UNIVERSIDAD DE ALCALÁ

Escuela Politécnica Superior

GRADO EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE GRADO

Desarrollo de un portal Web de administración con Django y una aplicación Android cliente.

SILVIA GARCÍA BAUTISTA

2015

UNIVERSIDAD DE ALCALÁ

Escuela Politécnica Superior

GRADO EN INGENIERÍA INFORMÁTICA

Trabajo Fin de Grado

Desarrollo de un portal Web de administración con Django y una aplicación Android cliente.

Autora: Silvia García Bautista

Director: Antonio Moratilla Ocaña

TRIBUNAL:

Presidente: ……

Vocal 1º: ……..

Vocal 2º: ……..

CALIFICACIÓN: ……

FECHA: ……..

A ella, mi luz...

"Pensar sin aprender es esfuerzo perdido;
aprender sin pensar, peligroso".

Confucio

Agradecimientos

A Stack Overflow y Google, fuente de toda sabiduría.

A mi familia, por apoyarme siempre.

A Antonio Moratilla, por su paciencia y vocación.

Pero sobre todo a Dani, por ser quien es y estar a mi lado.

Índice general

Índice de figuras

Índice de tablas

Parte I

Resumen

Norextin es una organización de ámbito europeo cuyo núcleo de negocio reside en el mantenimiento de centros de datos independientes para el alojamiento de equipos TIC. Es una empresa ficticia que se propone como prototipo de potencial cliente para este proyecto. El resultado de este desarrollo es un producto software ideal para servir a modelos de negocio similares al que aquí se propone.

Puestos a suponer, se considera que Norextin abarca una gran cuota de mercado, goza de página web y está presente en las redes sociales. Pero le surge la necesidad de mejorar los servicios prestados y cuidar a sus clientes desarrollando una aplicación móvil. Una App que ponga a disposición de los usuarios un nuevo canal de comunicación en el que proporcionar de manera sencilla y al alcance de la mano la información de los servicios en el CPD.

Este proyecto tiene como objetivo estudiar, analizar, diseñar y construir una solución que satisfaga dichas necesidades.

Estudiar qué tecnologías web y móviles existen en la actualidad, cuáles son sus ventajas e inconvenientes, cuánto se usan y cómo se podrían llegar a conectar entre ellas de la manera más beneficiosa para el sistema.

Analizar todas esas tecnologías y tomar decisiones. Explicar por qué se elige Django como módulo de administración web y por qué se elige Android como plataforma de desarrollo móvil. En esta fase también se materializan los requisitos del cliente en diagramas de casos de uso, vitales para la siguiente fase.

Diseñar, de acuerdo a lo estudiado y analizado, toda la arquitectura tanto del servidor web como de la aplicación móvil incluyendo los correspondientes diseños de interfaz.

Y por último construir, codificar, implementar y subir a producción el sistema completo. La solución completa.

Palabras clave: Android, Python, Django, Web, servicio REST, portal de administración.

Parte II

Memoria del trabajo

Capítulo 1

Introducción

1.1 Presentación del proyecto

Los centros de datos son un espacio exclusivo que las empresas utilizan para mantener las infraestructuras TIC que gestionan su actividad empresarial. En ellos se alojan los servidores y sistemas de almacenamiento donde se ejecutan las aplicaciones y se almacena el contenido. Hay empresas a las que les basta con un simple bastidor, mientras que otras necesitan una o varias salas privadas donde alojar un determinado número de bastidores, depende del tamaño de la empresa.

Cada vez más, las empresas deben tener la capacidad de enfrentarse a nuevas tecnologías, aumentos exponenciales en volúmenes de datos, y complejidad de los mismos. Para ello las compañías buscan métodos flexibles y escalables que gestionen las infraestructuras TIC sin comprometer la seguridad ni la fiabilidad. Por ello externalizar dicha gestión en un proveedor de alojamiento constituye la mejor solución.

En este proyecto se va a suponer una empresa especialista en centros de datos independientes para el alojamiento de equipos TIC. Una organización llamada Norextin de ámbito europeo y líder en su sector, que facilita la entrega segura de aplicaciones de misión crítica y contenidos al usuario final con excelente tiempo de respuesta.

Con el fin de crecer y poner a disposición de los usuarios un nuevo canal de comunicación fácil y cómodo para acceder a la información de los servicios en el CPD, Norextin quiere lanzar una App para gestionar el centro de datos. Gracias a esta aplicación, los clientes deben poder conocer todos los servicios que tengan contratados:

- El número, la ubicación y la identificación de los bastidores.
- La potencia eléctrica disponible.
- El cableado de datos (en uso y disponible).
- Los operadores de telecomunicaciones con los que están interconectados.

Permitir agilizar también algunos procesos como las peticiones de parcheo de circuitos de datos, que se deberán poder solicitar directamente a través de la aplicación.

También acceder a contenido relacionado con el sector de centros de datos y a información general de la compañía, como informes de las instalaciones, *whitepapers*, lista de operadores y certificaciones. La aplicación además de facilitar este contenido, dará la posibilidad de se pueda compartir con otros usuarios.

Para desarrollar todo este sistema, debe realizarse un estudio detallado de las necesidades de Norextin, ahora como cliente de un proveedor de desarrollo software a medida. En este documento se recogen cada una de las fases llevadas a cabo para su desarrollo, desde la toma de requisitos donde se materializan esas necesidades en formato técnico, pasando por la elección de tecnologías que ofrezcan mayores ventajas, el análisis y el diseño del sistema de información.

Pero antes de entrar en detalles técnicos, propios de otras partes de este documento, se va a presentar un 'diccionario' de conceptos que se van a utilizar a lo largo del documento, propios del modelo de negocio de esta empresa:

- **Huella**: O también llamada bastidor, ocupa un espacio específico del centro de datos donde el cliente puede instalar sus equipos. Estos bastidores están en una zona compartida con otros clientes, para aprovechar al máximo el espacio disponible. Cada bastidor cuenta con su propia toma de corriente independiente. El cliente puede instalar sus propios bastidores o solicitarlos a Norextin.

- **Sala**: Ofrecen espacio de uso exclusivo para mayor seguridad y pueden adaptarse a las necesidades específicas del cliente. Las salas privadas están separadas de otras zonas de clientes por tabiques. Los sistemas de alimentación eléctrica (PDU), la refrigeración, los controles de acceso y extinción de incendios son exclusivos para mayor protección.

- **Toma eléctrica**: Se refiere al cableado eléctrico que va por debajo del falso suelo, siempre separado del cableado de datos que va en bandejas fijadas al techo. Norextin ofrece un servicio estructurado de cableado y supervisa el trabajo de los técnicos externos para asegurar un servicio de calidad y flexibilidad para todos los clientes.

- **Circuito**: Este concepto se refiere a los operadores de telecomunicaciones con los que los mismos clientes están interconectados.

- **Documento**: Son archivos en formato PDF en los que Norextin ofrece contenido relacionado con el sector de centros de datos.

- **Noticia**: Norextin es una empresa bastante activa en redes sociales y contiene un blog en su página web donde mantiene a los usuarios al día de las últimas noticias, opiniones y debates.

- **Contacto**: Los contactos son personal perteneciente a la organización disponible para atender al cliente.

1.2 Objetivos

El objetivo principal de este proyecto es el **desarrollo de una aplicación web de administración con Django y una aplicación móvil en Android, para la gestión y administración de centros de procesamiento de datos** alojados en la empresa Norextin.

Para conseguirlo, se plantean una serie de objetivos específicos que se enumeran a continuación.

- **Objetivo 1:** Evaluar los entornos existentes para el desarrollo de este sistema software, identificando qué tecnologías o lenguajes son los más recomendados.

- **Objetivo 2:** Poner en práctica los procesos del ciclo de vida del software en un entorno real laboral.

- **Objetivo 3:** Desarrollar una aplicación web de administración.

- **Objetivo 4:** Desarrollar una aplicación para dispositivos móviles.

- **Objetivo 5:** Generar un API REST que comunique eficazmente ambas aplicaciones.

- **Objetivo 6:** Puesta en producción del sistema completo.

1.3 Estructura de la documentación

En estas líneas, se presenta el Trabajo Fin de Grado el cual se ha estructurado en diferentes partes:

- La primera parte se compone del resumen extendido del proyecto, el cual sintetiza el resto del trabajo.

- La segunda parte presenta el cuerpo del proyecto compuesto por la memoria.

- La tercera parte incluye otras características importantes, que se agrupan bajo el título de apéndices.

- La cuarta parte incorpora las referencias que sostiene los diferentes apartados en los que se ha dividido el proyecto.

La parte más extensa e importante del proyecto, la memoria, se ha dividido a su vez en los siguientes capítulos:

- El primer capítulo se corresponde con la introducción. Ésta incluye una vista general del problema planteado y la solución dada, los objetivos fundamentales del proyecto y la propia estructura del trabajo.

- El segundo capítulo comprende el concepto de estado del arte, con el estudio de las tecnologías móviles, de las aplicaciones web empresariales y las comunicaciones entre ambos mundos, así como el análisis de la situación actual de cada una de ellas.

- El tercer capítulo recolecta la toma de requisitos en un listado que los clasifica en: funcionales, de datos, de seguridad y de interfaz.

- El cuarto capítulo está dedicado a todo el proceso de análisis del sistema a desarrollar. En él se identifican los subsistemas que comprenden el global de la aplicación, se selecciona el *stack* de tecnologías a emplear, se reconocen los riesgos que pueden comprometer el desarrollo del proyecto y se detallan todos los posibles casos de uso para los distintos roles de usuario.

- El quinto capítulo abarca el proceso de diseño de ciclo de vida del software. Aquí se habla de la arquitectura de los componentes, así como de la del despliegue. Se diseña el modelo físico de datos, la arquitectura del servidor y la arquitectura de la aplicación móvil. De igual manera se muestra un primer esbozo de los diseños de las interfaces web y móvil.

- El sexto capítulo se corresponde con la planificación del proyecto, incluyendo un diagrama de Gantt ilustrativo.

- El séptimo capítulo está dedicado a recoger las conclusiones finales del proyecto y las posibles líneas de trabajo futuras.

En la tercera parte, se localizan los apéndices con información extra como el presupuesto del proyecto, cómo montar un API en muy pocos pasos y un manual de instalación en producción del servidor.

Capítulo 2

Estado del arte

Una buena práctica previa a un desarrollo, independientemente de que sea software o no, es estudiar el entorno existente y actual del tema que se va a tratar. Como el propósito general de este proyecto ya se ha expuesto en apartados anteriores, lo que se va a hacer aquí es estudiar el contexto en el que se va a realizar, con objeto de que el lector pueda entender el porqué de algunas de las elecciones que se han ido tomando a lo largo del proceso de desarrollo.

Las aplicaciones web y móviles no son conceptos nuevos a día de hoy, forman parte de la vida cotidiana sin que se repare casi en ello, pero ¿cuál es el estado actual de ambos mercados?, ¿en qué influyen para el desarrollo de este proyecto?, ¿cuáles son las tecnologías líderes y por qué?

2.1 Aplicaciones móviles, un mercado de gran crecimiento.

Tanto en España como en el resto del mundo, el acceso a la tecnología móvil con más capacidades y con un coste menor, ha permitido el desarrollo de un gran número de aplicaciones para prácticamente cualquier uso. Con esta base, las empresas se han volcado en ofrecer a sus clientes servicios sobre una gran variedad de alternativas, dentro de las cuales podemos encontrar:

- Inteligencia de negocio, donde consultar indicadores del desempeño financiero y operativo de una organización.
- Financiero, donde realizar transacciones electrónicas como consulta y transferencia de fondos.

- Entretenimiento, donde a través de las preferencias y los patrones de consumo, es posible sugerir a un cliente diferentes opciones como lo son restaurantes, cines, teatros, taxis, entre otros.
- Servicios de asistencia a los clientes, donde poder solicitar una grúa a través de las capacidades de geolocalización del dispositivo móvil o reportar un siniestro a la aseguradora con la que tenemos contratado el seguro de nuestro coche.

Podrían enumerarse un sin fin de alternativas, sin embargo queda claro que, al ser aplicaciones desarrolladas a medida, el universo de las mismas es tan amplio como de creatividad, necesidades y capacidades de inversión se disponga [1].

Como consumidor a día de hoy, el mundo de las aplicaciones móviles está tan integrado en la rutina diaria, que una aplicación ya no es una novedad, es prácticamente un requisito. Por primera vez en España, el tiempo de acceso a medios digitales desde dispositivos móviles supera al empleado desde el ordenador tradicional, dato que demuestra la irrupción que han protagonizado las mismas como puerta de entrada a Internet.

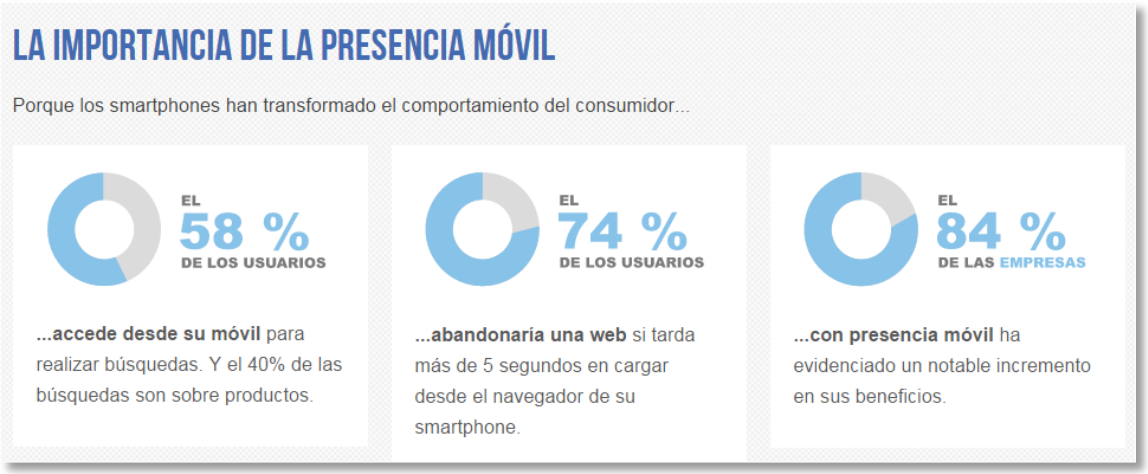


Ilustración 1: Importancia de la presencia móvil.

Dado el universo de aplicaciones que es posible desarrollar, los fabricantes han tomado decisiones de negocio en donde las empresas tienen la difícil tarea de decidir qué camino deben seguir. Esto lleva a establecer criterios como la penetración y popularidad de la plataforma, el tipo de aplicación a desarrollar y elementos propios de un desarrollo móvil como lo son la usabilidad, seguridad y portabilidad.

A día de hoy un sinnúmero de empresas compiten ferozmente por mantener el mayor margen de participación de mercado, siendo los más representativos:

- **Apple**: Con sus dispositivos Ipod, Iphone y Ipad, su plataforma propietaria iOS y Objetive C, además de la tienda Apple Store en donde hoy podemos localizar casi un millón de aplicaciones de todo tipo.

- **Google**: Con su plataforma Android, que en los últimos años ha crecido de una manera muy sólida, mejorando las capacidades de su entorno de desarrollo, además de llevar a cabo acuerdos comerciales con diversos fabricantes como Samsung, Sony y Nokia.

- **Microsoft**: Con Windows Phone y la evolución de Mobile, que ha venido consolidándose a través de acuerdos con diversos fabricantes como Nokia. Una ventaja de su plataforma, es la integración nativa del ambiente Office, Xbox y Explorer; eso además de la integración con el Marketplace en donde es posible acceder a un gran número de aplicaciones de entretenimiento como música y videos.

- **RIM** (Research in Motion): Con Blackberry que si bien en los últimos años ha venido perdiendo una participación de mercado muy importante en Estados Unidos y Europa, en países como México y Latinoamérica aún conserva una base instalada de equipos muy grande que le permite seguir compitiendo activamente.

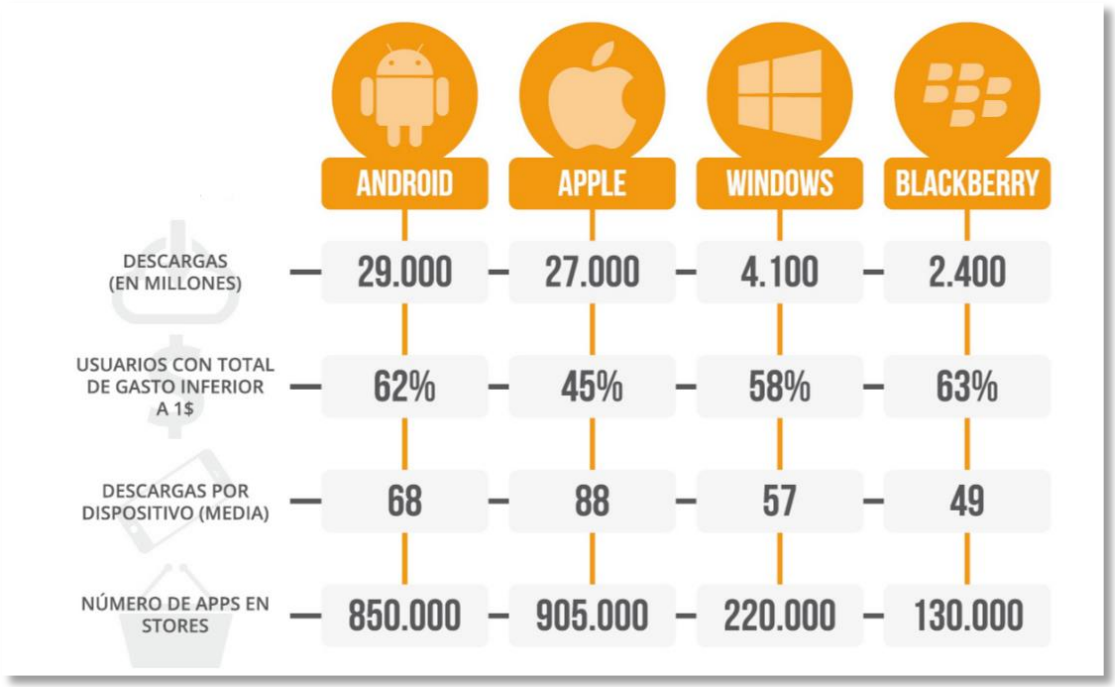


Ilustración 2: Estadísticas por plataforma en la UE [2].

Como se puede apreciar en la figura, hay dos claros ganadores: Android y Apple, aunque entre ambas tecnologías no está tan clara la victoria. Mientras Android alcanza el podio en lo que a millones de descargas se refiere, con 3.000 millones más, en el plano individual de cada dispositivo es Apple quien lidera el ranking con 20 aplicaciones más instaladas por dispositivo.

Aunque si se toma como referencia la cantidad de dispositivos que hay de un sistema operativo y de otro, cambia la visión. Android no sólo alcanza la mayoría absoluta en España, sino que crece hasta el 89'9%, desde el 89% que obtenía en el primer cuatrimestre de 2014. Sin embargo, el reparto es muy diferente en el mercado europeo, donde la cosa se nivela bastante. Android continúa siendo líder, pero su liderazgo no es tan absoluto alcanzando ahora mismo el 68'4%, lo que supone una reducción del 3'1% frente a lo que obtenía en 2014 [3].

EU	3 m/e Mar 2014	3 m/e Mar 2015	% pt. Change
Android	71.1	68.4	-3.1
iOS	18.6	20.3	1.8
Windows	8.1	9.9	1.8
Other	1.8	1.4	-0.5

Spain	3 m/e Mar 2014	3 m/e Mar 2015	% pt. Change
Android	89	89.9	0.9
iOS	7.2	7	-0.2
Windows	3.1	2.8	-0.3
Other	0.7	0.3	-0.4

2.2 Aplicaciones web empresariales

En la actualidad existe un amplio ecosistema de tecnologías y estándares que nos permiten construir una aplicación empresarial basada enteramente en la web. Una de las ventajas de trabajar con tecnologías web es la portabilidad y fácil escalabilidad que tendrá nuestro producto, puesto que lo único que necesita para funcionar es un navegador web.

