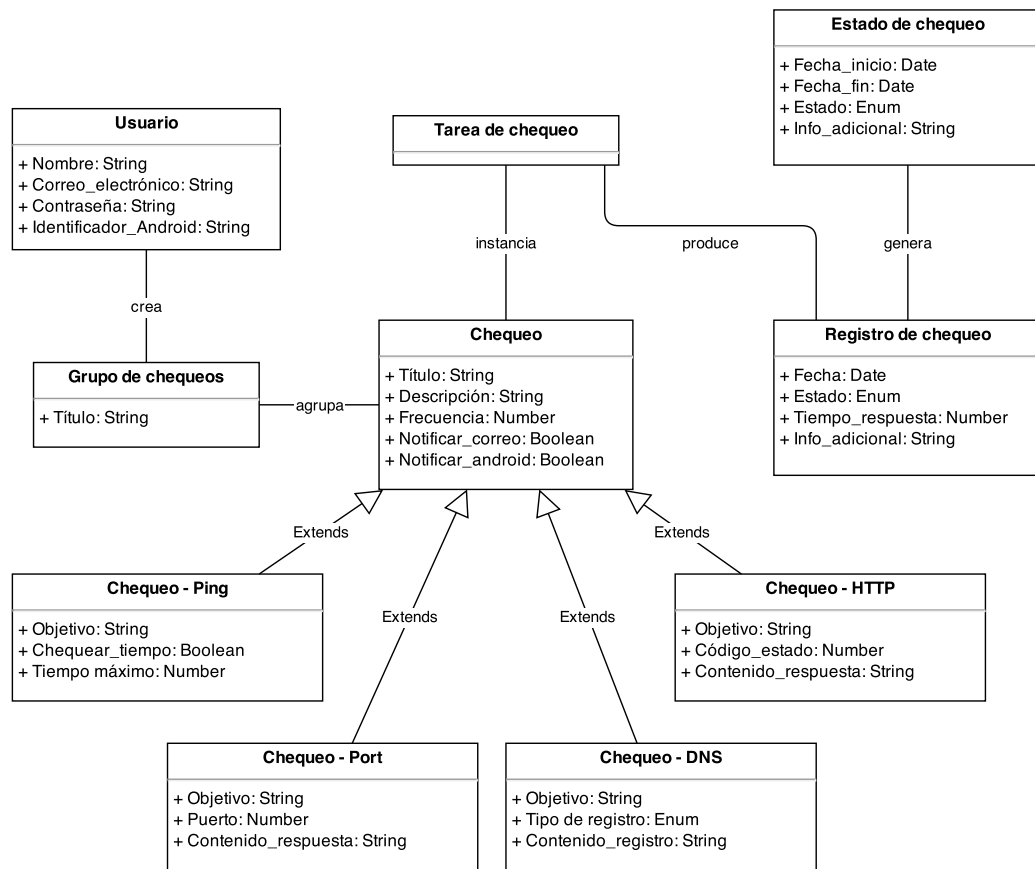


Figura 4.1: Modelo conceptual

Identificador Android Identificador único del dispositivo Android del usuario.

4.2.2. Grupo de chequeos

Título Título para identificar al grupo de chequeos.

4.2.3. 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.

4.2.4. 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*.

4.2.5. 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.

4.2.6. 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.

4.2.7. 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.

4.2.8. 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.

4.3. Modelo de casos de uso

4.3.1. Actores

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

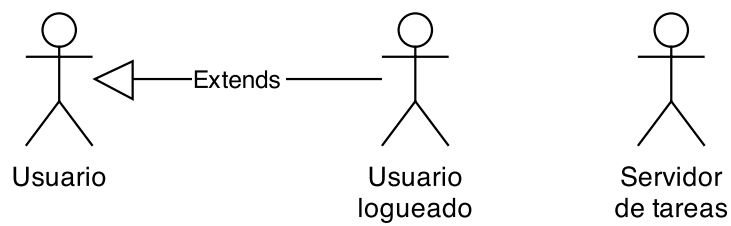


Figura 4.2: Diagrama de actores

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.

4.3.2. Subsistema de gestión de usuarios

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

Caso de uso: registro de usuario

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

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.

4.3.3. Subsistema de gestión de grupos de chequeos

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

Caso de uso: creación de grupo de chequeos

Descripción Un usuario logueado decide crear un grupo de chequeos.

