ESCUELA SUPERIOR DE INGENIERÍA

Ingeniería Informática

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

José Tomás Tocino García

Cádiz, 14 de abril de 2014

ESCUELA SUPERIOR DE INGENIERÍA

Ingeniería Técnica en Informática de Sistemas

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

DEPARTAMENTO: Lenguajes y Sistemas Informáticos.
DIRECTOR DEL PROYECTO: Antonio García Domínguez y Manuel Palomo Duarte.
AUTOR DEL PROYECTO: José Tomás Tocino García.

Cádiz, 14 de abril de 2014

Fdo.: José Tomás Tocino García

Agradecimientos

Quiero agradecer y dedicar la presente memoria y, por extensión, el proyecto SiteUp al completo:

Índice general

Índice de figuras

1. Introducción

1.1. Contexto y motivación

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.

Se pone pues de manifiesto la importancia de la fiablidad 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 [**?**].

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. Objetivos

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

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

1.2.1. 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.

1.2.2. 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.

1.3. 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.

1.3.1. 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 cuenta con ninguna funcionalidad para gestionar los chequeos de un usuario, sino que sirve para recibir notificaciones instantáneas provenientes de la aplicación web. Ésta, por otro lado, está 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 han limitado la aplicación a una estructura monolítica en la que los chequeos se hacen desde una sola máquina, la misma que sirve el servicio web.

1.3.2. 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 framework de desarrollo web en el que se basa la aplicación es **Django** [**?**], que utiliza la licencia *BSD (Berkeley Software Distribution)* [1].
- El servidor web **nginx** [**?**] también utiliza la licencia *BSD*.
- Los siguientes paquetes de Python utilizan también la licencia *BSD*:
  - Celery
  - Sqlparse
  - iPython
  - dnspython
  - coverage
  - django-rest-framework

[1] https://github.com/django/django/blob/master/LICENSE

    - 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

1.4. Estructura del documento

El presente documento se rige según la siguiente estructura:

- **Introducción**. Se exponen 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**, que explica las labores de documentación y experimentación previas al desarrollo, que han servido para labrar una base de conocimientos que nos diera las suficientes garantías para afrontar el proyecto.
- **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:

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

1.4.1. Acrónimos

BSD Berkeley Software Distribution

CRUD Create-Read-Update-Delete

DNS Domain Name Server

FDL Free Documentation License

FSF Free Software Foundation

GPL General Public License

HTTP Hyper Text Transfer Protocol

ICMP Internet Control Message Protocol

MIT Massachusetts Institute of Technology

MVC Model View Controller

URL Uniform Resource Locator

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).

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

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 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).