La elección del software sobre el cual vamos a construir una aplicación dependerá en gran medida de las necesidades de la misma, del lenguaje sobre el que se vaya a desarrollar y de la popularidad que estos tengan entre las comunidades de desarrolladores. Algunos de los *frameworks* más usados en la actualidad los podemos encontrar en la siguiente gráfica.

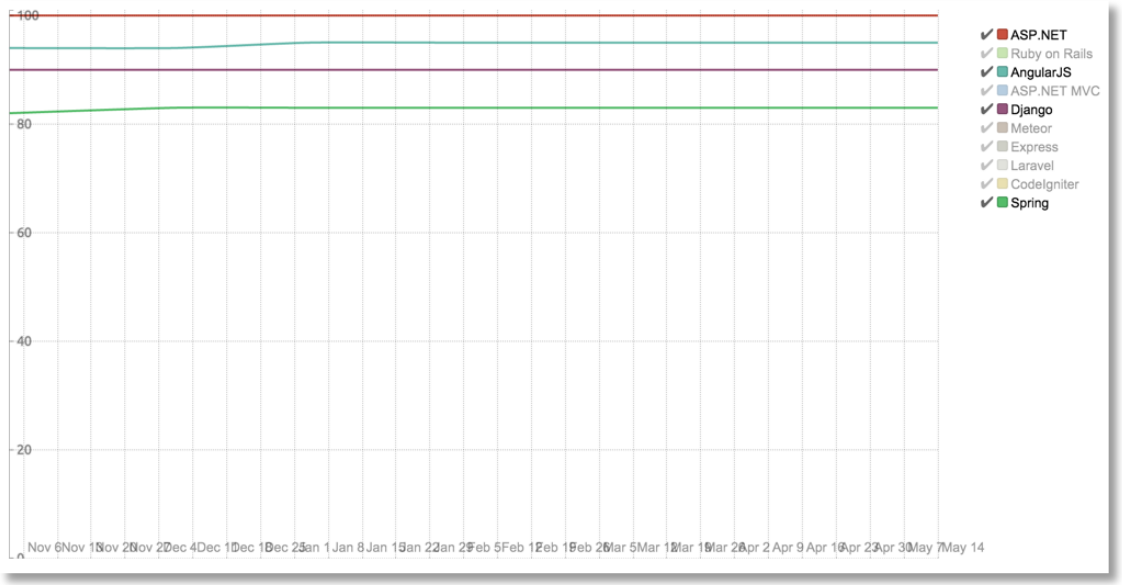


Ilustración 3: Estadísticas de uso de framworks basadas en comunidad de desarrolladores [4].

Cada uno de estos *frameworks* de desarrollo se sustentan con la actividad de las comunidades de desarrolladores que da soporte y evolucionan los productos. Un *framework* sin apoyo de la comunidad es un *framework* muerto. Es por esto que el índice de entradas en sitios clave como *github* o *stackoverflow* son indicadores potentes de su popularidad e impacto en el mercado. Véase en la siguiente imagen.

Rankings

Framework	Github Score	Stack Overflow Score	Overall Score
ASP.NET		100	100
Ruby on Rails	95	98	96
AngularJS	100	91	95
ASP.NET MVC		93	93
Django	89	92	90
Meteor	95	75	85
Express	92	77	84
Laravel	90	79	84
CodeIgniter		84	84
Spring	79	88	83
Ember.js	89	77	83
Symfony	85	82	83

Ilustración 4: Ranking por estrellas/forks en github y preguntas activas en stackoverflow [5].

Analizando estos datos podemos inferir que lenguajes clave como .NET, Java, Python y PHP dominan el mercado de tecnologías para el desarrollo de aplicaciones web empresariales. Si se toma como referencia estos lenguajes se podrá contrastar estos datos con los ofrecidos por estudios independientes en popularidad por entorno de desarrollo.

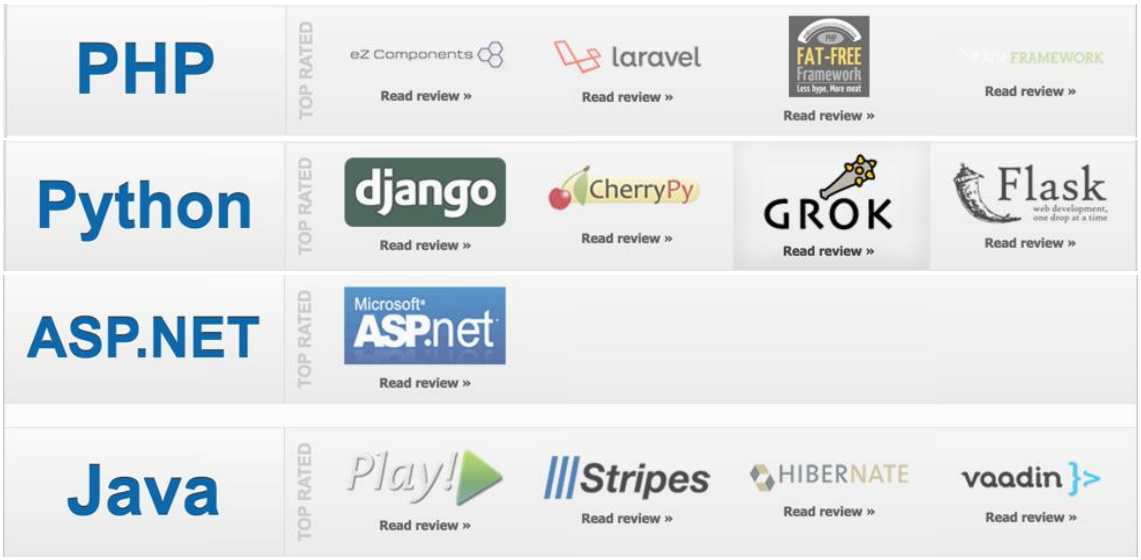


Ilustración 5: Popularidad basada en feedback de la comunidad de desarrolladores[5].

2.3 Comunicación entre ambos mundos

Cuando se trata de ofrecer un mismo servicio tanto a usuarios del mundo móvil como a usuarios del mundo web, es fundamental que el intercambio de información esté basado en estándares bien definidos que faciliten la construcción de la lógica de negocios de la aplicación. Algunas de las formas de interconectar sistemas pueden ser el uso de sockets, protocolos SOAP, servicios REST e intercambio de ficheros XML.

Al tratar de interconectar una aplicación móvil con una lógica de negocio en servidor, por norma general se estará hablando de construir un API que encapsule dicha lógica de negocio y la exponga a modo de servicios para que otros la puedan consumir. Los métodos más usados para la creación de un API de estas características están resumidos en la siguiente figura.

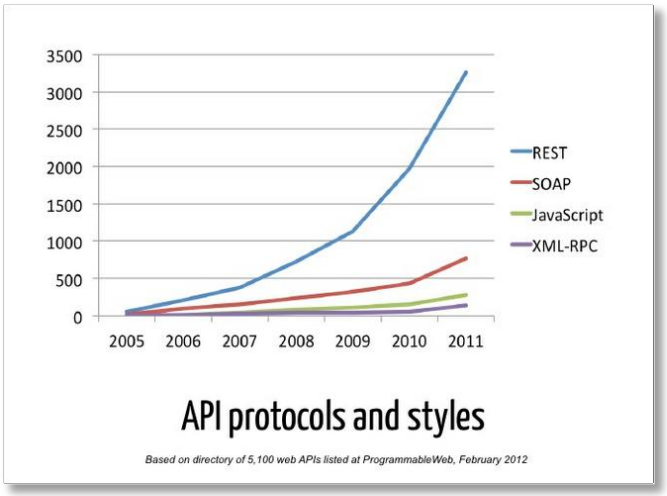


Ilustración 6: Popularidad de APIs

La figura anterior muestra la enorme diferencia que existe entre los dos protocolos más extendidos y utilizados en el desarrollo de APIs. REST es el protocolo más que asentado y constituye la base fundamental de comunicación entre la mayoría de aplicaciones móviles de mercado y los servicios de backend, donde se sitúa la lógica de negocio de una aplicación. Este protocolo basado en comunicaciones HTTP nos ofrece la posibilidad de autenticar cada petición empleando tres mecanismos de verificación de la identidad.

1. **Autenticación básica**. Este tipo de sistema emplea autenticación con sesión HTTP en el que las peticiones al API deben ir autenticadas mientras dure la sesión. Es muy empleado en todo tipo de aplicaciones web, ya que se basa en la combinación de usuario y contraseña.

2. **Autenticación basada en tokens**. Este tipo de sistema genera una credencial única por usuario que no requiere de una sesión HTTP para funcionar. En su defecto se establece una fecha de caducidad tras la cual el token perderá validez y las llamadas no se ejecutarán.

3. **Autenticación OAUTH2**. Este es el tipo de autenticación más compleja y permite compartir un recurso en un servidor de confianza sin necesidad de manejar la credencial del usuario. Se emplea en muchos casos para integrarse con servicios de terceros.

La elección de un mecanismo de autenticación depende en gran medida del tipo de recurso que queremos proteger y de la naturaleza del servidor donde se hospeda la lógica de negocio de la aplicación.

Capítulo 3

Toma de requisitos

A continuación se detallan los requisitos del sistema clasificados en cuatro tipos: funcionales, de datos, de seguridad y de interfaz. Esta lista de requisitos es el resultado de dos reuniones con el cliente.

En la primera el cliente esbozó la línea de negocio sobre la que había que trabajar, de manera que, sin tener muy claro lo que quería como producto final, se le ofreciera una solución a medida.

En la segunda reunión, el cliente perfiló la solución ofrecida, consiguiéndose así una lista detallada de los requerimientos que debe cumplir el sistema, la cual se muestra a continuación:

3.1 Requisitos funcionales

Código	RQF - 01
Nombre	Insertar noticia
Versión	1.0
Autor	Silvia
Tipo	Funcional
Descripción	El sistema deberá poder insertar una noticia nueva. La acción se llevará a cabo desde la web de administración.
Prioridad	ALTA

Tabla 1: RQF – 01

Código	RQF - 02
Nombre	Visualizar noticias
Versión	1.0
Autor	Silvia
Tipo	Funcional
Descripción	El sistema deberá poder mostrar todas las noticias. Tanto en la web de administración como en la aplicación móvil, podrán visualizarse todas de un vistazo en forma de tabla o cada una en vista de detalle.
Prioridad	ALTA

Tabla 2: RQF – 02

Código	RQF - 03
Nombre	Ordenar noticias
Versión	1.0
Autor	Silvia
Tipo	Funcional
Descripción	Se deberán poder ordenar las noticias en el sistema, indicando su orden de prioridad.
Prioridad	MEDIA

Tabla 3: RQF - 03

Código	RQF - 03
Nombre	Ordenar noticias
Versión	1.0
Autor	Silvia
Tipo	Funcional
Descripción	Se deberán poder ordenar las noticias en el sistema, indicando su orden de prioridad.
Prioridad	MEDIA

Tabla 4: RQF - 04

Código	RQF - 05
Nombre	Eliminar noticias
Versión	1.0
Autor	Silvia
Tipo	Funcional
Descripción	El sistema deberá poder eliminar noticias. Esta acción sólo estará disponible en el lado administrador y deberán poder eliminarse por selección.
Prioridad	ALTA

Tabla 5: RQF - 05

Código	RQF - 06
Nombre	Compartir noticia
Versión	1.0
Autor	Silvia
Tipo	Funcional
Descripción	El sistema deberá poder compartir noticias en las siguientes redes sociales redes sociales: • Google + • Facebook • Twitter
Prioridad	ALTA

Tabla 6: RQF - 06

Código	RQF - 07
Nombre	Insertar contacto
Versión	1.0
Autor	Silvia
Tipo	Funcional
Descripción	El sistema deberá poder insertar un contacto. Esta acción sólo estará disponible en el lado administrador.
Prioridad	ALTA

Tabla 7: RQF - 07

Código	RQF - 08
Nombre	Visualizar contactos
Versión	1.0
Autor	Silvia
Tipo	Funcional
Descripción	El sistema deberá poder mostrar todos los contactos. Tanto en el lado administrador como en el lado del cliente móvil deberán poderse visualizar todos los contactos.
Prioridad	ALTA

Tabla 8: RQF - 08

Código	RQF - 09
Nombre	Modificar contactos
Versión	1.0
Autor	Silvia
Tipo	Funcional
Descripción	El sistema deberá poder modificar un contacto. Esta acción se llevará a cabo desde la web de administración y se podrá modificar cualquiera de los campos.
Prioridad	ALTA

Tabla 9: RQF - 09

Código	RQF - 10
Nombre	Eliminar contactos
Versión	1.0
Autor	Silvia
Tipo	Funcional
Descripción	El sistema deberá poder eliminar contactos. Esta acción estará sólo disponible para los administradores del sistema, pudiendo eliminar contactos de manera individual o mediante selección.
Prioridad	ALTA

Tabla 10: RQF - 10

Código	RQF - 11
Nombre	Llamar a un contacto
Versión	1.0
Autor	Silvia
Tipo	Funcional
Descripción	El sistema deberá poder llamar por teléfono a un contacto. Obviamente esta funcionalidad sólo estará disponible en la aplicación móvil y en un solo *click*.
Prioridad	ALTA

Tabla 11: RQF - 11

Código	RQF - 12
Nombre	Enviar un email a un contacto
Versión	1.0
Autor	Silvia
Tipo	Funcional
Descripción	El sistema deberá poder enviar un email a un contacto. De forma similar al anterior, desde la aplicación móvil y en un solo *click*.
Prioridad	ALTA

Tabla 12: RQF - 12

Código	RQF - 13
Nombre	Añadir un directorio nuevo
Versión	1.0
Autor	Silvia
Tipo	Funcional
Descripción	El sistema deberá poder añadir un directorio nuevo y lo hará desde el lado administrador. La estructura será en forma de árbol, de forma que cada carpeta puede contener o no otras carpetas.
Prioridad	ALTA

Tabla 13: RQF - 13

Código	RQF - 14
Nombre	Modificación de directorios
Versión	1.0
Autor	Silvia
Tipo	Funcional
Descripción	Se podrá modificar cualquier campo de los directorios, incluida su ubicación dentro de la estructura de árbol.
Prioridad	ALTA

Tabla 14: RQF - 14

Código	RQF - 15
Nombre	Eliminación de directorios
Versión	1.0
Autor	Silvia
Tipo	Funcional
Descripción	Los directorios podrán eliminarse del sistema de forma permanente y con ellos todo lo que contengan.
Prioridad	ALTA

Tabla 15: RQF - 15

Código	RQF - 16
Nombre	Subida de documentos
Versión	1.0
Autor	Silvia
Tipo	Funcional
Descripción	El sistema dará la posibilidad de añadir y subir documentos en formato PDF desde el lado administrador.
Prioridad	ALTA
Comentarios	Es necesario que cada documento esté incluido en algún directorio.

Tabla 16: RQF - 16

Código	RQF - 17
Nombre	Visualización de documentos
Versión	1.0
Autor	Silvia
Tipo	Funcional
Descripción	El sistema debe dar la posibilidad de visualizar los documentos PDF.
Prioridad	ALTA
Comentarios	Esta funcionalidad debe estar disponible tanto en la parte administrativa web, como en la parte cliente móvil. En concreto, en la aplicación móvil, de debe poder visualizar desde la propia aplicación, sin necesidad de tener instalada una aplicación de terceros.

Tabla 17: RQF - 17

Código	RQF - 18
Nombre	Modificación de documentos
Versión	1.0
Autor	Silvia
Tipo	Funcional
Descripción	Desde el lado administrador del sistema se podrá modificar cada documento, incluyendo el archivo PDF.
Prioridad	ALTA

Tabla 18: RQF - 18

Código	RQF - 19
Nombre	Eliminación de documentos
Versión	1.0
Autor	Silvia
Tipo	Funcional
Descripción	De la misma forma que se hacen todas las eliminaciones, el administrador podrá eliminar los documentos que haya seleccionado.
Prioridad	ALTA

Tabla 19: RQF - 19

Código	RQF - 20
Nombre	Importar documentos CSV
Versión	1.0
Autor	Silvia
Tipo	Funcional
Descripción	El sistema ofrecerá la posibilidad de generar, a partir de ficheros con extensión .csv, los modelos de datos necesarios para almacenar la información contenida.
Prioridad	ALTA
Comentarios	Esta funcionalidad estará disponible sólo desde el lado administrador de la aplicación y para el módulo “Mi Cuenta”.

Tabla 20: RQF - 20

Código	RQF - 21
Nombre	Comunicación entre web y móvil
Versión	1.0
Autor	Silvia
Tipo	Funcional
Descripción	Todos los datos que se muestren en la aplicación móvil tendrán como fuente de información la Web de administración. El formato de traspaso de información entre ambos módulos será JSON. La aplicación móvil realizará llamadas a la API de la administración cada vez que necesite información.
Prioridad	ALTA

Tabla 21: RQF - 21

3.2 Requisitos de datos

Código	RQD – 01
Nombre	Repositorio de noticias
Versión	1.0
Autor	Silvia
Tipo	Datos
Descripción	La información que se almacenará de cada una de las noticias se organiza de la siguiente manera: - Título - Descripción breve - Cuerpo de la noticia - Imagen (opcional) - URL de imagen externa - ID vídeo Youtube - Link externo a noticia completa - Orden
Prioridad	Alta

Tabla 22: RQD - 01

Código	RQD – 02
Nombre	Repositorio de contactos
Versión	1.0
Autor	Silvia
Tipo	Datos
Descripción	Cada uno de los contactos contendrá los siguientes campos: - Nombre - Apellidos (opcional) - Departamento al que pertenezca - Email - Teléfono
Prioridad	Alta

Tabla 23: RQD - 02

Código	RQD – 03
Nombre	Repositorio de documentos
Versión	1.0
Autor	Silvia
Tipo	Datos
Descripción	Se deberá ofrecer un catálogo de documentos en formato PDF.
Prioridad	Alta
Comentarios	Las dimensiones máximas permitidas para un documento son 5MB.

Tabla 24: RQD - 03

Código	RQD – 04
Nombre	Árbol de directorios
Versión	1.0
Autor	Silvia
Tipo	Datos
Descripción	De forma similar a un sistema de archivos, el sistema organizará los documentos en directorios. Cada directorio podrá contener otros directorios y/o documentos.
Prioridad	Alta

Tabla 25: RQD - 04

Código	RQD – 05
Nombre	Árbol de directorios
Versión	1.0
Autor	Silvia
Tipo	Datos
Descripción	De forma similar a un sistema de archivos, el sistema organizará los documentos en directorios. Cada directorio podrá contener más directorios o documentos.
Prioridad	Alta

Tabla 26: RQD - 05

Código	RQD – 06
Nombre	Cuentas de usuario
Versión	1.0
Autor	Silvia
Tipo	Datos
Descripción	El sistema almacenará en la base de datos la cuenta de usuario asociada a cada cliente con lo siguiente: - Nombre - Empresa - Teléfono de contacto - Cliente al que pertenece la cuenta - Email - Código de reseteo de contraseña - Caducidad de la cuenta - Si es administrador
Prioridad	Alta
Comentario	Sin una cuenta de usuario no se podrá acceder a la web de administración y la en la aplicación móvil sólo se podrá navegar en modo invitado.