Actores *Usuario logueado.*

Escenario principal

1. El *usuario* decide crear un nuevo grupo de chequeos.

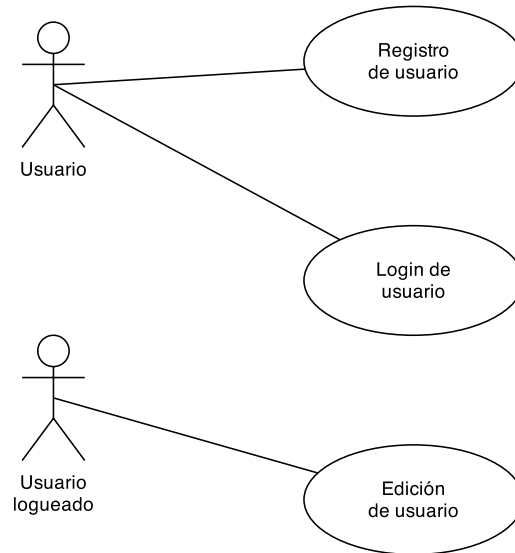


Figura 4.3: Casos de uso del subsistema de gestión de usuarios

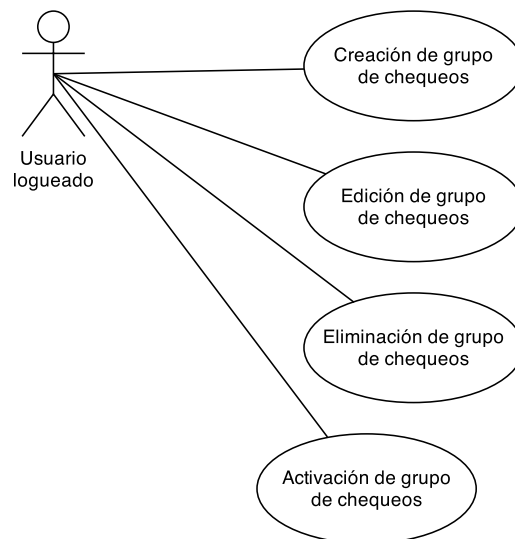


Figura 4.4: Casos de uso del subsistema de gestión de grupos de cheques

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 chequeos.
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 chequeos asociados al grupo.
6. El *sistema* elimina el grupo de chequeos.

Flujo alternativo

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

Caso de uso: activación de grupo de chequeos

Descripción Un usuario logueado decide activar un grupo de chequeos.

Actores *Usuario logueado*

Escenario principal

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

Flujo alternativo

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

4.3.4. Subsistema de gestión de chequeos

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

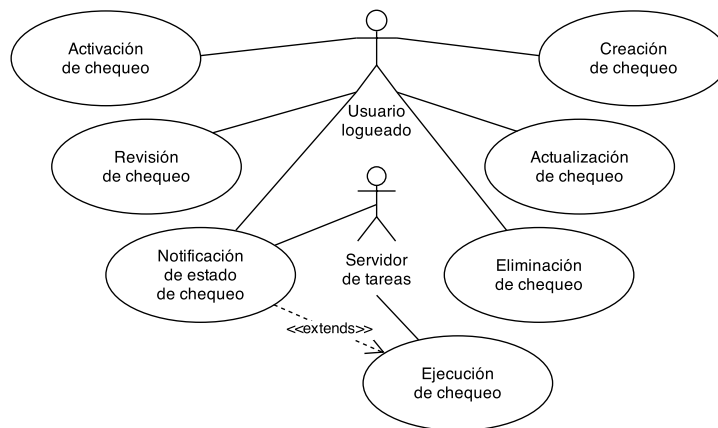


Figura 4.5: Casos de uso del subsistema de gestión de chequeos

Caso de uso: creación de chequeo

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

Actores *Usuario logueado*

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 rellenos 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.

4.4. 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 los contratos de estas operaciones.