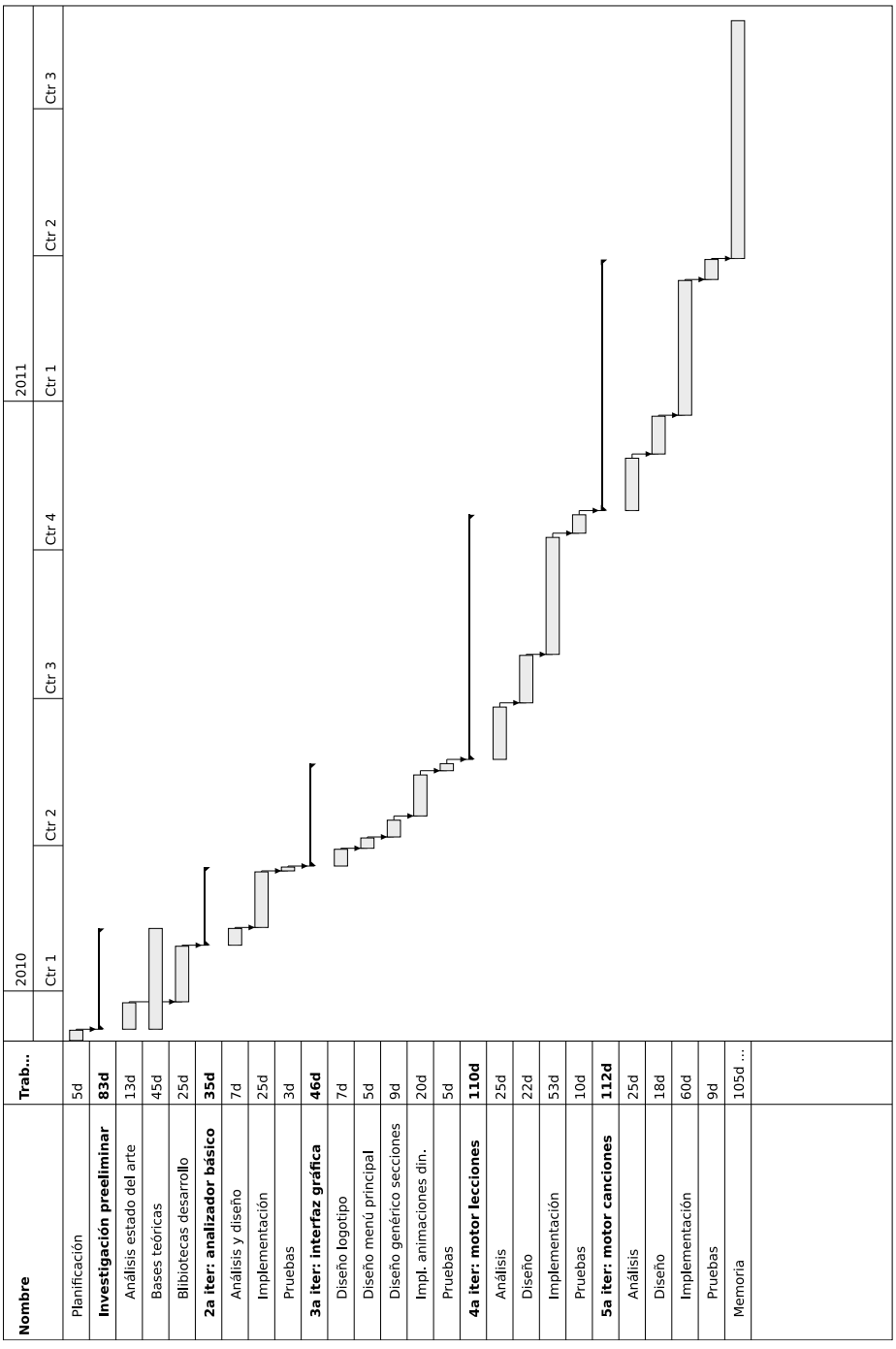

Figura 2.1.: Diagrama Gantt de iteraciones

2.3. Organización

El proyecto ha sido desarrollado en su totalidad por el que escribe, alumno de la Universidad de Cádiz, llevando a cabo las labores de análisis, diseño y desarrollo, así como de diseño visual de las interfaces y *branding* del proyecto. El desarrollo del proyecto ha sido revisado y guiado de forma continua por el tutor 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 [**?**], 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 sistema están configurados para detectar esta clase de comprobaciones y bloquearlas, al identificarlas (a veces erróneamente) como ataques de denegación de servicio [**?**].

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.

3. Requisitos del sistema

3.1. 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.

3.1.1. Motivación personal

En lo personal, la idea de desarrollar este proyecto surgió a raíz de una necesidad personal. Durante un importante proceso de selección que estaba gestionando por correo electrónico, 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. Las alertas que tenía puestas para verificar que mi web estaba operativa no dieron la alarma, ya que el dominio ahora apuntaba a

un espacio de párking genérico que no generaba ningún código de error, pero los correos electrónicos no se estaban enrutando correctamente. Esta situación se prolongó durante varios días, en los cuales perdí varios correos importantes

Esto se podría haber detectado rápidamente si hubiera tenido un chequeo que verificase que los registros DNS apuntaban a las IP correctas. A partir de ahí, se hizo un estudio de mercado, cuyas conclusiones se han reflejado en la sección anterior, y dadas las obvias limitaciones de las alternativas existentes se decidió elaborar un proyecto nuevo.

3.2. Objetivos del sistema

Los objetivos principales de SiteUp son:

Título	Crear chequeos de múltiples 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

3.3. Catálogo de requisitos

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

3.3.1. 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: *Ping check*, *Port check*, *DNS Check* y *HTTP Check*. 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

3.3.2. Requisitos de información

	Usuario del sistema
Datos específicos	▪ Nombre de usuario ▪ Dirección de correo electrónico ▪ Contraseña ▪ (*Opcional*) Identificador de dispositivo Android

Cuadro 3.17.: IRQ-1

	Chequeo tipo DNS
Datos específicos	▪ 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 Ping
Datos específicos	▪ 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	▪ Título ▪ Descripción ▪ Objetivo – Debe ser una IP o un hostname ▪ Puerto objetivo ▪ *Opcional* - 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	▪ Título ▪ Descripción ▪ Objetivo – Debe ser una URL HTTP ▪ Código de estado HTTP ▪ *Opcional* - 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	▪ 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	▪ 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

3.3.3. 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 3.24.: 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 3.25.: 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 3.26.: 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 3.27.: 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 3.28.: NRQ-5

3.3.4. Alternativas de solución

En esta subsección se presentan las diferentes alternativas existentes para cubrir las necesidades del proyecto SiteUp.

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 basta 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.

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.

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

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

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.

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 **??** se presenta un diagrama en el que se identifican las clases

5. Diseño

6. Implementación

7. Pruebas

8. Conclusiones

A. Herramientas utilizadas

B. Manual de instalación

C. Manual de usuario

D. GNU Free Documentation License