Tabla 27: RQD - 06

Código	RQD – 07
Nombre	Salas de bastidores
Versión	1.0
Autor	Silvia
Tipo	Datos
Descripción	El sistema almacenará los datos de las salas que cada cliente tenga reservadas. De cada una de ellas se guardará: - Nombre de la sala - Planta del edificio en la que se aloja físicamente - Mapa de la sala - Cliente al que pertenece
Prioridad	Alta

Tabla 28: RQD - 07

Código	RQD – 08
Nombre	Racks/Huellas
Versión	1.0
Autor	Silvia
Tipo	Datos
Descripción	Se mantendrá información sobre los bastidores, también llamados huellas, contenidos en cada sala: - Nombre identificativo - Sala en la que esté ubicado - Cliente al que pertenezca
Prioridad	Alta

Tabla 29: RQD - 08

Código	RQD – 09
Nombre	Tomas eléctricas
Versión	1.0
Autor	Silvia
Tipo	Datos
Descripción	El sistema preservará la información de los distintos tipos de tomas eléctricas que puede contener un mismo rack. Contendrá los siguientes campos: - Tipo de corriente (continua o alterna) - Tipo de servicio contratado - Código descriptivo de la toma - Amperaje - Unidad de medida de la corriente
Prioridad	Alta

Tabla 30: RQD - 09

Código	RQD – 10
Nombre	Circuitos
Versión	1.0
Autor	Silvia
Tipo	Datos
Descripción	Dentro del mismo CPD el cliente puede tener contratados circuitos de conexión. Para ellos se reserva una tabla de la base de datos con los campos: - Código identificativo - Tipo (coaxial, UTP, fibra óptica) - Subtipo - Si está parcheado o no - Cliente al que pertenece - Circuito al que esté conectado
Prioridad	Alta

Tabla 31: RQD - 10

3.3 Requisitos de seguridad

Código	RQS - 01
Nombre	Control de acceso
Versión	1.0
Autor	Silvia
Tipo	Seguridad
Descripción	El sistema cuenta con 3 tipos de acceso: - **Administrador**: Teniendo acceso a la Web con todas las funcionalidades disponibles y a todos los módulos de la aplicación móvil. - **Cliente**: Teniendo acceso sólo a la aplicación móvil, pero con todos los permisos. - **Invitado**: Pudiendo acceder a los módulos de la aplicación móvil que no requieran contratar ningún servicio con Norextin.
Prioridad	ALTA

Tabla 32: RQS - 01

3.4 Requisitos de interfaz

Código	RQI - 01
Nombre	Interfaz de administración
Versión	1.0
Autor	Silvia
Tipo	Interfaz

Descripción	El módulo web de administración deberá ser muy claro y sencillo. Con los módulos de la aplicación perfectamente diferenciados: - Noticias - Documentación - Mi cuenta - Contactos - Importaciones Dentro de cada módulo se deberán poder realizar las acciones de "Añadir" y "Modificar" en un solo click. En el caso de que existan datos, deberá aparecer un listado de los mismos con opción de "Eliminar" los seleccionados.
Prioridad	ALTA

Tabla 33: RQI - 01

Código	RQI - 02
Nombre	Interfaz móvil
Versión	1.0
Autor	Silvia
Tipo	Interfaz
Descripción	En la aplicación móvil habrá cuatro secciones, independientes entre sí: - Noticias - Documentación - Mi cuenta - Contactos La navegación entre pantallas dentro de cada sección será independiente, actuando cada sección como una cola de pantallas donde se guarda el estado.
Prioridad	ALTA

Tabla 34: RQI - 02

Capítulo 4

Análisis del Sistema de Información

Durante esta fase de la ejecución del proyecto se deben identificar y esclarecer todos los objetivos previos a la fase de diseño del sistema de información. Los cuatro puntos que se van a abordar en las siguientes secciones son los descritos a continuación:

4.1 Identificación de todos los subsistemas que componen el global de la aplicación a desarrollar.
4.2 Justificación del *stack* de tecnologías a emplear durante la fase de desarrollo.
4.3 Identificación de los riesgos.
4.4 Identificación de todos los posibles casos de uso para los distintos roles de usuario

4.1 Identificación de los subsistemas

Con la lista de requisitos de usuario en mente se pueden identificar como necesidades fundamentales el acceso a la información desde una aplicación web y el acceso a la información desde un dispositivo móvil. Para poder crear un sistema que permita interconectar estos dos componentes se deben identificar los subsistemas que permitan llevarlo a cabo.

1. **Portal web** para la administración y consulta de la aplicación de Norextin.
2. **Aplicación móvil** para la consulta de servicios contratados con Norextin.
3. **Subsistema de comunicación** de datos entre web y móvil.

El portal web constituye el principal acceso de gestión de la aplicación. Desde él, tanto clientes como administradores podrán gestionar los servicios contratados con Norextin.

La aplicación móvil será el punto de entrada para los clientes que desean consultar el estado de sus servicios contratados con Norextin.

Por último el subsistema de comunicación debe permitir gestionar y consultar los datos tanto desde la web como desde el terminal móvil, por lo que supone un candidato ideal para la implementación de un servicio REST sobre los datos de la aplicación.

4.2 Justificación del stack de tecnologías a emplear

Durante la presentación del estado del arte se habló sobre los posibles *frameworks* que existen en la actualidad para la construcción de una aplicación web, así como los principales mercados de aplicaciones para dispositivos móviles. Es fundamental definir ahora con claridad la pila de tecnologías que se va a emplear durante la construcción del sistema, que va a dividirse en dos grandes bloques.

1. **Pila de tecnologías en servidor.** Donde se incluye la lógica de negocio, la aplicación web y los mecanismos de comunicación.
2. **Pila de tecnologías móviles**. Donde se escogerá la plataforma más adecuada para la implementación del cliente móvil y la filosofía de comunicación con servidor.

4.2.1 Pila de tecnologías en servidor

Tras identificar los subsistemas que componen la aplicación a desarrollar, se ha visto que debe implementarse tanto un portal web como un servidor que permita la consulta de los datos desde una aplicación móvil. De todos los *frameworks* vistos durante el estudio del estado del arte, quizás Python y Java, sean los dos lenguajes que más posibilidades ofrecen a la hora de construir un sistema de estas características.

En el mundo Java se pueden encontrar *frameworks* conocidos como *full stack*, o *frameworks* que proporcionan todo lo necesario para construir una aplicación web así como su integración con bases de datos, gestión de usuarios, sistemas de seguridad, servicios REST, etc. Tanto Play como Spring son dos buenos ejemplos de *frameworks full stack*, pero su configuración y manejo no son aptos para todos los públicos. Spring requiere de un proceso de formación tanto en MVC como en Spring Data y Spring Web, y si bien en grandes proyectos son recomendables por ofrecer innumerables ventajas, en proyectos de corto alcance y rápido desarrollo puede que no lo sean tanto por falta de experiencia en el equipo de desarrollo.

En el lado de Python se encuentra Django, otro buen ejemplo de *framework full stack* que permite desarrollar una aplicación de estas características. La ventaja de Django frente a Spring es principalmente su curva de aprendizaje. Una aplicación Django es auto contenida, no necesita un sistema de resolución de dependencias como Maven, y permite de fábrica

manejar el acceso a la base de datos, servir los ficheros HTML o gestionar las peticiones de un servicio REST. Los módulos se importan fácilmente en un fichero de configuración y se emplea uno de los lenguajes de alto nivel más fáciles del mercado: Python.

Se propone el entorno de Python y Django como pila de tecnologías en servidor entre otros, por los siguientes motivos:

1. Incorpora un ORM que permite crear y mapear las tablas de la base de datos con sólo definir el modelo de datos.
2. Incorpora los mecanismos de seguridad necesarios para la autenticación de usuarios.
3. Permite construir una web fácilmente empleando estándares HTML5 y CSS3 con plantillas predefinidas para un desarrollo ultra rápido.
4. Permite construir un servicio REST con solo importar un módulo de Django.

4.2.2 Pila de tecnologías móviles

En el estado del arte se han identificado las cuatro plataformas más importantes en cuanto a volumen de usuarios y aplicaciones en el mercado. De esas cuatro Android e iOS son las que dominan el mercado actual.

Android es un sistema operativo Open Source con millones de usuarios activos y una extensa comunidad de desarrolladores. Las aplicaciones en Android se codifican en Java, uno de los lenguajes más usados y extendidos en aplicaciones de todo tipo. Android supera a iOS en volumen de usuarios y descarga de aplicaciones, aunque no en número de aplicaciones publicadas.

Por otra parte, iOS es el sistema operativo propietario desarrollado por Apple Inc. Está presente en sus teléfonos y tabletas y cuenta con un número mayor de aplicaciones publicadas en su tienda. Las aplicaciones en iOS se desarrollan empleando el lenguaje de programación Objetive-C, un lenguaje específico para esta plataforma con una sintaxis parecida a C, pero con algunas peculiaridades. El desarrollo en esta plataforma requiere obligatoriamente de un ordenador Mac con sistema operativo Macintosh.

Se propone Android como entorno inicial para el desarrollo de la aplicación móvil por los siguientes motivos:

1. Facilidad del lenguaje de programación frente al de la plataforma de Apple.
2. El volumen de usuarios y descargas en Android supera al de iOS. Por cuota de mercado inicial es más recomendable Android.
3. El entorno de desarrollo es gratuito y multiplataforma. No es necesario invertir en equipos Mac nuevos para el desarrollo de la aplicación.
4. Entre las personas que van a formar parte del equipo de desarrollo al menos una de ellas tiene amplia experiencia en el lenguaje de programación Java.

4.3 Identificación de los riesgos

Se entiende como riesgo cualquier variable del proyecto que pone en peligro o impide el éxito del mismo. Es la "probabilidad de que un proyecto experimente sucesos no deseables, como retrasos en las fechas, excesos de costes, o la cancelación directa".

El propósito del plan es identificar los elementos que pueden comprometer el desarrollo del proyecto, analizarlos, calcular la exposición y en base a ello poder priorizarlos, para establecer estrategias de control y resolución, que permitan ejercer una correcta supervisión de los mismos.

A continuación se presenta una identificación de los mismos:

Riesgo	Descripción	Probabilidad	Efecto
Complejidad del software	Se ha supuesto una dificultad media en ambos módulos: web y móvil, lo que podría provocar un desajuste en la planificación ralentizando tiempos y retrasando entregas.	Alta	Alto
Estimación del presupuesto	Con objeto de facilitar la venta del desarrollo, se han ajustado mucho los precios. Es posible que la estimación sea demasiado baja.	Media	Alto
Personal clave	Los recursos asignados son esenciales, sus posibles ausencias provocarían efectos devastadores.	Media	Alto
Comunicación con el cliente	No es un cliente nuevo y la experiencia en el trato con él delata el poco o nulo *feedback* que ofrece.	Alta	Medio
Cambios en los requisitos	Sobre todo cuando el cliente no es consciente del impacto de algunos requisitos sobre el sistema, es un riesgo alto que se produzcan cambios en los mismos.	Alta	Alto
Inexperiencia de los desarrolladores en las tecnologías elegidas	La inexperiencia es un hecho, el riesgo es su curva de aprendizaje. Se ha considerado la facilidad de los mismos para adaptarse a nuevas tecnologías aunque podría ser una sobreestimación.	Alta	Medio

Módulo 'Mi Cuenta'	De manera muy concreta, especificar que se inicia este desarrollo teniendo total desconocimiento de este módulo. Desconocimiento en el modelo de negocio y en su papel dentro del sistema.	Muy alta	Muy alto

4.4 Diagramas de casos de uso

El siguiente paso es definir los diferentes casos de uso que representarán una vista externa del sistema. Mediante estos diagramas podremos ver con más detalle las funcionalidades más generales de la aplicación, así como quiénes pueden acceder a las mismas.

Los diagramas se van a presentar en función de los roles que un usuario juega con respecto al sistema, destacando que el uso de la palabra rol no necesariamente representa a una persona en particular, sino más bien la labor que realiza frente al sistema.

4.4.1 Diagrama general del sistema (Rol administrador)

La siguiente figura muestra los diferentes módulos y funcionalidades que tendrá a su disposición el rol administrador del sistema.

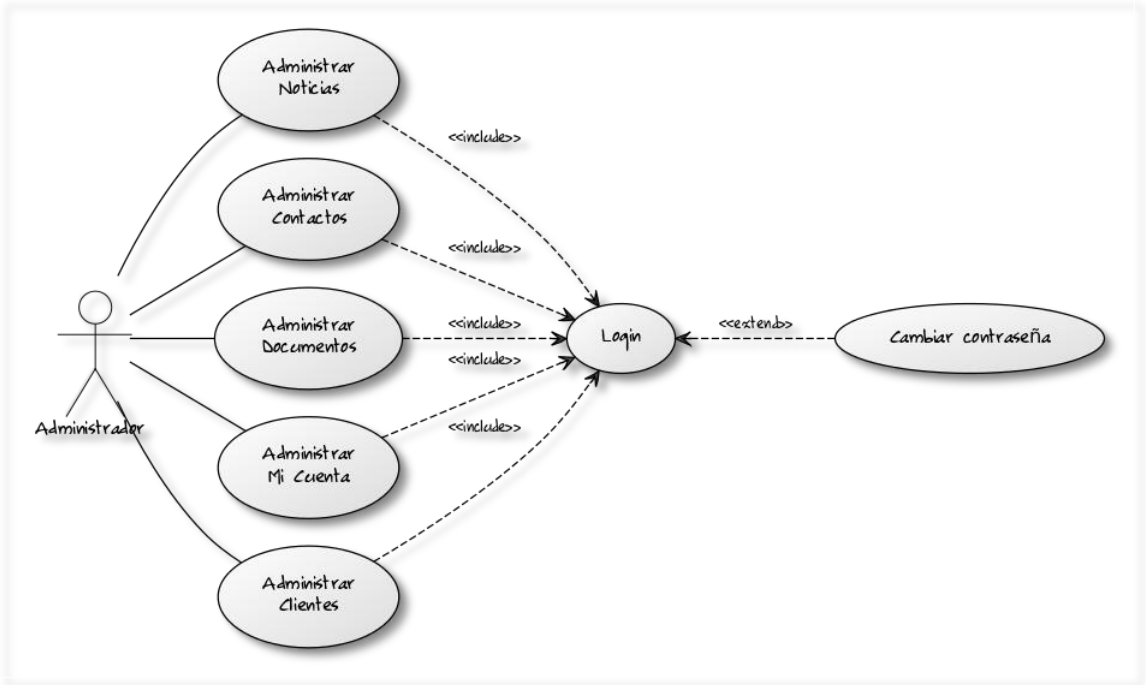


Figura 1: Diagrama general del sistema (Rol Administrador)

Como tal, un administrador se va a encargar de todas las labores de gestión y va a tener permisos para acceder a todos y cada uno de los módulos contenidos en la aplicación. Este proyecto va a constar de cinco módulos:

- Noticias
- Contactos
- Documentos
- Mi Cuenta
- Clientes

El usuario de la aplicación web de administración deberá estar logado correctamente y comprobados sus permisos para simplemente acceder al sistema web. Una vez dentro del sistema tendrá la posibilidad de modificar la contraseña que se le haya asignado por defecto y llevar a cabo las actividades indicadas en el diagrama anterior.

Con objeto de que se entienda y quede detallado, se van a exponer a continuación todos los módulos de manera individual.

4.4.1.1 Caso de uso 'Administrar Noticias'

El módulo 'Noticias', en la parte web del sistema, proporciona acceso a todas las acciones posibles sobre la base de datos, lo que se conoce coloquialmente como operaciones CRUD (Create, Read, Upload and Delete).

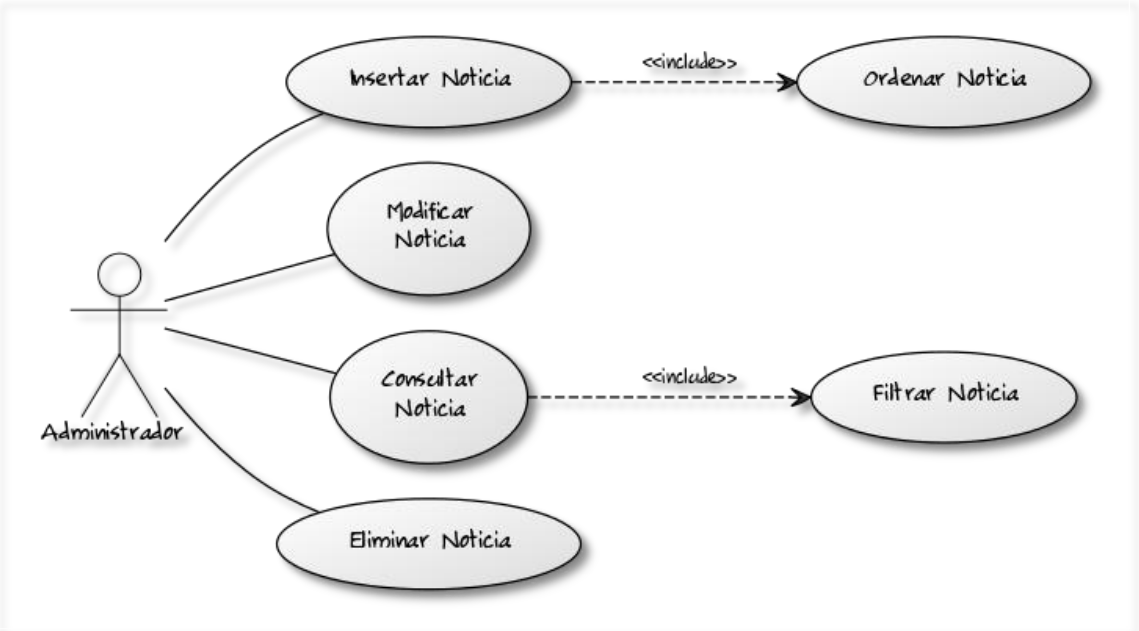


Figura 2: Caso de uso 'Administrar Noticias'

Acciones:

- **Insertar Noticia:** Cuando se añade una noticia nueva en el sistema hay que indicar en qué orden se desea que aparezca en la aplicación móvil. De esta manera, los usuarios administradores podrán dar más o menos importancia a según qué noticias.

- **Modificar Noticia:** Una vez añadida a la base de datos, el usuario podrá modificar en cualquier momento cualquiera de los campos pertenecientes al objeto, desde el propio título hasta la imagen. Esta acción es muy útil sobre todo para actualizar el orden de aparición de las mismas.

- **Consultar Noticias:** Dentro del módulo, se mostrará en forma de tabla el conjunto de noticias que existan en la base de datos, pudiendo llegar a filtrarlas por cualquiera de los campos y acotar la búsqueda.

- **Eliminar Noticia:** Desde la misma tabla de consulta o cuando se está en el área de consulta individual, pueden descartarse noticias. Una forma muy útil de hacerlo cuando hay varias que se desean eliminar es la eliminación por selección múltiple, que a golpe de un solo clic borra de la base de datos todas las que se deseen.

4.4.1.2 Caso de uso 'Administrar Contactos'

El módulo dedicado a administrar los contactos de la aplicación, de forma similar al módulo 'Noticias', facilita todas las operaciones CRUD sobre la base de datos.