4.5. Modelo de interfaz de usuario

4.5.1. 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

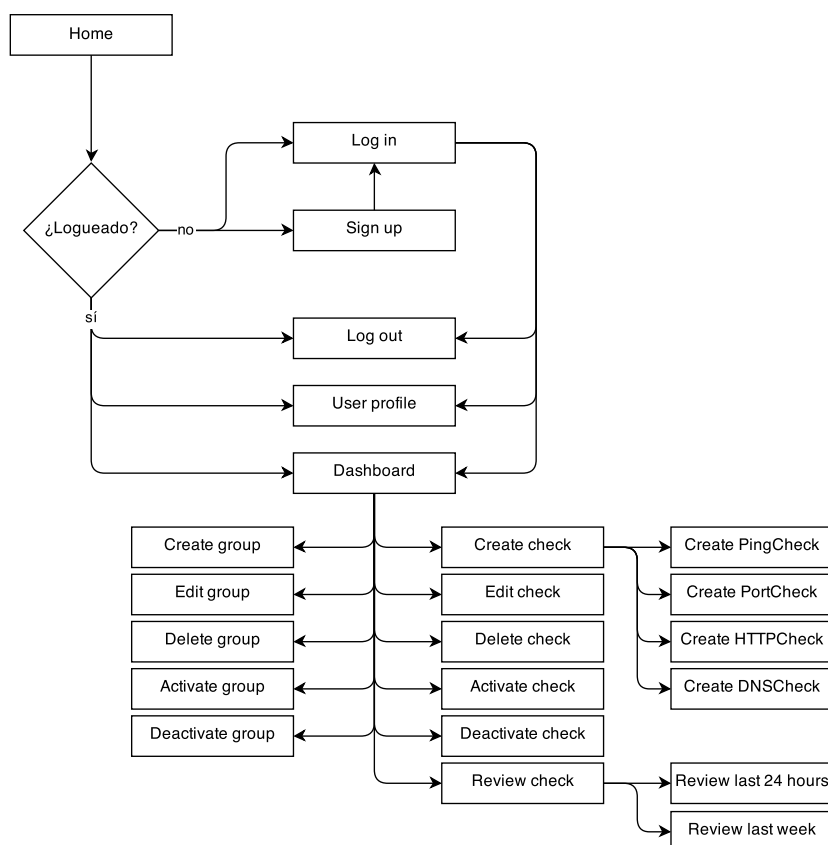


Figura 4.6: Modelo de navegación de la aplicación web

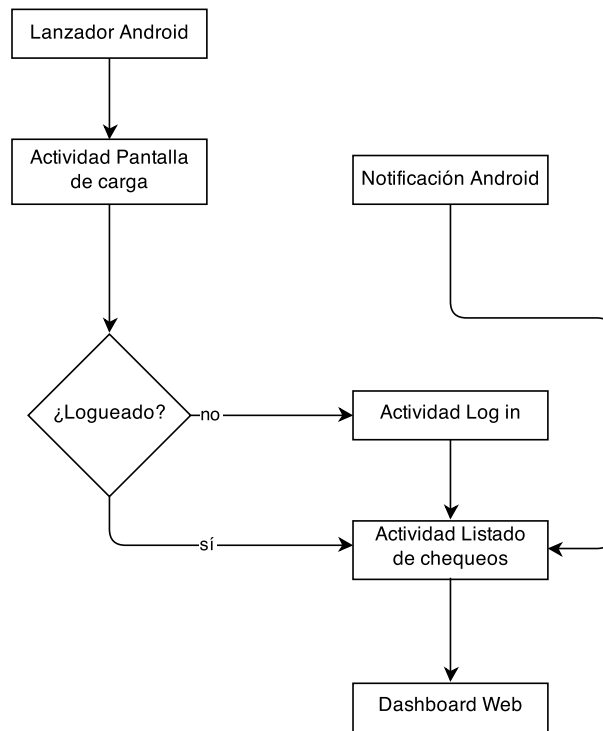


Figura 4.7: Modelo de navegación de la aplicación Android

4.5.2. Prototipos visuales de la interfaz web

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

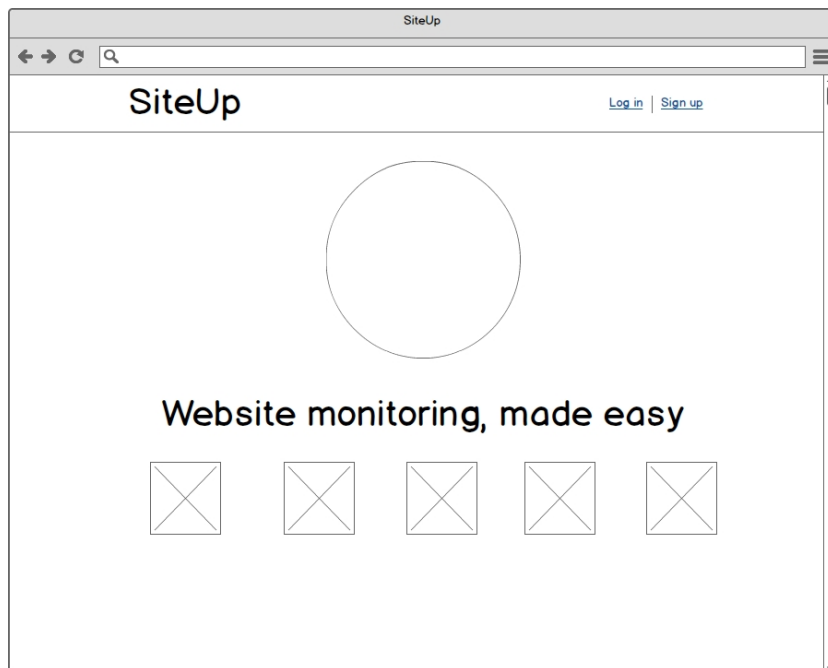


Figura 4.8: Home

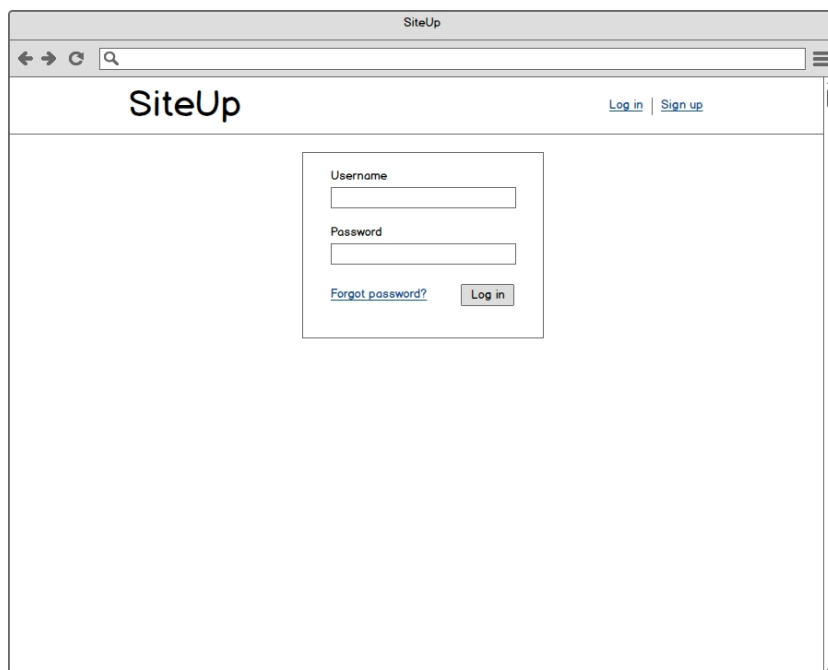