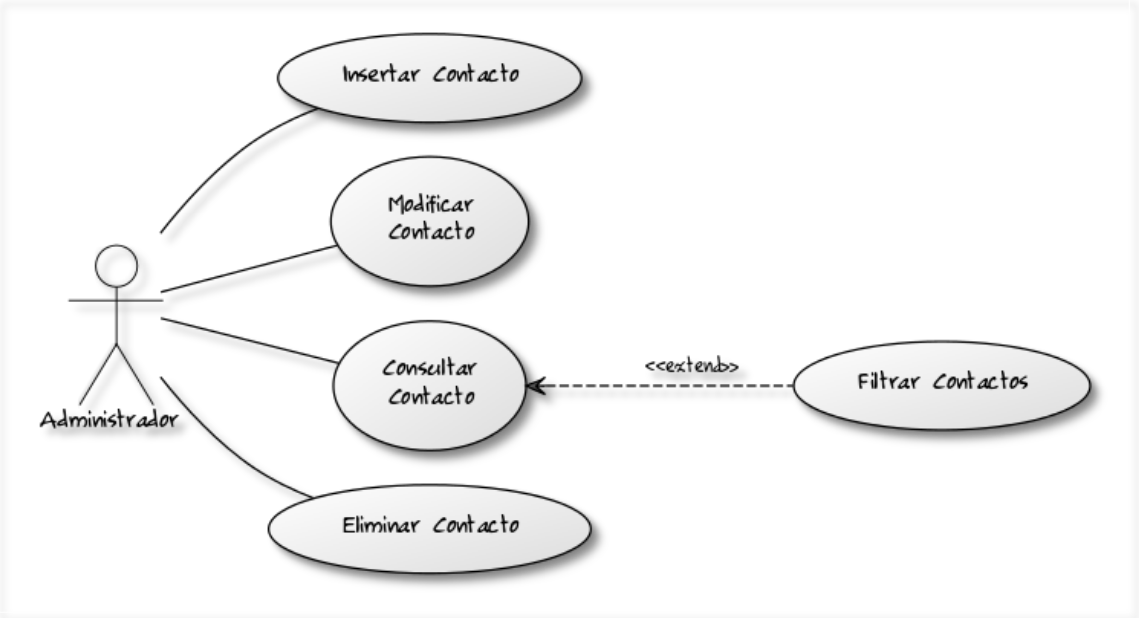


Figura 3: Caso de uso 'Administrar Contactos'

4.4.1.3 Caso de uso 'Administrar Documentos'

La parte del sistema que administra el árbol de documentos de la aplicación funciona de igual manera que los CRUDs anteriores, con la única diferencia de que los directorios pueden estructurarse en forma de árbol y a su vez pueden contener archivos PDF.

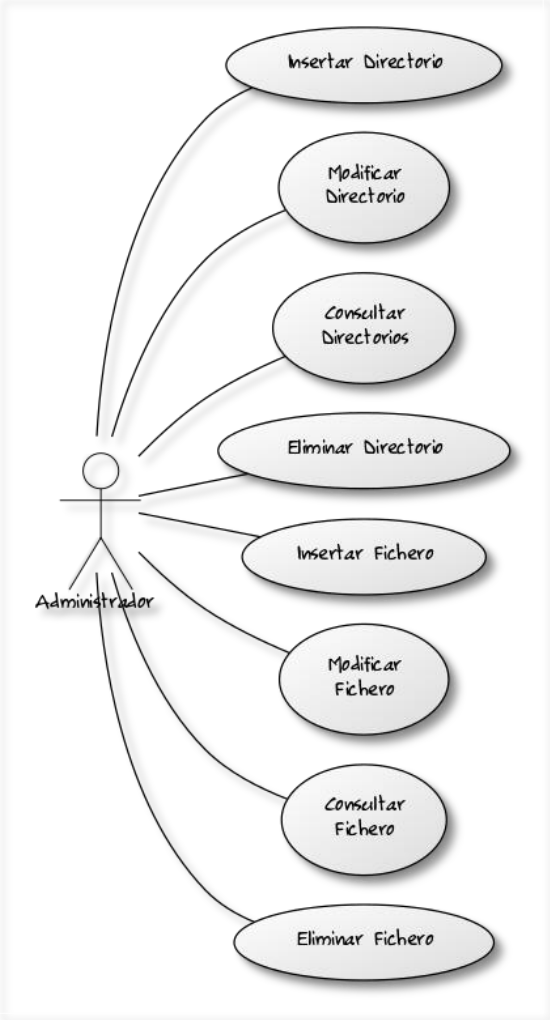


Figura 4: Caso de uso 'Administrar Documentos'

4.4.1.3 Caso de uso 'Administrar Mi Cuenta'

El módulo 'Mi Cuenta' es un poco más extenso que los anteriores porque engloba la administración de los siguientes módulos en su interior:

- Administración de Salas
- Administración de Huellas

- Administración de Circuitos
- Administración de Tomas eléctricas

A continuación se van a detallar los diagramas de cada uno de ellos.

4.4.1.3.1 Caso de uso 'Administrar Salas'

Las salas, al igual que los módulos anteriores, se administran mediante operaciones CRUD, de inserción, consulta, actualización y eliminación.

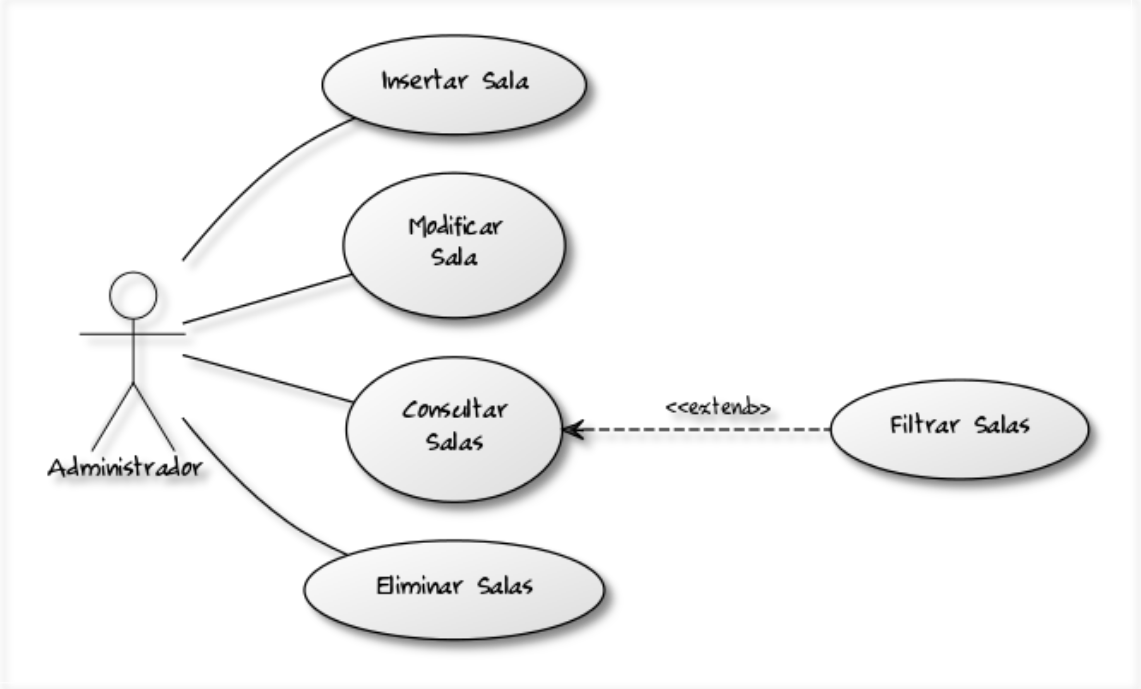


Figura 5: Caso de uso 'Administrar Salas'

4.4.1.3.2 Caso de uso 'Administrar Huellas'

Cada huella, perteneciente a una sala, añade una nueva forma de realizar una de las tareas CRUD. La inserción de huellas, a pesar de conservar la manera manual, también puede realizarse importando un archivo de tipo CSV. A su vez, la importación de huellas crea en la base de datos las salas a las que pertenezca cada huella.

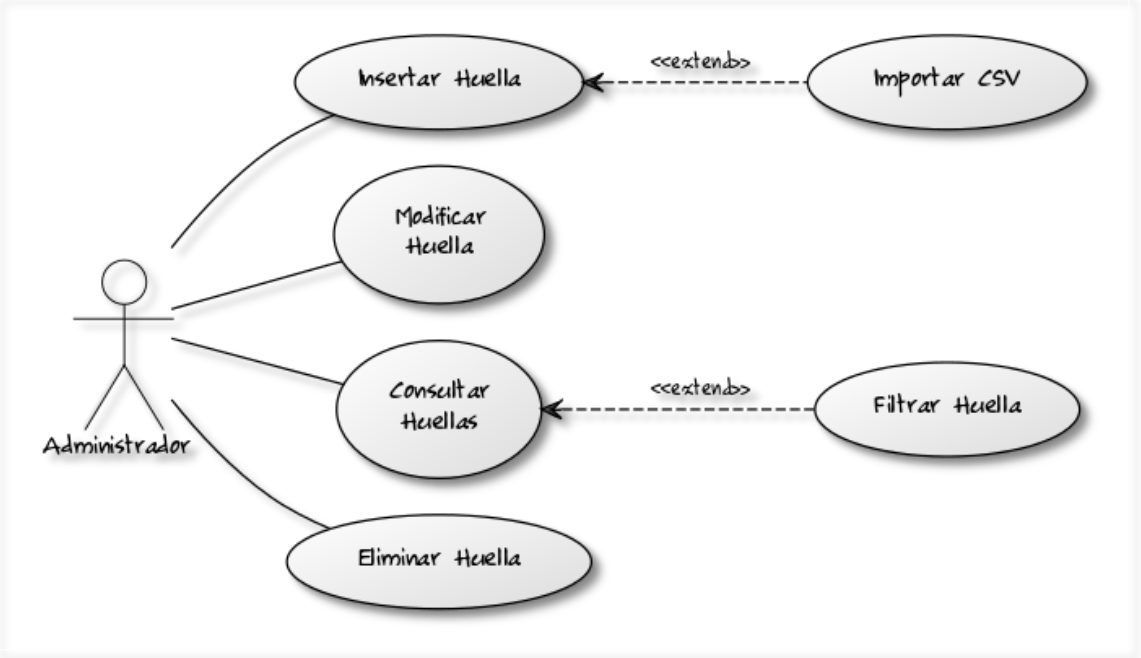


Figura 6: Caso de uso 'Administrar Huellas'

4.4.1.3.3 Caso de uso 'Administrar Circuitos'

Los circuitos tienen un mecanismo similar a las huellas, a lo que son las operaciones CRUD le añaden la posibilidad de insertar de forma múltiple mediante un archivo de extensión CSV que se importa.

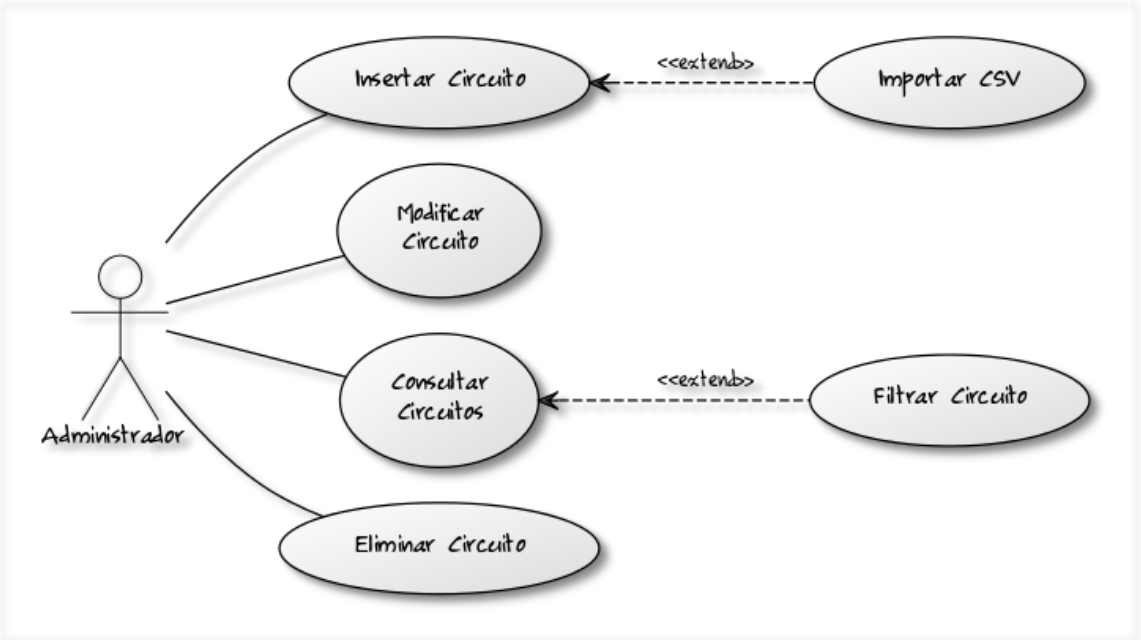


Figura 7: Caso de uso 'Administrar Circuitos'

4.4.1.3.4 Caso de uso 'Administrar Tomas eléctricas'

El submódulo que administra las tomas eléctricas, facilita las cuatro operaciones CRUD necesarias para insertar, consultar, modificar y borrar en la base de datos.

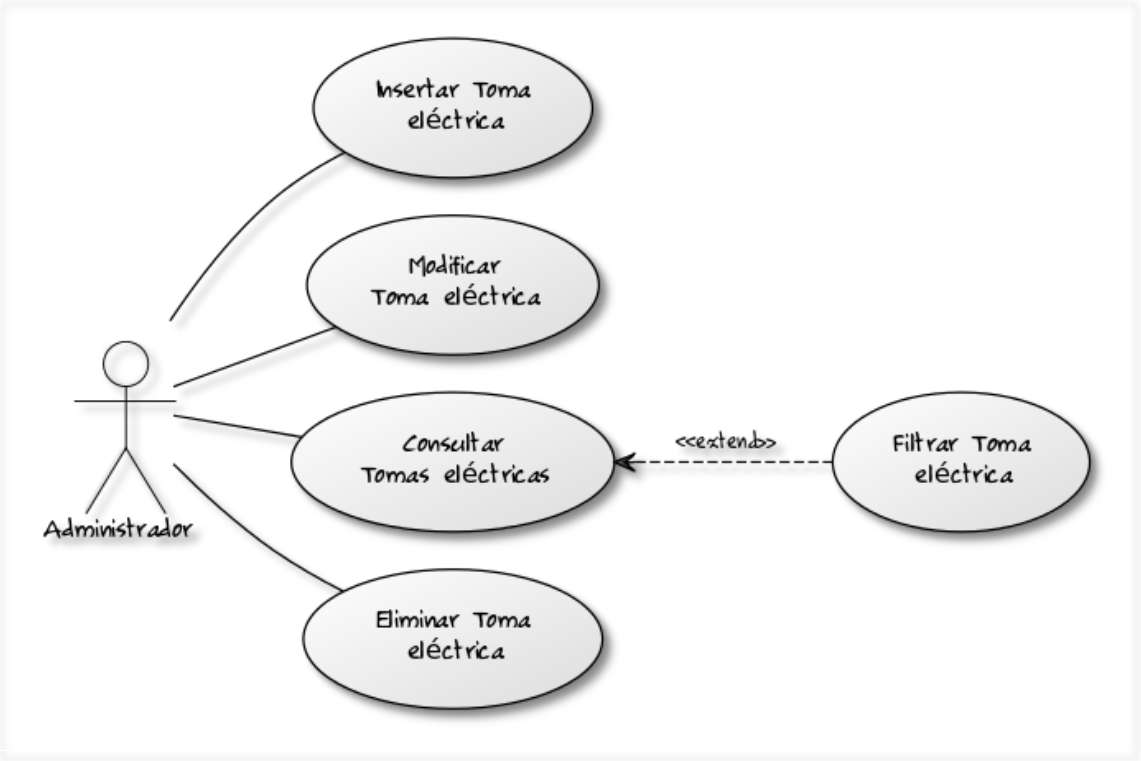


Figura 8: Caso de uso 'Administrar Tomas eléctricas'

4.4.1.4 Caso de uso 'Administrar Clientes'

Los clientes y los usuarios de este sistema son roles que interactúan de forma diferente con la aplicación pero que el administrador los gestiona de igual forma, como todas las entidades de este proyecto.

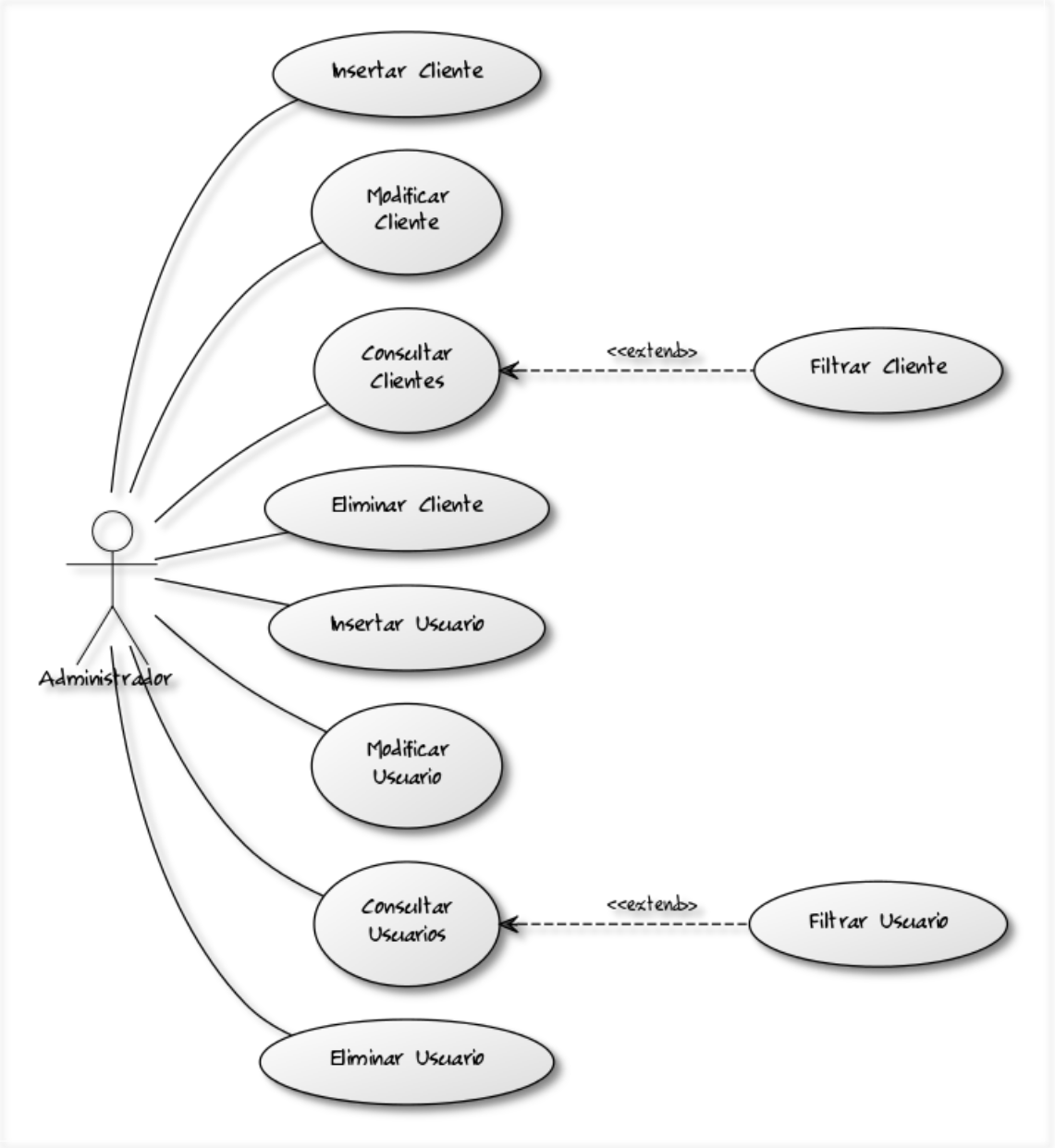


Figura 9: Caso de uso 'Administrar Clientes'

4.4.2 Diagrama general del sistema (Rol Cliente - Invitado)

Se ha separado en un caso de uso aparte las acciones comunes a los roles Cliente e Invitado porque son exactamente las mismas para ambos.

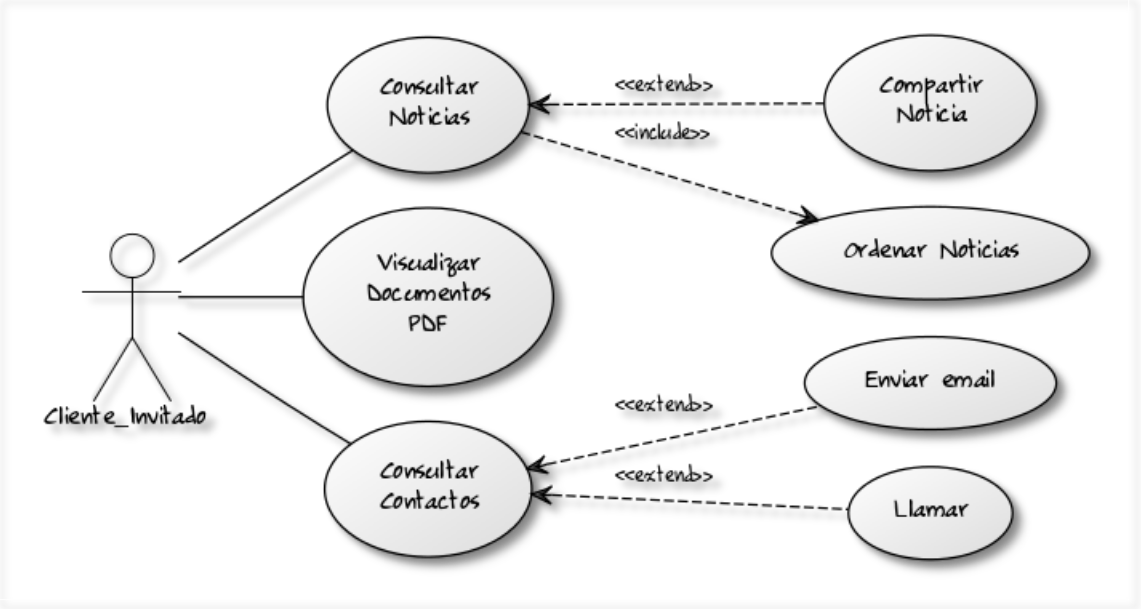


Figura 10: Diagrama general del sistema (Rol Cliente – Invitado)

Estas actividades son las que se llevan a cabo desde la aplicación móvil, la única parte del sistema a la que puede acceder un usuario de tipo cliente o invitado. A diferencia del cliente, el rol invitado no tiene que tener nombre de usuario ni contraseña para acceder.

- **Consultar Noticias:** La aplicación móvil mostrará en forma de Grid las noticias contenidas en la zona de administración y lo hará según el orden que tengan establecido. Una vez seleccionada una noticia y entrado en la vista de detalle de la misma, se podrá compartir por correo electrónico y en las siguientes redes sociales, siempre y cuando estén instaladas en el dispositivo Android:
    - Twitter
    - Facebook
    - Google +
    - Linkedin

- **Visualizar Documentos PDF:** Como ya se ha expuesto en apartados anteriores, los documentos se organizan en forma de árbol de directorios, de forma que el usuario debe ser capaz de navegar entre los mismos sin ningún problema. También podrá ver los documentos PDF sin necesidad de salir de la propia aplicación.

- **Consultar Contactos:** Los contactos contenidos en la base de datos serán visibles para todos los clientes e invitados en forma de lista, desde la cual se podrá poner en contacto con cualquiera de ellos a golpe de un solo clic por correo electrónico o mediante llamada telefónica.

4.4.3 Diagrama general del sistema (Rol Cliente)

Por último queda el caso de uso dedicado al rol Cliente. El sistema está diseñado para que cada usuario de tipo cliente pueda consultar desde su propio dispositivo móvil los servicios que tenga contratados, es decir, los racks, los mapas de sala, los circuitos…

Para ello deberá estar previamente logado y que la aplicación muestre los servicios contratados asociados a ese cliente. Si el usuario está logado (tiene asociado un usuario y contraseña) pero sus servicios no están asociados al cliente correspondiente, no le aparecerá nada.

Cabe destacar también que, aunque no se ha indicado en el diagrama, un usuario con credenciales de administrador (los únicos que pueden acceder al módulo web) puede utilizar la aplicación móvil adoptando el rol de cliente. Esto no pasa de forma viceversa.

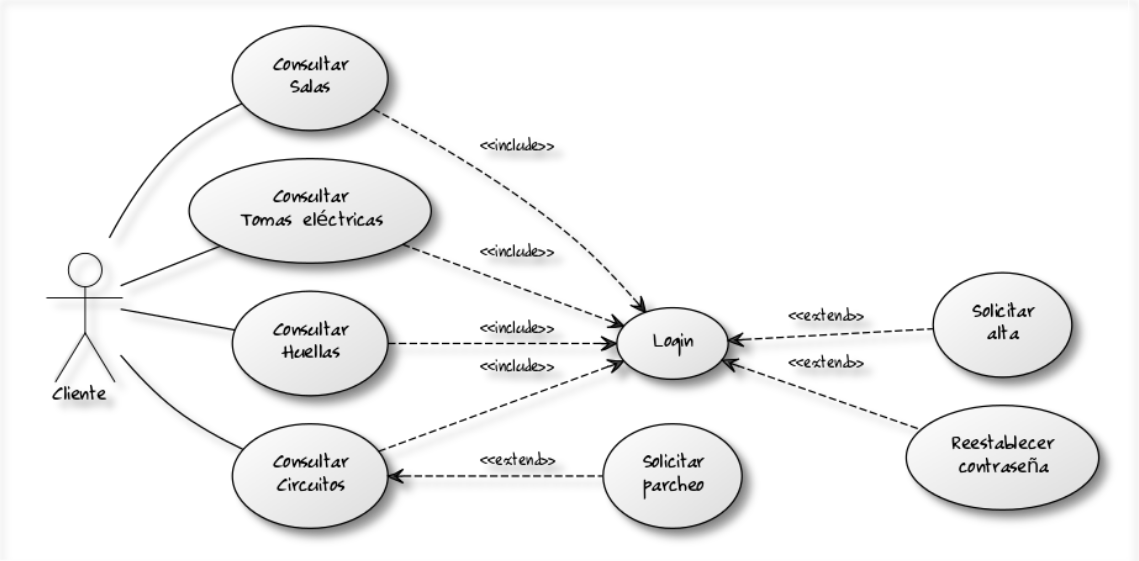


Figura 11: Diagrama general del sistema (Rol Cliente)

Capítulo 5

Diseño del Sistema de Información

Una vez identificados los subsistemas y componentes de alto nivel de la aplicación que se va a desarrollar, la siguiente fase a ejecutar es el diseño del sistema de información. Llegados a este punto se van a completar los siguientes objetivos:

5.1 Arquitectura de componentes.

5.2 Arquitectura de despliegue.

5.3 Diseño del modelo físico de datos.

5.4 Arquitectura detallada del servidor.

5.5 Arquitectura de la aplicación móvil Android.

5.6 Diseño de la interfaz para el portal de administración Web.

5.7 Diseño de la interfaz para la aplicación móvil Android.

5.1 Arquitectura de componentes

El diagrama de arquitectura de componentes que describe la aplicación servidor, y que sirve de interfaz tanto para la web como para el móvil es el que sigue.

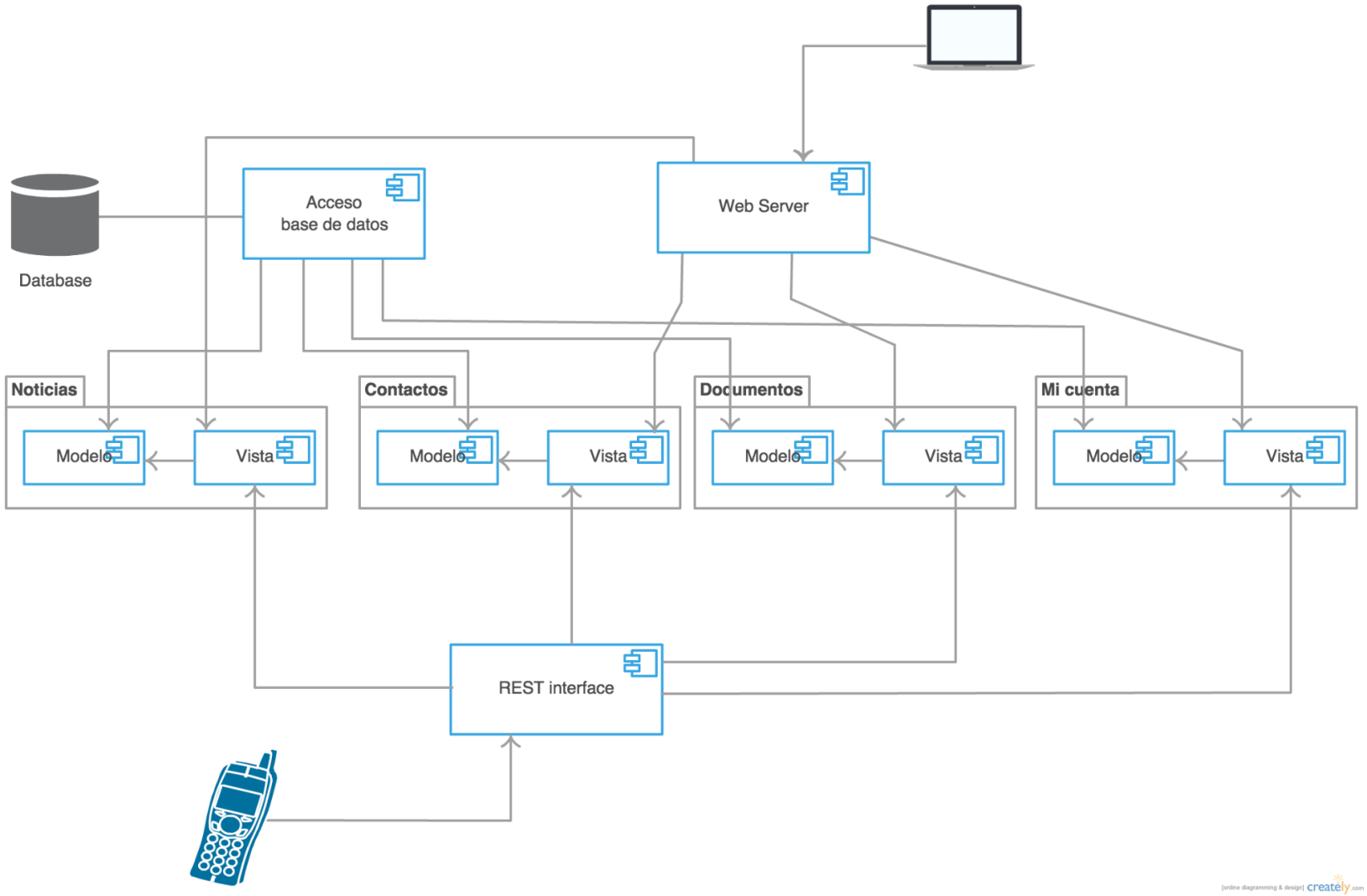


Figura 12: Diagrama de componentes del sistema

En este diseño podemos observar un núcleo central que sirve la lógica de negocio, y que está compuesto por los paquetes Noticias, Contactos, Documentos y Mi cuenta. Cada uno de estos paquetes tiene en su interior dos módulos encargados de gestionar el modelo de datos y la vista.

Conectados a cada uno de los modelos de datos de los distintos paquetes que conforman el aplicativo tenemos el acceso de la base de datos, una interfaz que se encargará de ejecutar todas las operaciones típicas de gestión sobre la base de datos final.

Por último tenemos dos componentes que son la interfaz REST y el servidor web conectados a las vistas de cada uno de los paquetes de la aplicación, esto es posible gracias a que las vistas son las interfaces entre el modelo y el usuario final.

Este diseño basado en modelos y vistas nos permite compartir el mismo núcleo de negocio para los dos tipos de acceso que tenemos que garantizar; web y REST. Los usuarios de la aplicación no accederán directamente al modelo de datos, si no que cualquier acceso debe realizarse pasando por estos dos componentes ya mencionados.

5.2 Arquitectura de despliegue

Una vez definidos los componentes del sistema y sus posibles dependencias se debe especificar la separación física y el despliegue de dichos componentes en el entorno de producción final. El diagrama de despliegue que define esta situación es el que sigue.

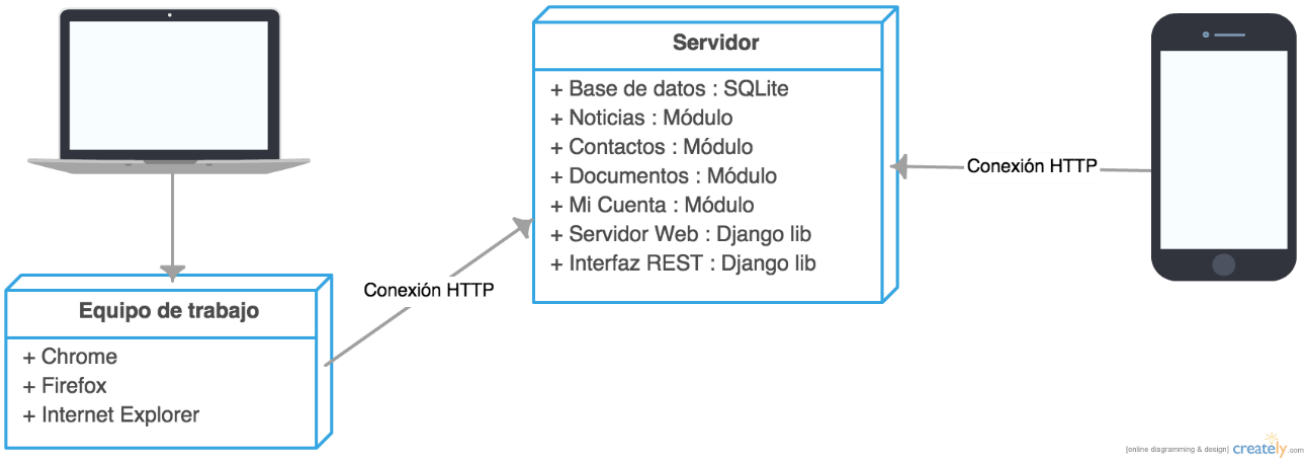


Figura 13: Diagrama de despliegue del sistema

En producción se desplegará la aplicación servidora en una máquina encargada de hospedar tanto la lógica escrita en Django como la base de datos relacional que almacena toda la información de los clientes. A este servidor se le podrán realizar las consultas basadas en el protocolo HTTP tanto desde el navegador web del usuario como desde el terminal móvil con la aplicación de Norextin.

Por el momento no se ha planteado la posibilidad de montar la aplicación en alta disponibilidad o mover la base de datos a un segundo servidor, por lo que el despliegue en un único nodo es más que suficiente.

5.2 Modelo físico de datos

Para el correcto almacenamiento de toda la información que va a manejar el sistema a desarrollar se ha definido el siguiente diagrama de entidad relación.

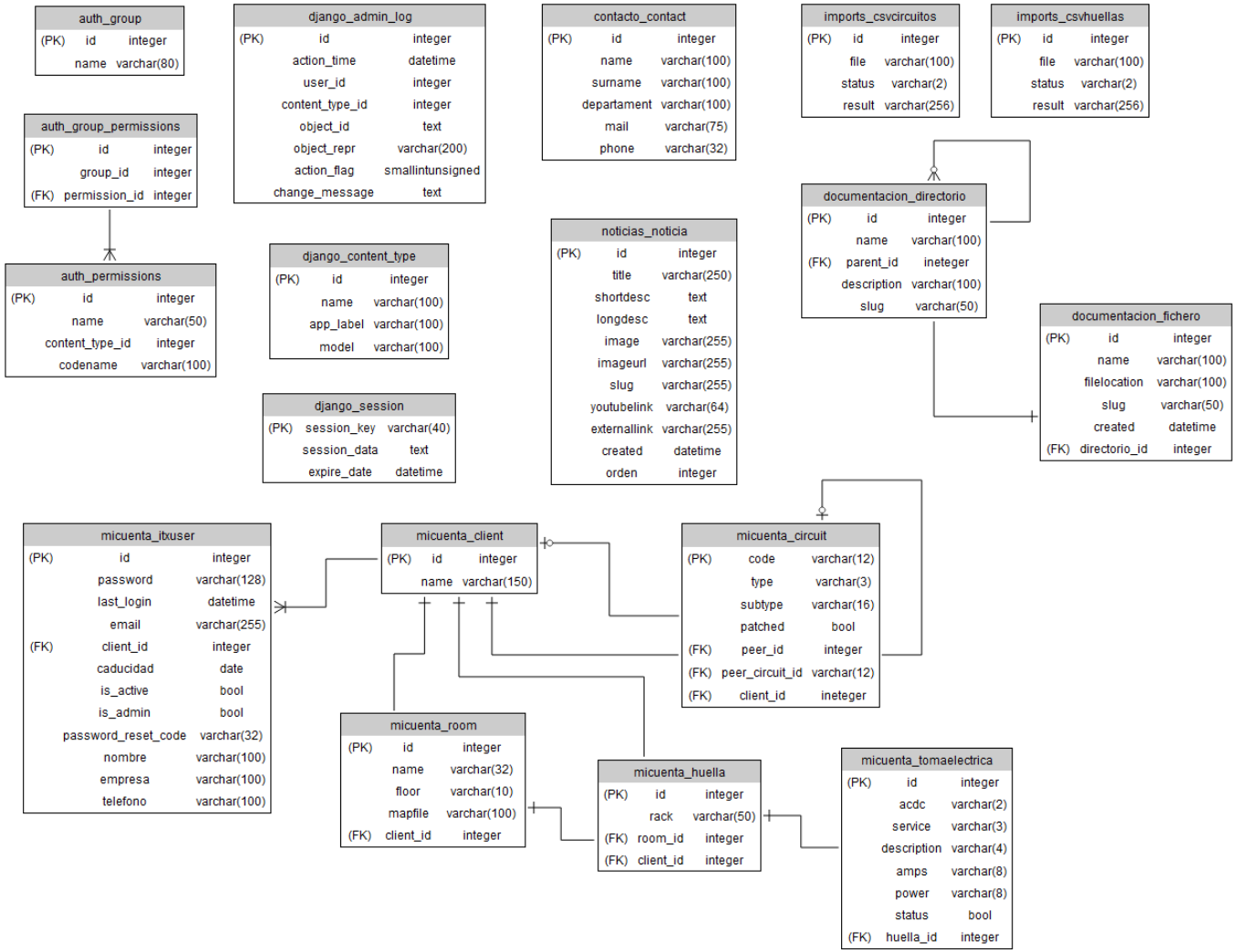


Figura 14: Diagrama Entidad-Relación de la Base de Datos

5.2.1 Tablas específicas de la aplicación

CONTACTO_CONTACT

Tabla que contiene datos de contacto de comerciales y otro tipo de profesionales en Norextin.

IMPORTS_CSVCIRCUITOS

Tabla con el estado de las importaciones de ficheros CSV de circuitos. Sirve para descubrir posibles anomalías en los procesos de importación.

IMPORTS_CSVHUELLAS

Tabla con el estado de las importaciones de ficheros CSV de huellas. Sirve para descubrir posibles anomalías en los procesos de importación.

NOTICIAS_NOTICIA

Tabla con noticias del blog de la página principal de Norextin.

DOCUMENTACION_DIRECTORIO

Representa una carpeta en un sistema de archivos de almacenamiento de documentos. La relación consigo misma permite una jerarquía de árbol en el sistema de ficheros.

DOCUMENTACION_FICHERO

Representa un fichero en un sistema de archivos de almacenamiento de documentos. Los ficheros podrán ser ficheros PDF de tamaño máximo 5MB.

MICUENTA_ITXUSER

Tabla de usuarios de la aplicación, tanto a nivel administrador como cliente final.