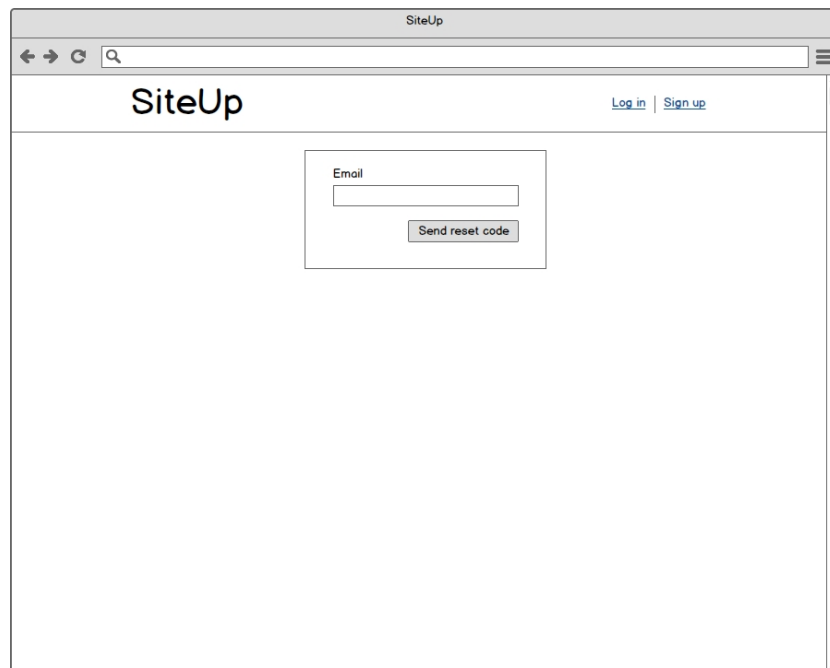


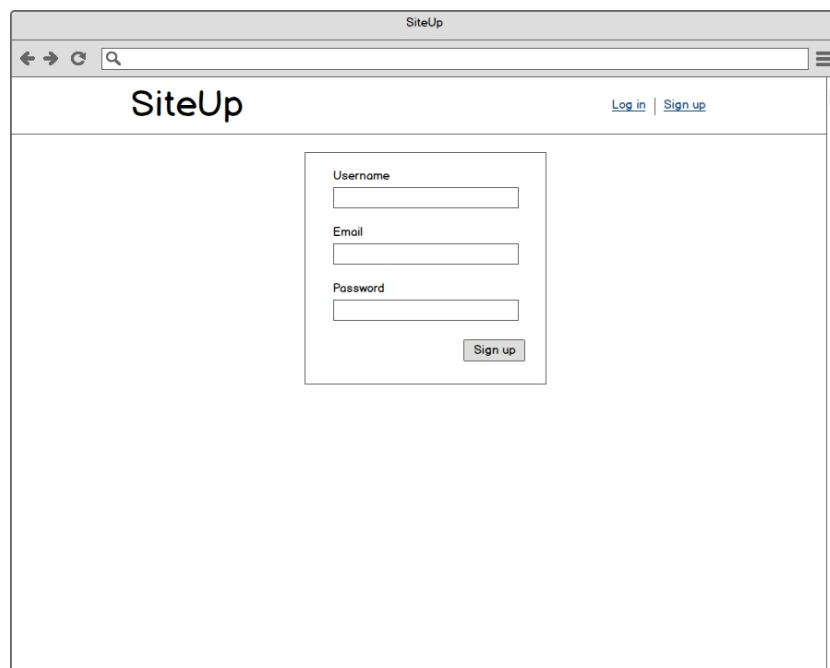
Figura 4.9: Login



The screenshot shows a web browser window titled "SiteUp". The browser's address bar is empty. The page header features the "SiteUp" logo on the left and "Log in" and "Sign up" links on the right. The main content area contains a centered form with the following elements:

- A label "Email" above a text input field.
- A "Send reset code" button located below the input field.