MICUENTA_CLIENT

Identificación y nombre de un cliente que hace uso de los servicios de Norextin. La relación con esta tabla será necesaria para poder consultar datos referentes a 'Mi Cuenta' en la aplicación móvil.

MICUENTA_CIRCUIT

Representación de un circuito según la lógica de negocio de Norextin. Un circuito podrá estar conectado con otro circuito del sistema.

MICUENTA_ROOM

Representación de una sala de bastidores según la lógica de negocio de Norextin.

MICUENTA_HUELLA

Representación de un bastidor o comúnmente conocido como Rack. Esta es la unidad central del negocio de Norextin.

MICUENTA_TOMAELECTRICA

Representación de las tomas eléctricas de las que dispone un bastidor en una sala. Esta tabla recoge las características de este tipo de componente.

5.2.2 Tablas específicas del core de Django

AUTH_GROUP

Representación de los distintos grupos a los que puede pertenecer un usuario de la web.

AUTH_GROUP_PERMISSIONS

Asignación de uno o varios permisos a un grupo de usuarios.

AUTH_PERMISSIONS

Unidad de permisos básica para la gestión de permisos de la aplicación.

DJANGO_ADMIN_LOG

Tabla de auditoría de eventos relacionados con la base de datos.

DJANGO_CONTENT_TYPE

Tabla que recoge todos los modelos instalados en la aplicación para su funcionamiento.

DJANGO_SESSION

Contiene la información de la sesión activa de los usuarios en el portal. Entre sus datos encontramos un token de identificación única de sesión y caducidad de la misma.

5.3 Arquitectura detallada del servidor Django

Un proyecto en Django es una estructura simple que se organiza en directorios: el directorio principal y una colección de aplicaciones Django. Éstas pueden ser aplicaciones creadas por el mismo desarrollador o aplicaciones de terceros que se han decidido incluir.

En este proyecto, el directorio principal contiene los ficheros <`manage.py`> y <`db.sqlite3`> y un conjunto de carpetas que en Django reciben el nombre de aplicaciones:

```
norextin/
      manage.py
      db.sqlite3
      norextin/
      templates/
      static/
      recursos/
      media/
      contacto/
      documentacion/
      imports/
      micuenta/
      noticias/
```

- El directorio superior `norextin/` es simplemente la carpeta contenedora de todo el proyecto. Su nombre no afecta a Django.

- `manage.py`: Es una utilidad en línea de comandos que se instala de manera automática en todos los proyectos Django y permite interactuar con el proyecto de diversas formas, entre ellas podemos encontrar:
    - Obtener ayuda en tiempo de ejecución con el comando `'help'`.
    - Determinar qué versiones están instaladas con el comando `'version'`.
    - Mostrar la salida en modo depuración con el comando `'verbosity'`.
    - Mostrar los datos contenidos en la base de datos con el comando `'dumpdata'`.
    - Ejecutar el servidor en la dirección IP y puerto especificado con el comando `'runserver'`.

- El subdirectorio `norextin/` es el paquete real de Python para el proyecto. En él residen los siguientes ficheros:

```
norextin/
      __init__.py
      serializers.py
      settings.py
      urls.py
      wsgi.py
```

    - `__init__.py`: Este es un fichero vacío que dice a Python que este directorio debe ser considerado como un paquete de Python.

- `settings.py`: Fichero de configuración de nuestro proyecto Django. En este fichero se especifican configuraciones fundamentales como lo son las aplicaciones instaladas, el motor de base de datos y demás.
- `urls.py`: En este fichero están las direcciones URL del proyecto Django, es como una 'tabla de contenido' de la web generada por Django.
- `wsgi.py`: Es el punto de partida para los servidores web compatibles con WSGI (Web Server Gateway Interface).

- El subdirectorio `templates/` contiene organizado en subcarpetas las plantillas de cada uno de los módulos de la aplicación. Los templates reciben información de la vista para mostrar el resultado de una forma más organizada, en HTMLs.

```
templates/
      admin/
      micuenta/
      noticias/
      login/
      rest_framework/
```

- `static/`: Es el directorio dedicado a los elementos estáticos de la aplicación tales como estilos CSS, imágenes y logos que aparecen a lo largo de la web y funciones JavaScript.

```
static/
      css/
      img/
      js/
```

- La subcarpeta `recursos/` donde residen algunos recursos que se consideren útiles para la aplicación y que estén ahí por defecto. En este caso, existen ficheros CSV para introducir datos por defecto, que no datos reales.

- Después de esta carpeta está la carpeta de los datos que se han ido introduciendo de manera manual en la aplicación. La carpeta media/:

```
media/
      csv/
      docs/
      images/
```

    - `csv/`: Contiene los archivos de extensión .csv que se han utilizado en algún momento para importar los modelos de datos: Circuitos y Huellas.
    - `docs/`: En esta carpeta residen los ficheros con extensión .pdf insertados en el apartado Documentos.
    - `images/`: Las imágenes en este sistema se le añaden a las Noticias.

- A continuación de esta carpeta lo que hay es una colección de lo que en Django son aplicaciones, una para cada módulo de la aplicación. Todas ellas contienen la misma estructura:

```
norextin/
      __init__.py
      admin.py
      models.py
      views.py
```

    - `__init__.py`: Este es un fichero vacío que dice a Python que este directorio debe ser considerado como un paquete de Python.
    - `admin.py`: Éste es el archivo donde se almacenan los objetos derivados de la clase ModelAdmin que es la representación de un modelo en la interfaz de administración.
    - `models.py`: En este archivo se identifican los modelos de la aplicación. Los modelos son una forma sencilla pero potente de interactuar con la base de datos. Las clases de los modelos son objetos de models.Model.
    - `views.py`: Fichero en el que residen las funciones que contendrán toda la lógica necesaria para dar respuesta a una solicitud.

5.4 Arquitectura detallada de la aplicación Android

A continuación se muestra el diagrama completo de clases, ofreciendo una vista de la estructura estática del sistema, el cual contiene las clases, atributos, métodos y relaciones entre los objetos.

Con el objetivo de optimizar el espacio y conseguir un diagrama lo más claro posible se han omitido:

- Los métodos menos relevantes, por ser semejantes en todas las clases del mismo tipo.
- Los Adapter, cuya función es sobre todo estética. Los objetos Adapter son colecciones de datos que se asignan a una vista para que los muestre, facilitando la personalización gráfica y gestión de eventos.

Véase el diagrama de clases general:

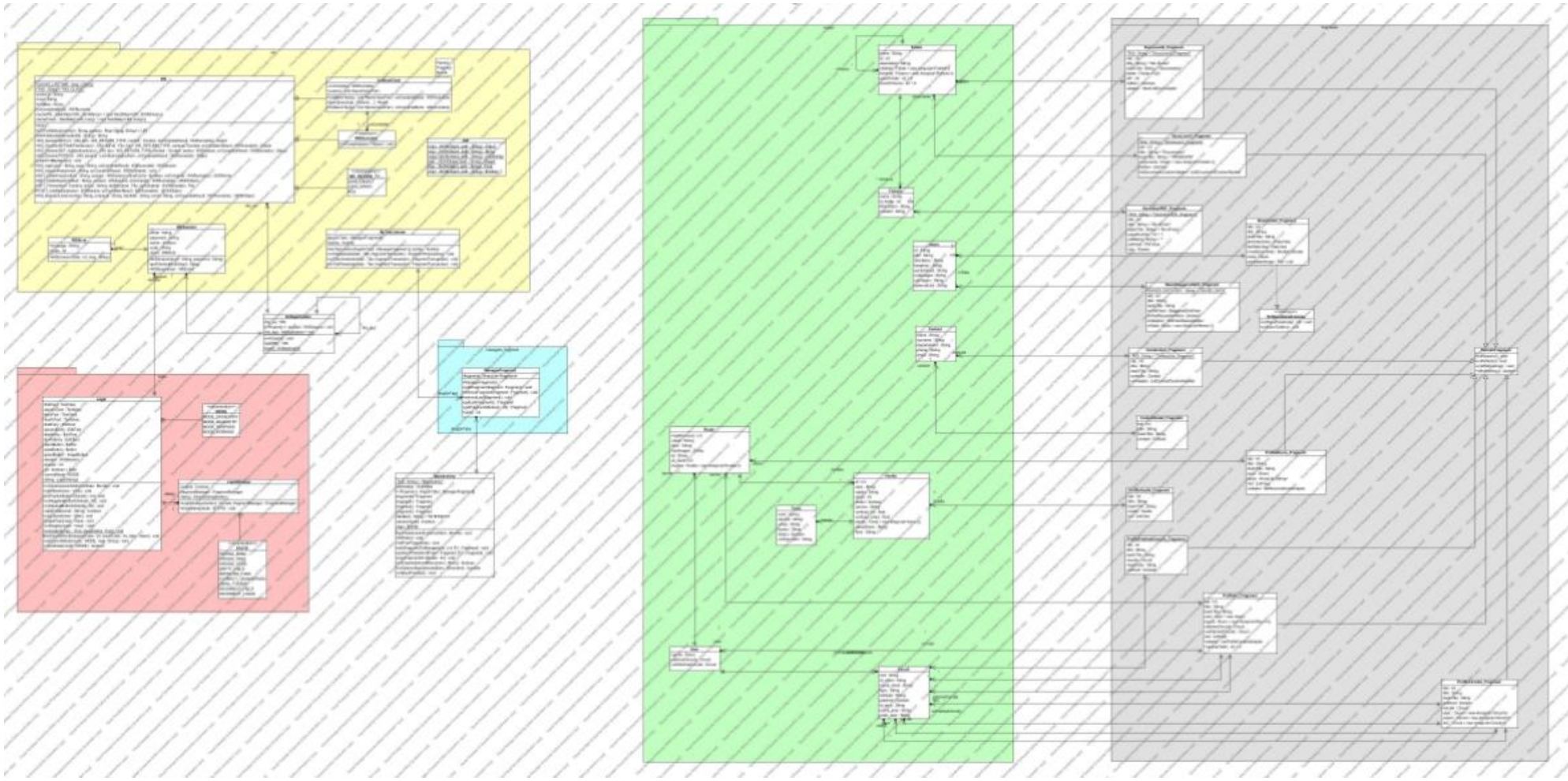


Figura 15: Diagrama de clases de Android

El conjunto de clases de la aplicación Android podría dividirse en tres partes, según sus funcionalidades:

5.4.1 Login
5.4.2 Comunicación con API REST del servidor web
5.4.3 Flujo general de la aplicación

A continuación se va a proceder a explicar cada una de las tres partes, así como sus clases contenidas.

5.4.1 Login

La primera actividad que se ejecuta nada más arrancar es la de Login, donde el usuario deberá identificarse. A efectos de aplicación móvil hay dos tipos de usuarios:

- **Cliente**: Para el cual está enfocada la aplicación. Tiene acceso a todos los módulos y en concreto, en el módulo 'Mi Cuenta' podrá consultar los servicios que tenga contratados con Norextin.
- **Invitado**: Es un rol que se ha creado con el fin de mostrar a los usuarios de aplicaciones móviles, que no sean clientes de Norextin, sus servicios para animarles a formar parte de ellos. Este tipo de usuarios no necesita tener credenciales asignadas y tendrán la posibilidad de acceder a todos los módulos de la aplicación excepto 'Mi Cuenta'.

La forma en que se loga un usuario forma parte de un sistema de autenticación básica en el que nombre de usuario y contraseña se envían como parámetros de una petición Http, codificados en Base64. Después será el servidor el que compruebe si las credenciales son correctas.

Las clases que se ocupan de desempeñar esta función son: Login, LoginDialogs, WSSession, WSError, RegisterActivity y InitApplication. Se detallan a continuación:

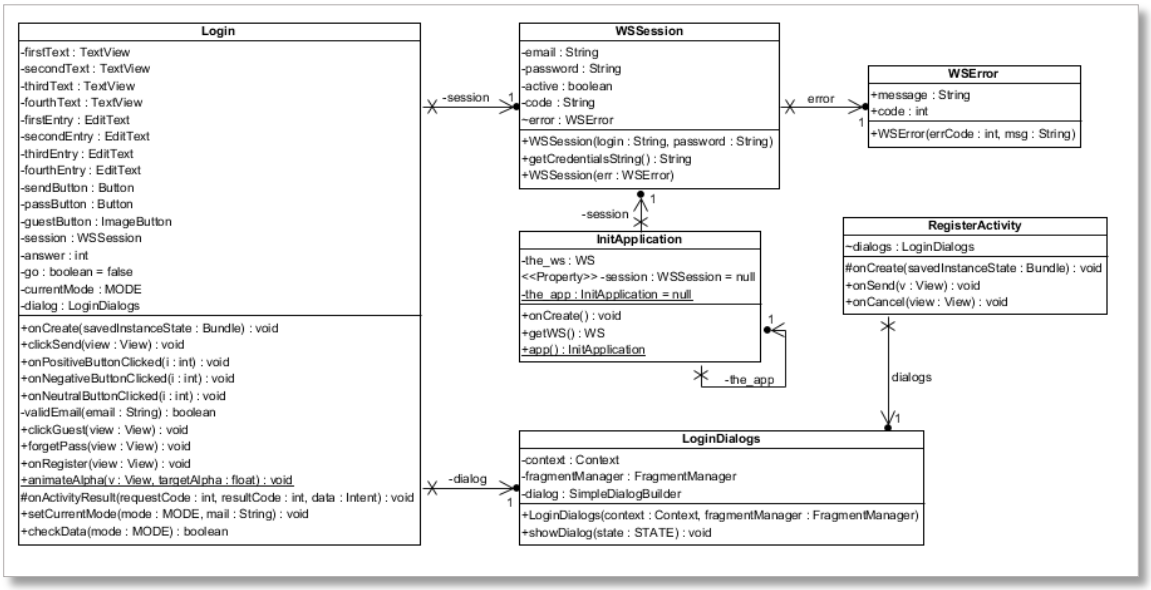


Figura 16: Clases para el Login

WSError.java

Clase que guarda posibles errores con el servidor durante la autenticación.

WSSession.java

Esta clase se instancia cuando el usuario ha conseguido realizar el login satisfactoriamente y tiene como atributos:

- Email → Con el que el usuario se identifica.
- Password → Contraseña.
- Active → Indica si la cuenta está activa en el servidor o no.
- Code → Código que se le asigna al usuario cuando reestablece la contraseña.
- Error → Instancia de WSError, para almacenar un posible error en la autenticación.

InitApplication.java

Extiende de Application, que son clases base de Android para aquellas aplicaciones que necesitan mantener el estado de la aplicación global. Al estar incluida en el AndroidManifest.xml se crea una instancia de la misma en cuanto se crea el proceso de la aplicación.

En concreto InitAppliation, a esta parte de la aplicación le sirve para mantener una instancia de WSSession de forma global y ofrece un método público de consulta *getSession()*, para que cualquier clase que lo necesite lo pueda utilizar.

LoginDialogs.java

En este proceso de autenticación, como en todos, puede aparecer una serie de errores fundamentalmente humanos que deben notificarse al usuario. En esta clase se definen todos esos posibles mensajes de error.

Login.java

Esta es la clase principal de esta parte, en ella residen los cuatro posibles casos que se pueden dar y se encarga de adaptar la interfaz a las circunstancias:

- MODE_DATAENTRY: Estado inicial de la interfaz, donde aparecen los campos de usuario y contraseña vacíos y un botón para navegar como invitado para que el usuario introduzca sus credenciales o entre como invitado.
- MODE_EMAILENTRY: La clase adoptará este modo cuando el usuario indique que ha olvidado su contraseña. Facilitará una dirección de correo electrónico para que el servidor le permita reestablecerla.
- MODE_NEWPASS: En este modo la clase Login modifica la interfaz para que el usuario introduzca una contraseña nueva.
- MODE_WORKING: Este modo simplemente muestra una barra de progreso mientras espera una respuesta por parte del servidor.

RegisterActivity.java

Es una actividad muy sencilla creada para que el usuario pueda solicitar al administrador el alta en la aplicación.

5.4.2 Comunicación con API REST del servidor web

Esta parte es fundamental en la aplicación, es la que se encarga durante toda la ejecución de surtir los datos. Por un número de clases parece sencillo, pero nada más lejos de la realidad.

Consta de tres clases: InitAplication, WS y JXP, que se detallan a continuación:

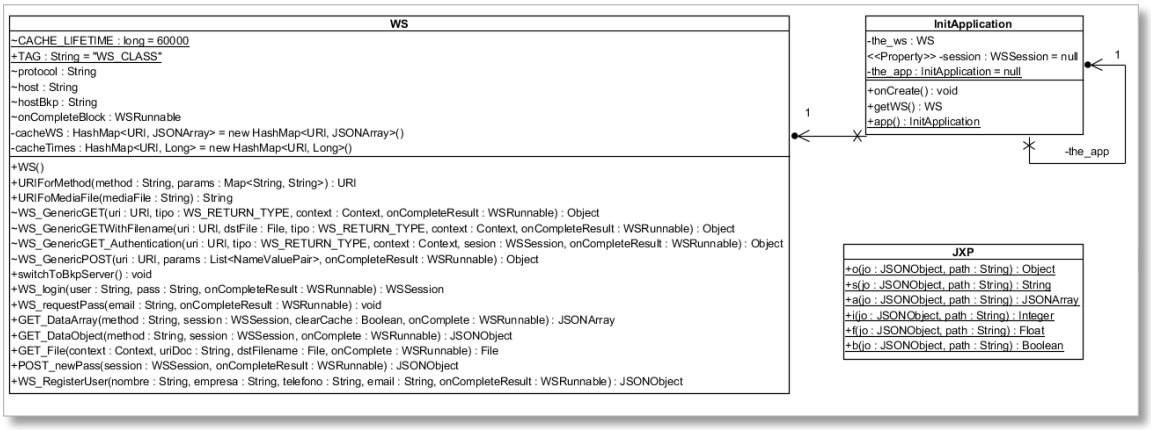


Figura 17: Clases para el API REST

InitAplication.java

Esta clase ya se ha mencionado en el Login que sirve para mantener el estado de la aplicación global. En cuanto a la comunicación con el servidor sirve para mantener una instancia del Web Service y poder acceder a ella desde cualquier clase.

WS.java

Abreviatura de Web Service y clase encargada de toda la lógica de comunicación entre móvil y web.

En ella se llevan a cabo todas las conexiones con el servidor, que se hacen en segundo plano gracias a la clase abstracta creada que extiende de AsyncTask.

En cuanto a los métodos encargados de ello, se ha buscado la mayor reutilización posible, dado que acciones como abrir la conexión, establecer protocolos de comunicación, parsear tipos de datos y demás son comunes para todos.

Entrando en detalle, se han dividido las peticiones que se realizan al servidor en cuatro métodos genéricos y reutilizables según sus características, con el fin de poder usarlos en futuros proyectos Android.