Figura 4.10: Recuperación de contraseña



The screenshot shows a web browser window titled "SiteUp". The browser's address bar is empty. The page header features the "SiteUp" logo on the left and "Log in" and "Sign up" links on the right. The main content area contains a centered form with the following elements:

- A label "Username" above a text input field.
- A label "Email" above a text input field.
- A label "Password" above a text input field.
- A "Sign up" button located below the input fields.

Figura 4.11: Registro de usuario

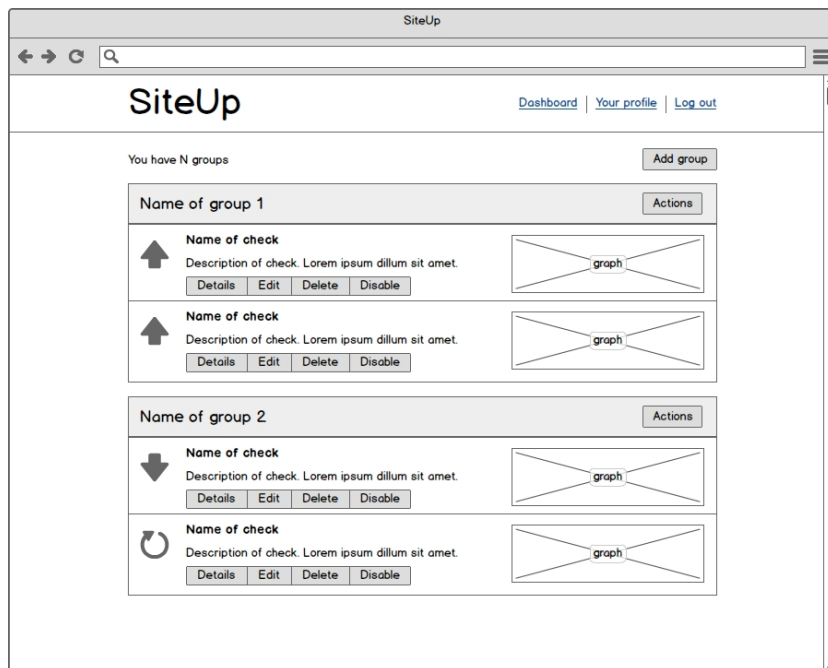


Figura 4.12: Dashboard

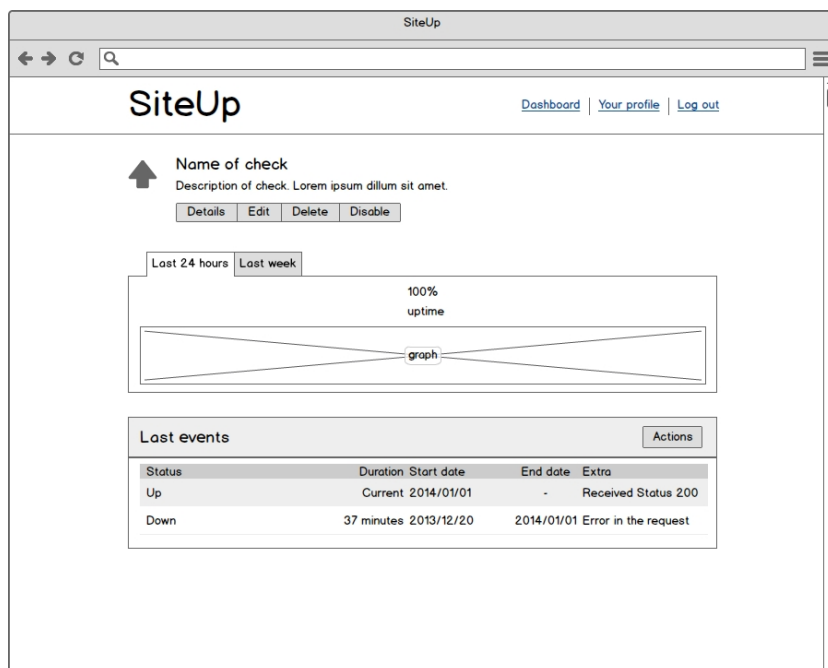
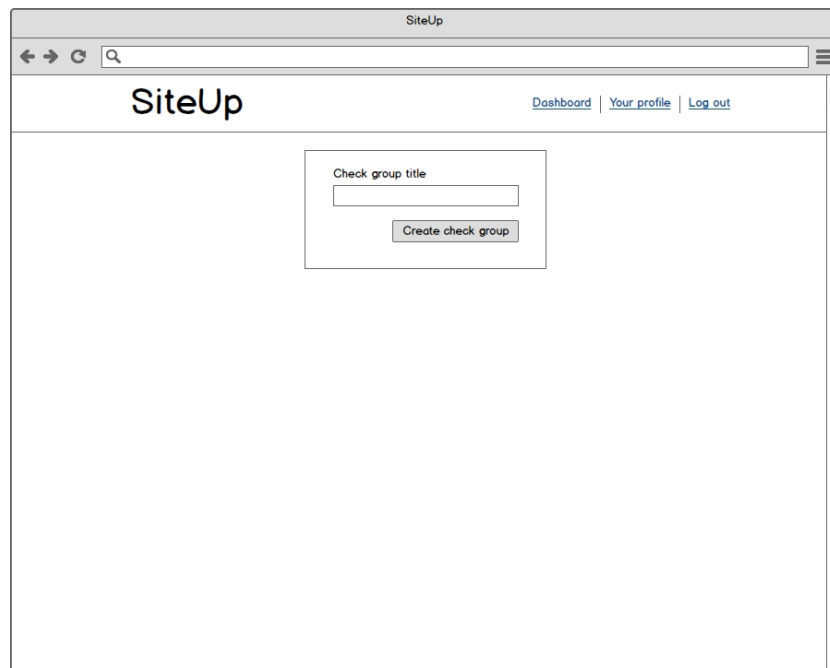
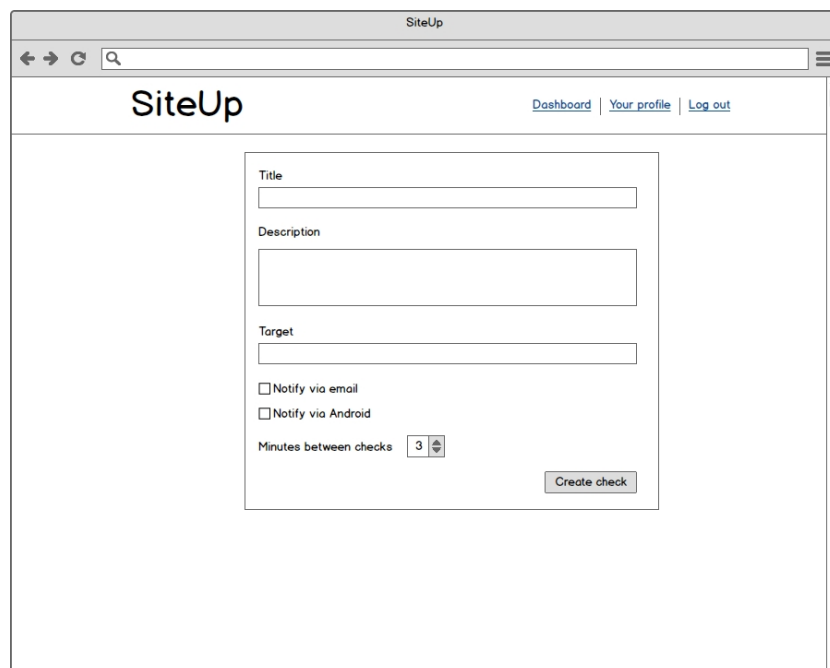


Figura 4.13: Detalle de chequeo



The screenshot shows a web browser window with the title 'SiteUp'. The address bar contains a search icon. The page header includes the 'SiteUp' logo and navigation links: 'Dashboard', 'Your profile', and 'Log out'. The main content area features a form titled 'Check group title' with a single text input field and a 'Create check group' button.

Figura 4.14: Creación de grupo de chequeos



The screenshot shows the same web browser window as Figure 4.14, but with a more detailed form titled 'Create check'. The form includes the following fields and options: 'Title' (text input), 'Description' (text input), 'Target' (text input), 'Notify via email' (checkbox), 'Notify via Android' (checkbox), and 'Minutes between checks' (a numeric input set to 3 with a spinner icon). A 'Create check' button is located at the bottom right of the form.

Figura 4.15: Formulario genérico de creación de chequeo

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 antes de 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.

Una vez satisfechos los anteriores requisitos, el proyecto SiteUp instalará una serie de dependencias propias, de forma encapsulada dentro de un entorno virtual. Algunas de las dependencias más importantes son:

- **Django**, framework web en el que se basa el proyecto.
- **Celery**, cola de tareas asíncronas.
- **Fabric**, servicio para la creación de scripts de despliegue en servidores remotos.
- **Gunicorn**, servidor para aplicaciones web desarrolladas en Python basadas en el estándar WSGI.
- Bibliotecas auxiliares para el lanzamiento de diversos tipos de chequeos en red: **requests**, **dnspython** y **urllib**.

De igual modo para el desarrollo del lado *front-end* de la web existen ciertas dependencias, instalables en forma de módulos de Node JS. Las dependencias más importantes son:

- **Sass**, lenguaje de hojas de estilo que amplía las funcionalidades de CSS. El código Sass compila a código CSS válido.
- **Compass**, framework para Sass que añade numerosas funciones y *mixins* para facilitar el desarrollo front-end.

- **Grunt**, un lanzador que automatiza la ejecución de tareas comunes y tediosas escrito en JavaScript.

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.

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.