- **Método GET genérico:**

Llamada simple al método GET del API que recibe un Object (que será un JSON) y envía como parámetros: la url a la que va dirigida, el tipo de objeto que espera recibir, el contexto de la aplicación y el hilo que se ocupará de realizar el trabajo en segundo plano.

```
Object WS_GenericGET (URI uri,
                      WS_RETURN_TYPE tipo,
                      final Context context,
                      WSRunnable onCompleteResult)
```

- **Método GET con autenticación:**

Llamada similar a la anterior solo que incluye como parámetro un objeto de tipo WSSession. Este parámetro añadirá como parámetro Http las credenciales de usuario, para aquellas llamadas que lo soliciten.

```
Object WS_GenericGET_Authentication (URI uri,
                                     WS_RETURN_TYPE tipo,
                                     final Context context,
                        WSSession sesion,
                                     WSRunnable onCompleteResult)
```

- **Método GET para archivos:**

Los archivos que va a manejar este método son las imágenes de las noticias y los ficheros PDF de los documentos. Ambos no necesitan mecanismo de autenticación, luego no llevan ese parámetro. Pero sí necesita el objeto de tipo fichero (no JSON como los anteriores) para poder pasárselo a las funciones correspondientes de visualización.

```
Object WS_GenericGETWithFilename (URI uri,
                                  final File dstFile,
                                  WS_RETURN_TYPE tipo,
                                  final Context context,
                                  WSRunnable onCompleteResult)
```

- **Método POST genérico:**

La llamada al método POST del API recibe un Object y le añade a la url y el hilo, una lista genérica de parámetros que variará en función de los parámetros especificados en el API. En esta aplicación se hacen dos tipos de POST: el que envía la nueva contraseña cuando se reestablece y el que envía los datos de alta de un nuevo usuario.

```
Object WS_GenericPOST (URI uri,
                       List<NameValuePair> params,
                       WSRunnable onCompleteResult)
```

JXP.java

Esta clase es muy útil cuando se opera con objetos JSON. Es completamente genérica y reutilizable para cualquier proyecto Android que tenga que parsear este tipo de objetos. Su función es recorrer un objeto JSON determinado y parsearlo a su tipo de dato correcto. Los que están implementados son: Object, String, JSONArray, Integer, Float y Boolean. Pero sería muy sencillo incluir más si fuera necesario.

5.4.3 Flujo general de la aplicación

Antes de profundizar en cada una de las clases que constituyen esta parte, se va a explicar el comportamiento general que se desea para la aplicación.

Una barra, en la parte superior de la interfaz, será estática en toda la navegación y hará la función de menú principal. Inmediatamente debajo, cuatro submenús: Noticias, Documentos, Mi Cuenta y Contactos, actuarán como módulos independientes, con navegaciones independientes.

En la siguiente figura se presenta un esquema donde se puede ver cómo el Manager hace de controlador o Menú principal manejando una lista de submenús con estructura de pila de Fragments. Cada uno de los Fragments serán los que se vayan mostrando y ocultando a medida que el usuario vaya navegando.

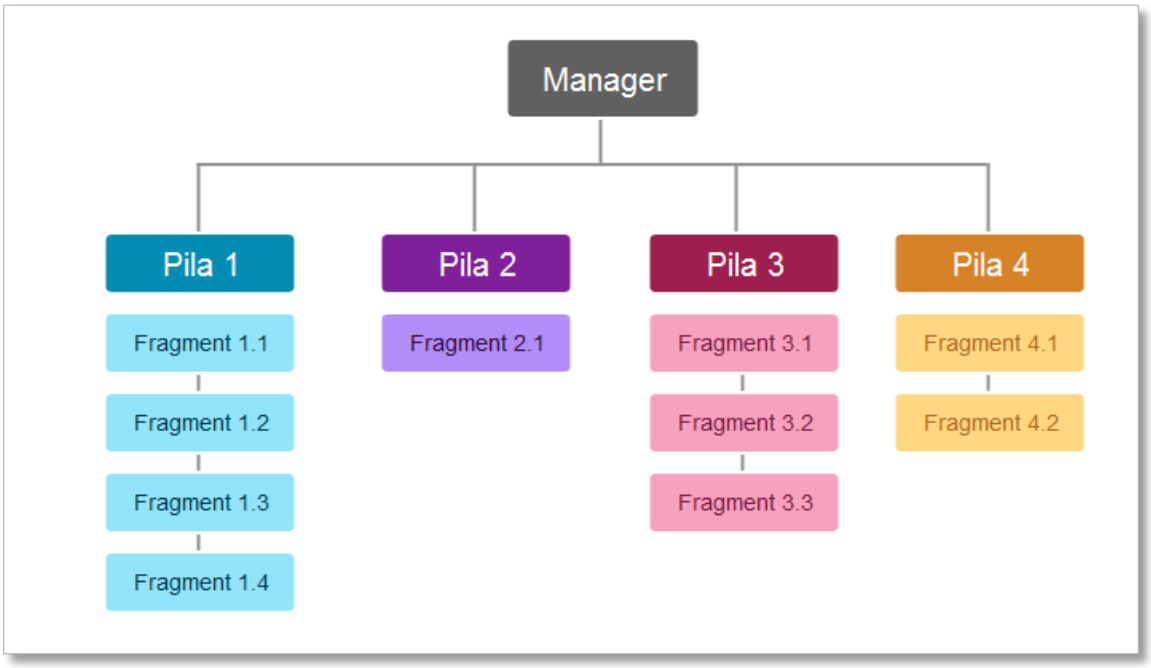


Figura 18: Estructura del flujo general de la aplicación Android

¿Cómo conseguir esto? No ha sido muy sencillo, dado que en Android no existe el componente que, los que programan para iOS tienen la suerte de poder utilizar: NavigationController. Este controlador maneja una lista de vistas y permite la navegación de una hasta otra en ambos sentidos y conservando el estado de cada una de ellas.

En cambio en Android, la manera más sencilla que se ha encontrado para llevar a cabo este comportamiento es: un Activity principal, que contenga los elementos de la interfaz estáticos en toda la aplicación y una colección de Fragments que simule la navegación entre submenús y dentro de los mismos [6, 7].

Con objeto de entender mejor esto conviene definir los conceptos [8]:

- **Activity**: Una actividad es un componente de la aplicación que proporciona una pantalla con la que los usuarios pueden interactuar con el fin de hacer algo. Cada actividad ofrece una ventana en la que se podrá dibujar la interfaz de usuario. La ventana normalmente llena la pantalla, aunque puede ser más pequeña (como va a ser el caso de este proyecto).

  Cada actividad tiene su ciclo de vida, el cual consta de varias fases como: iniciarse, pausarse, detenerse, destruirse…

- **Fragment**: Un fragmento representa un comportamiento o porción de interfaz de usuario en una actividad. Se puede pensar en un fragmento en una sección modular de una actividad, que tiene su propio ciclo de vida, recibe sus propios eventos de entrada y se puede agregar o quitar mientras que la actividad está en marcha.

  Un fragmento siempre debe estar integrado en una actividad y el ciclo de vida del fragmento siempre está directamente afectado por el ciclo de vida de la actividad que lo acoge.

En este proyecto, las clases: MainActivity, ManagerFragment y MyTabListener controlan el flujo de las pilas de Fragments, capturan los eventos de entrada y guardan el estado de navegación.

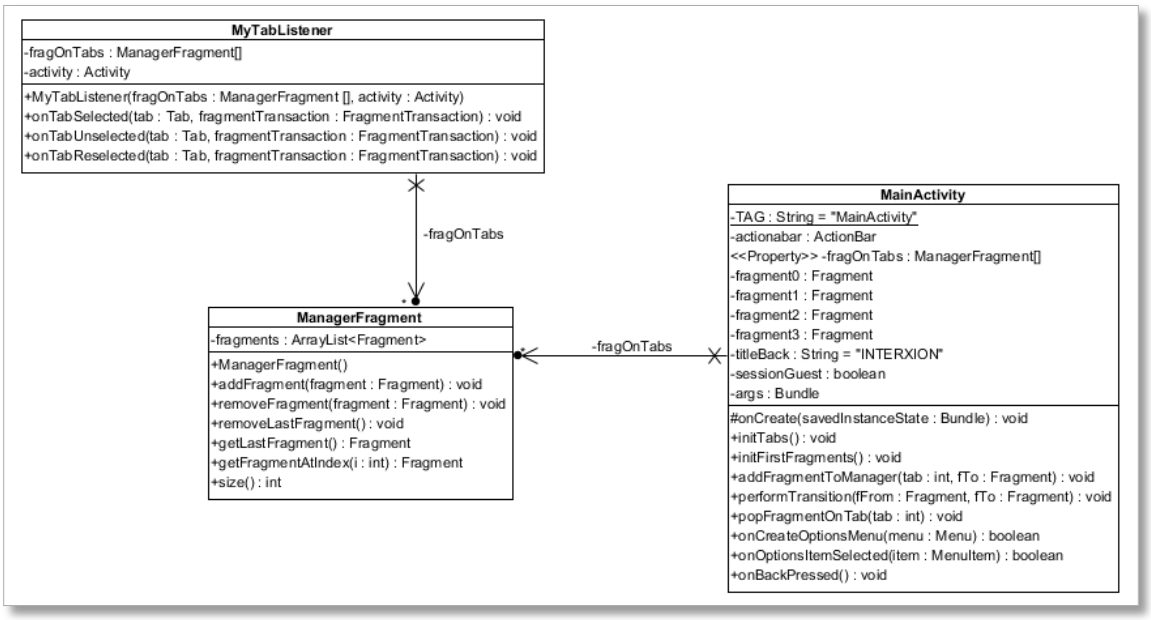


Figura 19: Clases para el flujo general de la App

ManagerFragment.java

Esta clase contiene una lista de Fragments y sus correspondientes métodos de inserción, borrado y consulta. Su función en la aplicación va a ser la de cada una de las cuatro

pilas de Fragments ya mencionadas, cada una será una instancia de esta clase. De su control se encargará la clase MainActivity.

MyTabListener.java

Este Listener será el que capture el evento de entrada que se produce cuando el usuario selecciona un módulo para mostrar en la interfaz. Se encarga de realizar las transiciones entre módulos y de avisar a la actividad principal del evento.

MainActivity.java

Esta es la actividad principal de la aplicación. Se inicia una vez que el usuario se ha logado (o entra como invitado), tras la actividad de Login.

Lo primero que hace es instanciar e inicializar las cuatro pilas de Fragments (con el primer Fragment por defecto de cada pila) y el Listener, de manera que todas las pilas tengan el primer Fragment preparado por si el usuario cambia de Tab (módulo).

También se encarga de llevar a cabo las transiciones entre los Fragments, actualizando la instancia de la pila correspondiente y salvando o recuperando el estado del Fragment anterior, con los siguientes métodos:

```
+ addFragmentToManager (tab: int, fTo: Fragment): void
```

Añade a la pila del módulo identificado con 'tab' el Fragment que le llega por parámetro 'fTo'.

```
+ performTransition (fFrom: Fragment, fTo: Fragment): void
```

Ejecuta la transición entre dos Fragments: el que estaba y se oculta 'fFrom' y el nuevo que se muestra 'fTo'.

```
+ popFragmentOnTab (tab: int): void
```

Elimina el último Fragment del 'tab' seleccionado. Lo elimina del Manager correspondiente y ejecuta la transición en la interfaz.

5.4.3.1 Módulos de la aplicación

Cada módulo conceptual de la aplicación se corresponde con cada una de las pilas de Fragments que conforman la estructura global. A continuación se muestran en detalle:

Módulo 'Noticias'

Las principales clases de este módulo son tres: NewsStaggeredFragment, NewsDetailFragment y News. Una pila de dos Fragments y el modelo de datos que se muestra.

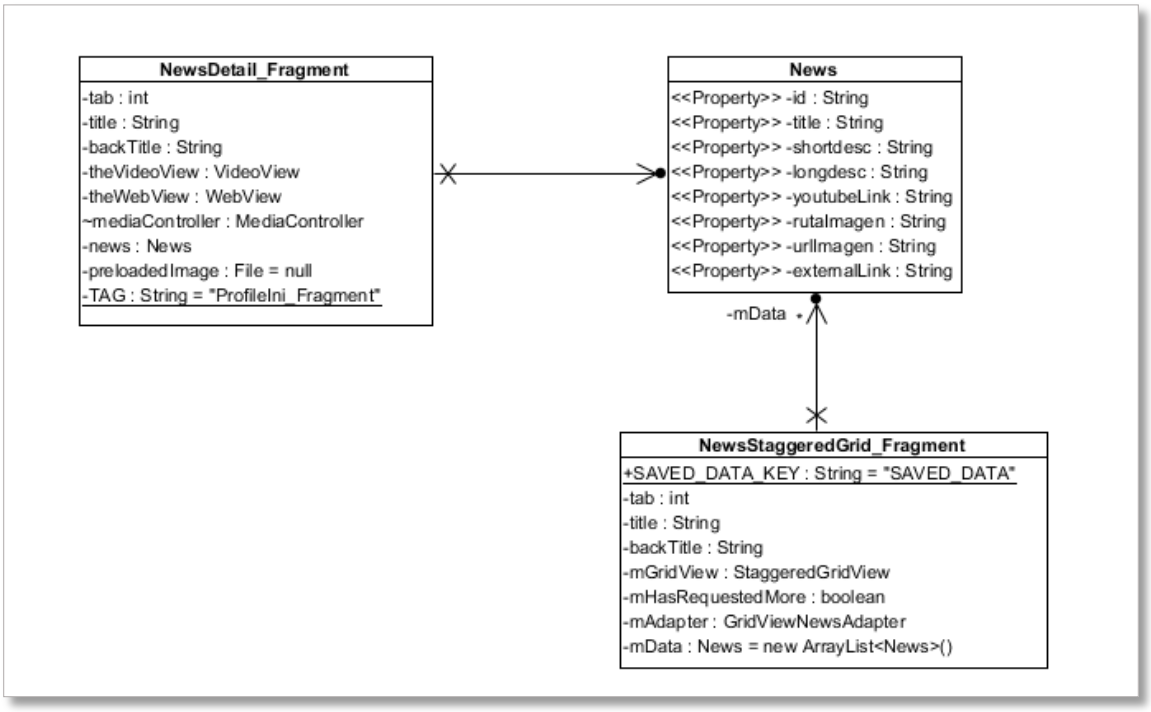


Figura 20: Clases para el módulo 'Noticias' en Android

NewsStaggeredFragment.java

Primero en la pila de Fragments. Contiene un GridView, importado de una librería eterna: '`com.etsy.android.grid:library`' que muestra en tamaño reducido cada una de las noticias recibidas por el API.

NewsDetailFragment.java

Último Fragment de la pila. Ofrece una vista detallada de la noticia que se haya seleccionado en la pantalla anterior y la posibilidad de compartir la noticia en RRSS.

News.java

Modelo de datos con el que se opera en este módulo. Las instancias de este objeto se crean a partir de los datos recibidos del servidor.

Módulo 'Documentos'

Las principales clases de este módulo son: Folder, Fichero, DocsLevel1_Fragment, DocsLevel2_Fragment y DocsViewPDF_Fragment. Una pila de tres Fragments y dos modelos de datos.

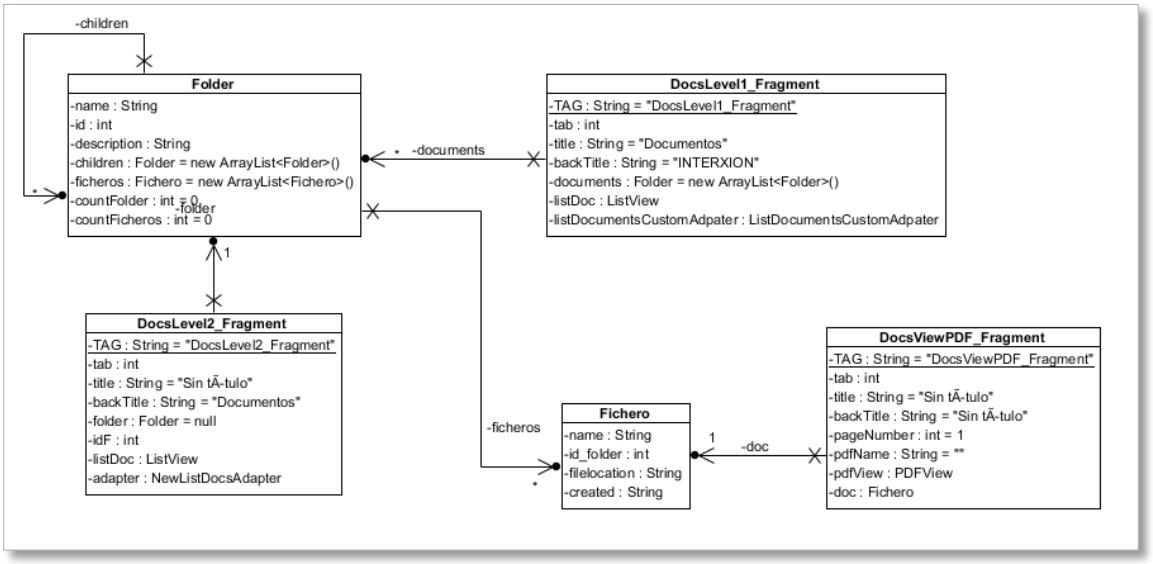


Figura 21: Clases para el módulo 'Documentos' en Android

DocsLevel1_Fragment.java

Primer Fragment de la pila. En esta pantalla sólo puede haber directorios de documentos y la interfaz tiene restricciones concretas, por ello se separa en una clase diferente a la siguiente.

DocsLevel2_Fragment.java

De este tipo de Fragment puede haber tantos en la pila como niveles en el árbol de directorios en el que está estructurado este módulo.

DocsViewPDF_Fragment.java

Último Fragment en la pila de Documentos si se navega hasta un fichero. Muestra el PDF correspondiente al documento seleccionado en el Fragment anterior. Lo visualiza gracias a una librería externa: `com.joanzapata.pdfview:android-pdfview`, dentro de la aplicación.

Folder.java

Modelo de datos que representa los directorios en los que se organizan los documentos.

Fichero.java

Objetos que encapsulan los datos de cada fichero.

Módulo 'Mi Cuenta'

Este módulo está habilitado sólo para usuarios de la aplicación registrados.

Las clases contenidas en 'Mi Cuenta' son una colección de Fragments y modelos de datos que se relacionan entre sí para conseguir la funcionalidad normal de la aplicación.

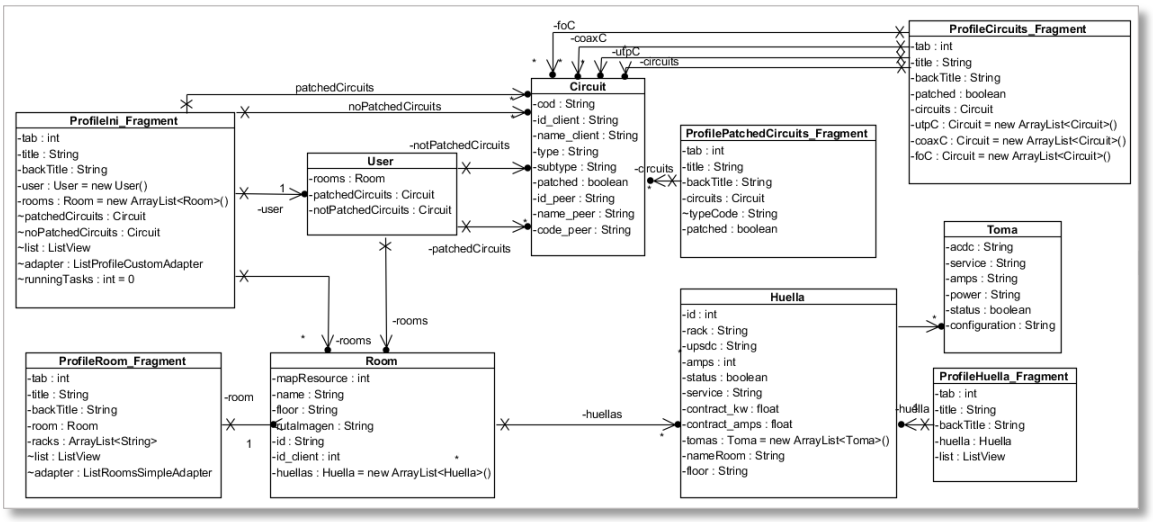


Figura 22: Clases para el módulo 'Mi Cuenta' en Android

ProfileIni_Fragment.java

Primero de los Fragments de la pila que se muestra. Muestra una lista con los nombres de las salas y los circuitos que el usuario tiene contratados.