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

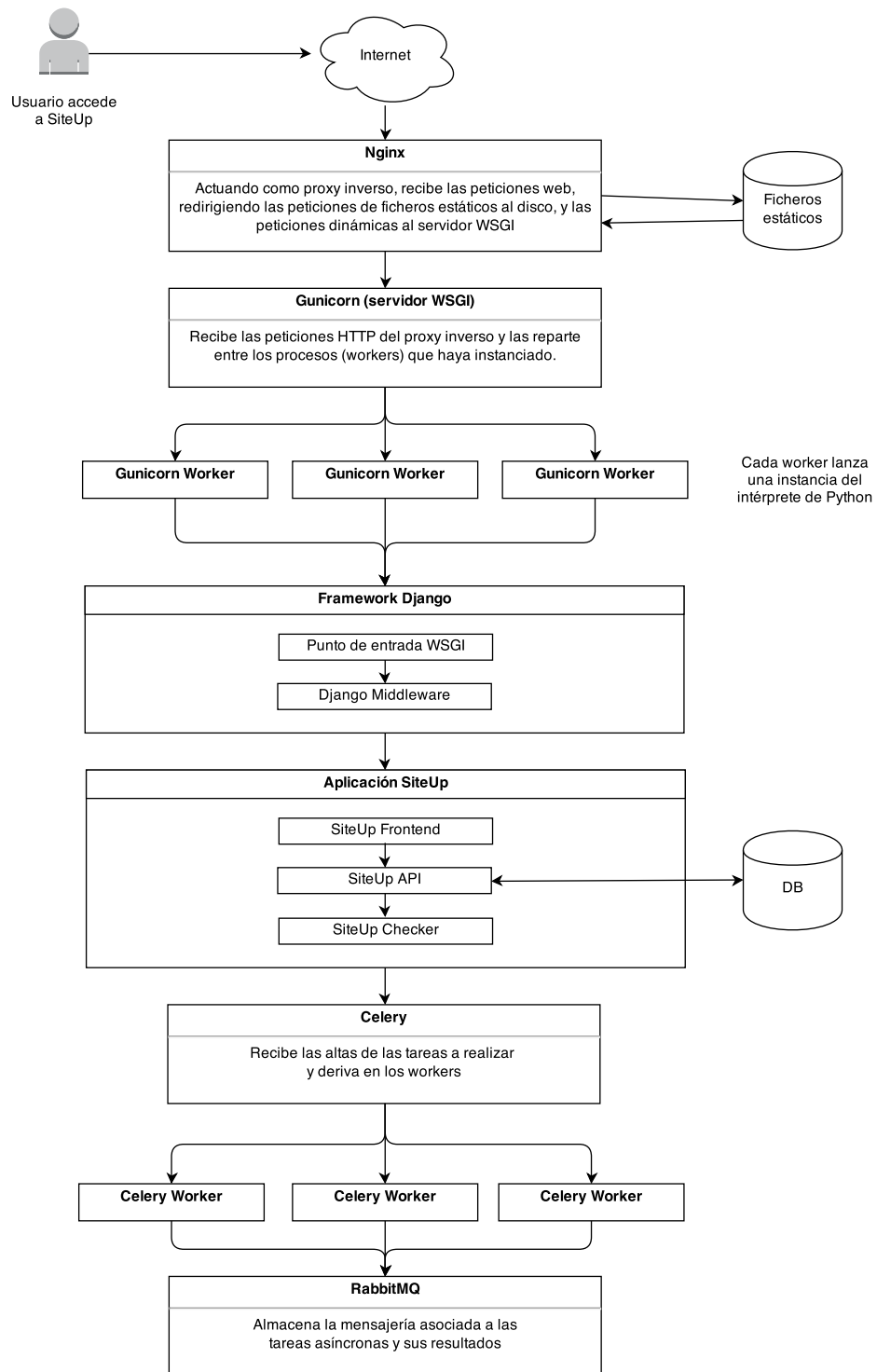


Figura 5.1: Arquitectura lógica del sistema web

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 procesamiento 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! (WSGI!)**, 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 serie 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, mánagers, 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 *bróker* de mensajes, que implementa el ya mencionado estándar **AMQP!**. El bróker 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.

5.2. Diseño físico de datos

En la figura 5.2 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 predeterminado de Django y forma parte de la aplicación `django.contrib.auth`.

Sus principales campos son:

- `username` - Nombre de usuario.
- `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.

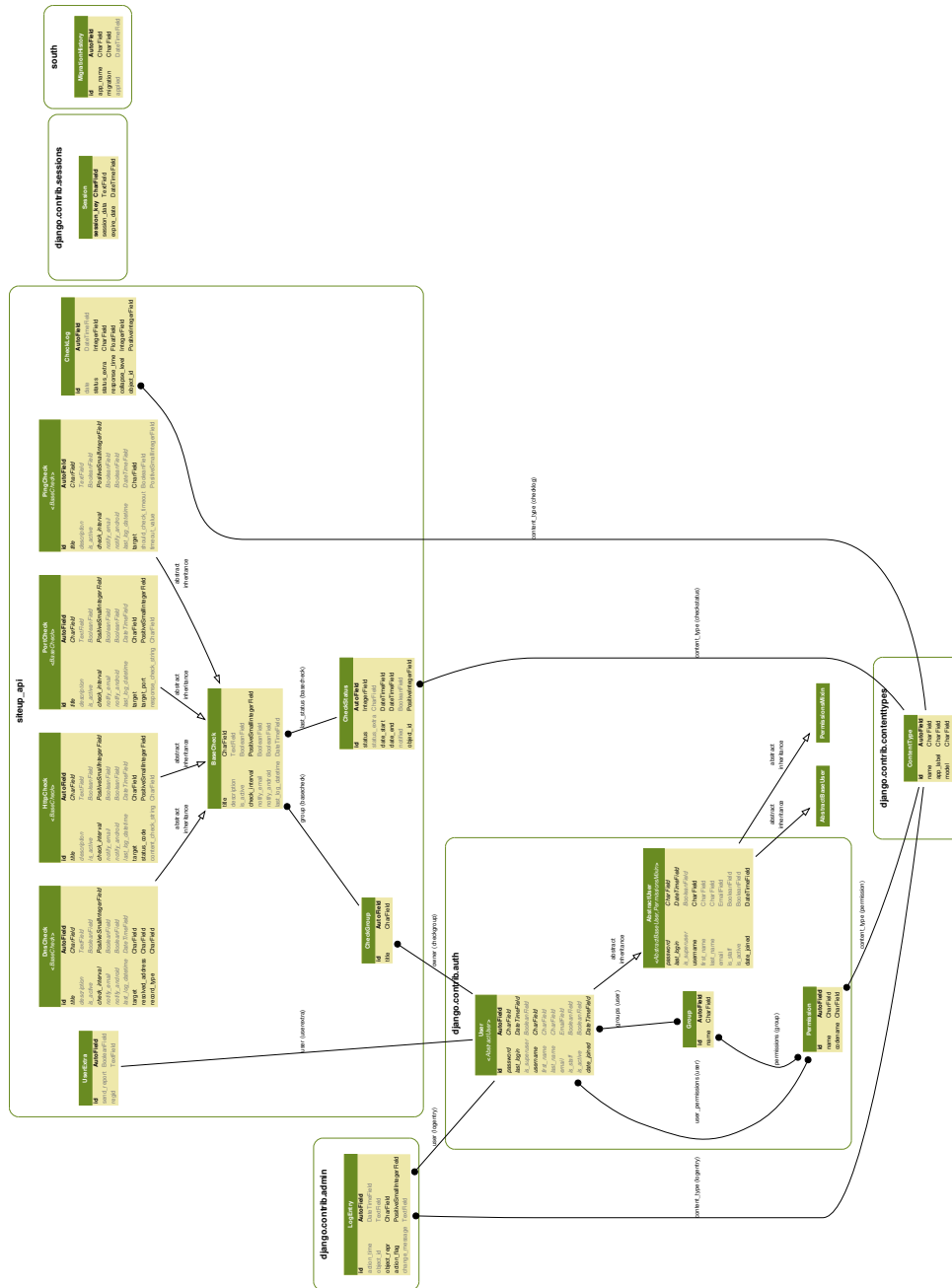


Figura 5.2: Diseño de la base de datos

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 chequeos.
- `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 chequeos. Todos los chequeos 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 chequeos. 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.

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 [?].

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** [?], 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** [?], que sigue siendo la mejor opción a la hora de editar documentos de \LaTeX .

Se ha utilizado **draw.io** [?] 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** [?] y **VirtualEnvWrapper** [?]. 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** [?] 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** [?], haciendo uso de los planes gratuitos. GitHub es una forja que utiliza el sistema de control de versiones **Git** [?]. 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** [?], 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** [?], 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 puedan 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.

Dependencias de Django

En el proyecto se han utilizado un gran número

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** [?], 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 periodos 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 asertos definidos o por otro error no controlado.

Capítulo 8

Conclusiones

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 [?].

A.1. Instalación inicial

A.1.1. Descarga de código

Para instalar la plataforma web es necesario clonar el repositorio [?] 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 unicorn 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 unicorn y el servidor de tareas celery:

```
sudo service supervisor 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

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. Secondly, 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 Li-

cense.

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]

[2] *Amazon just lost \$4.8M after going down for 40 minutes.* <http://www.geekwire.com/2013/amazon-lost-5m-40-minutes/>.

[3] *Django, the web framework for perfectionists with deadlines.* <http://www.djangoproject.com>.

[4] *Facebook to Acquire Whatsapp for \$19B.* <http://newsroom.fb.com/news/2014/02/facebook-to-acquire-whatsapp/>.

[5] *Google Apps for Business.* <http://www.google.com/enterprise/apps/business/>.

[6] *Hetzner vServer VQ7.* http://www.hetzner.de/en/hosting/produkte_vserver/vq7.

[7] *Nginx web server.* <http://nginx.org>.

[8] *Wikipedia, Denial-of-service attack.* http://en.wikipedia.org/wiki/Denial-of-service_attack.