ProfileRoom_Fragment.java

Es una vista de detalle de la sala seleccionada en el Fragment anterior. Contiene una imagen del mapa de la sala las huellas o racks contenidos en esa sala.

ProfileHuella_Fragment.java

Vista de detalle de la huella seleccionada.

ProfileCircuits_Fragment.java

Vista con el número de circuitos que tiene el cliente en cada uno de los 3 tipos posibles: coaxia, utp y fibra óptica.

ProfilePatchedCircuits_Fragment.java

Listado de circuitos parcheados o no. Ofreciendo la posibilidad de solicitar un parcheo por correo electrónico en caso de que no lo esté.

User.java

Modelo de datos para los usuarios. Con él se solicita la información correspondiente al usuario logado de todo este módulo.

Room.java

Representa una sala en el modelo de negocio de Norextin.

Huella.java

Lo que comúnmente se conoce como rack. Una huella debe pertenecer siempre a una sala.

Toma.java

Objetos que contienen la información de las tomas eléctricas de los racks.

Circuit.java

Modelo de datos que contiene los datos de los circuitos.

Módulo 'Contactos'

Último módulo de la aplicación que presenta una estructura muy similar a los anteriores. Sus clases son: ContactList_Fragment, ContactDetail_Fragment y Contact.

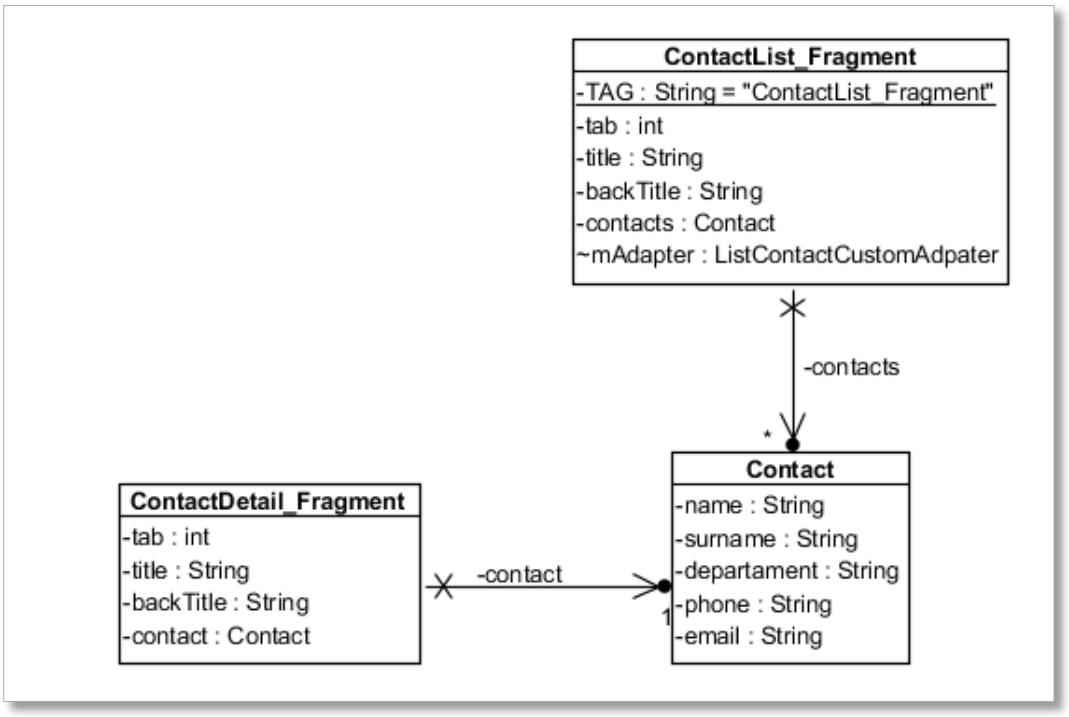


Figura 23: Clases para el módulo 'Contactos' en Android

ContactList_Fragment.java

Primer Fragmento de la pila dedicada a este módulo. Ofrece un listado de todos los objetos Contact y además la posibilidad de realizar una llamada o enviar un email a cada uno de ellos. Además de mostrar la información general de Norextin así como links a todas sus redes sociales.

ContactDetail_Fragment.java

Último Fragment de la pila con una vista de detalle del contacto seleccionado.

Contact.java

Modelo de datos del que se extrae la información para mostrarla en los Fragments.

5.5 Diseño de la interfaz para el portal de administración Web

Una vez se ha diseñado y definido toda la estructura interna del sistema, queda la parte más vistosa de la aplicación y la que, a pesar de todo, más suele importar a los clientes.

En concreto el framework Django ofrece un conjunto de vistas ya predefinidas para crear un portal de administración como éste, facilitando mucho el trabajo de diseño. Sobre esta premisa se han trabajado los CSS para darle al módulo una estática más corporativa, adaptando colores, logos y demás detalles.

Véase el resultado:

5.5.1 Login

La primera pantalla que se muestra en el navegador cuando el usuario administrador accede al portal es la de Login.

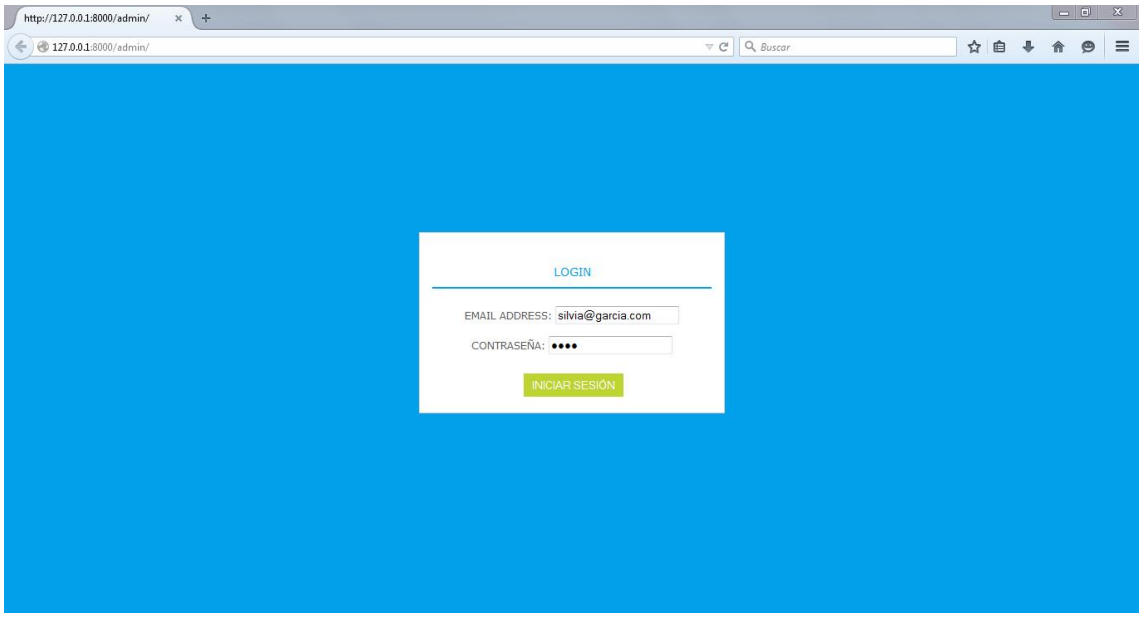


Ilustración 7: Pantalla de Login de la web de administración

Una vez el usuario ha introducido sus credenciales y el sistema las ha validado, tendrá la posibilidad de modificar de reestablecer la contraseña que se le haya dado por defecto en la siguiente pantalla:

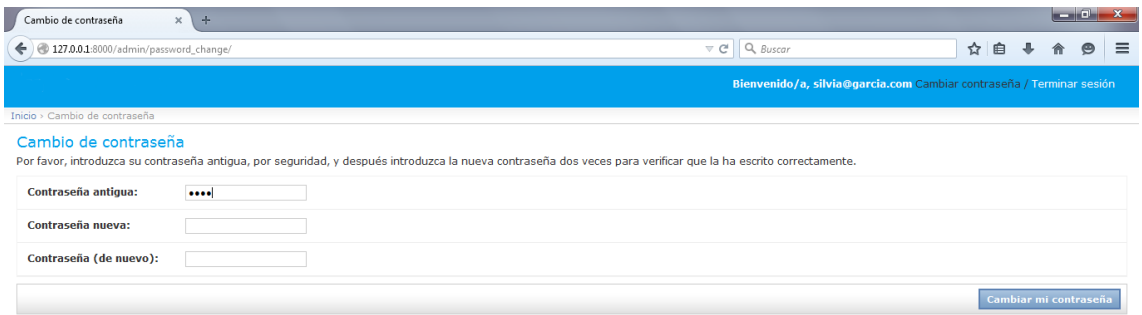


Ilustración 8: Pantalla para reestablecer la contraseña en la web de administración

5.5.2 Pantalla principal

La vista principal de la aplicación Web muestra un menú con todos los módulos a los que puede acceder el administrador. En cada uno de ellos, desde esta misma pantalla, podrá consultarlos, añadirlos o modificarlos al alcance de un solo clic. Además, en la parte derecha se visualiza un historial de las últimas acciones llevadas a cabo por él mismo.

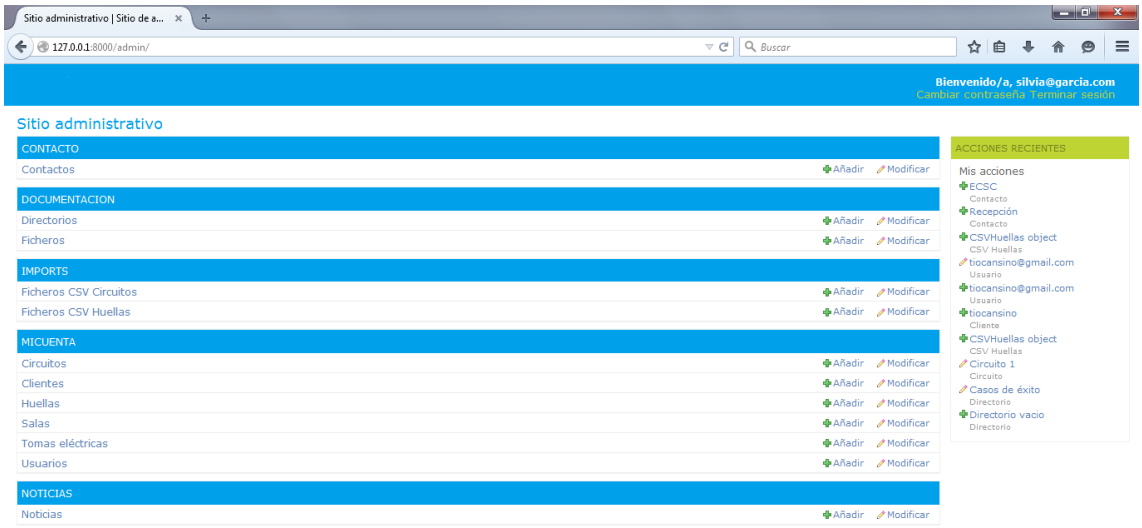


Ilustración 9 Pantalla de principal de la web de administración

5.5.3 Consulta de datos

Como ya se ha detallado en los requisitos y casos de uso, las acciones que se llevan a cabo con cada uno de los modelos de datos, son similares. Por ello se va a mostrar como ejemplo las pantallas destinadas a las operaciones con 'Huellas' sirviendo como ejemplo para todas las demás.

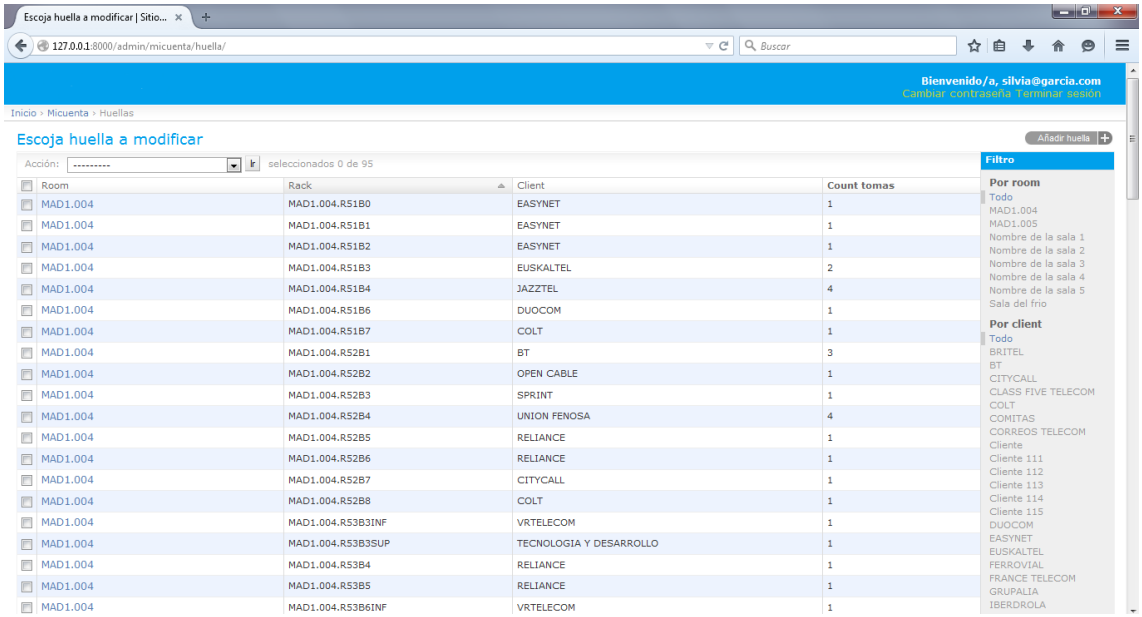


Ilustración 10: Pantalla de consulta de datos en la web de administración

Ésta es la vista que muestra todas las huellas que hay actualmente en la base de datos. En el lado derecho de la pantalla se ofrece un listado con todos los filtros que se le pueden aplicar a las huellas con el objetivo de encontrar más rápido lo que se busca.

5.5.4 Consulta/Modificación de un dato concreto

La pantalla destinada a la inserción, consulta o modificación de un modelo de datos concreto es igual, solo cambian los datos de las cajas de texto. En este caso se muestra la pantalla de una huella concreta como ejemplo.

Para los datos que, como en este caso las huellas, pueden incluir otros datos, se ofrece la posibilidad de insertarlos en la misma pantalla, aunque también se podría hacer aparte. Véase la siguiente huella y sus tomas eléctricas correspondientes:

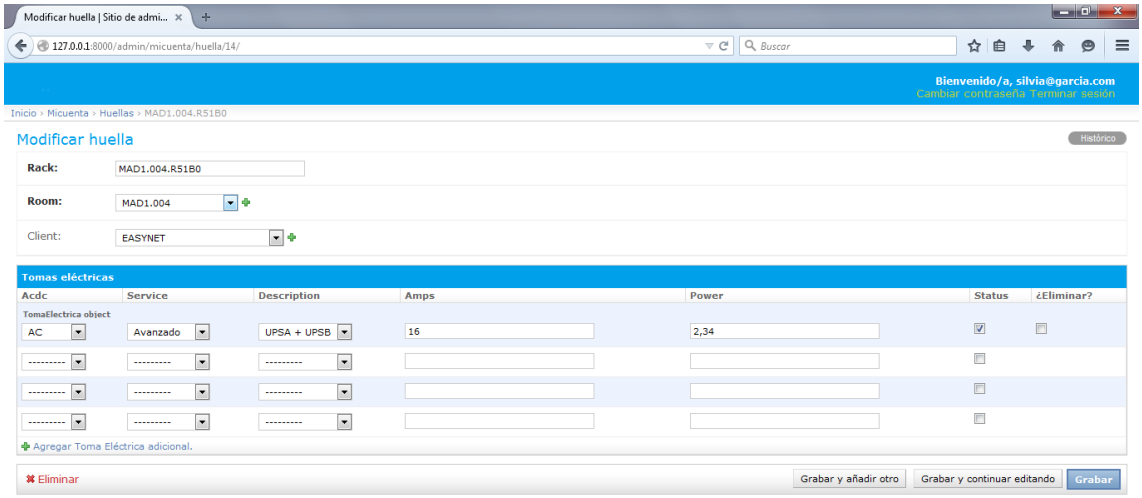


Ilustración 11: Pantalla de consulta/modificación de datos en la web de administración

5.5.5 Pantalla de error

A la hora de insertar o modificar algún dato, si el usuario no ha introducido todos los campos que son obligatorios o algún formato es incorrecto, el sistema se lo notificará impidiendo llevar a cabo la acción hasta que no se corrija el error.

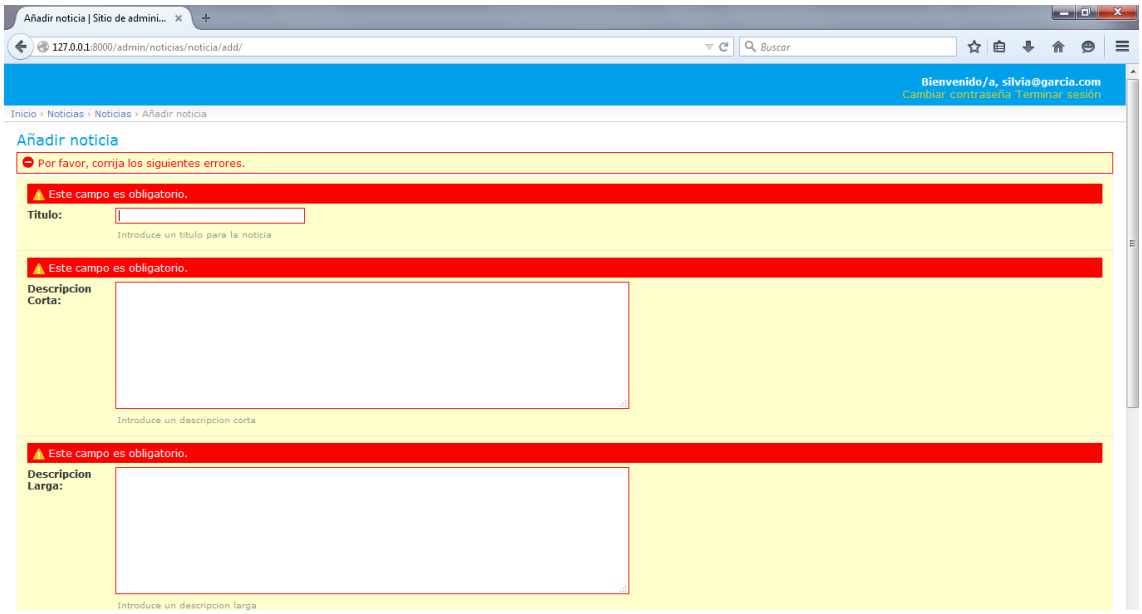


Ilustración 12: Pantalla de Error de la web de administración

5.6 Diseño de la interfaz para la aplicación Android

El diseño de esta parte del sistema es quizá en lo que más énfasis ha puesto el cliente. En las reuniones que se tuvieron con él manifestó su deseo de que fuera muy fácil de usar, con los cuatro módulos perfectamente definidos e independientes y con estética muy similar a la de la web.

A continuación se muestra una primera aproximación que se le plantea al cliente, esbozada en una herramienta de diseño gráfico. No se implementó nada hasta no obtener su confirmación.

Véase el resultado:

5.6.1 Icono de la App

El icono que se ha diseñado expresamente para la App sigue la línea corporativa del logo de la web, solo que con un toque un poco más actualizado.

Ilustración 13: Icono de la App

5.6.2 Login

La fase de Login engloba varias acciones posibles aparte de la obvia. En la primera figura se muestra la primera pantalla que se muestra nada más iniciarse la aplicación, la pantalla de Login. La segunda muestra las indicaciones que hay que seguir para reestablecer la contraseña en caso de olvido. Y en la tercera se piden los datos necesarios para solicitar el alta en el sistema.

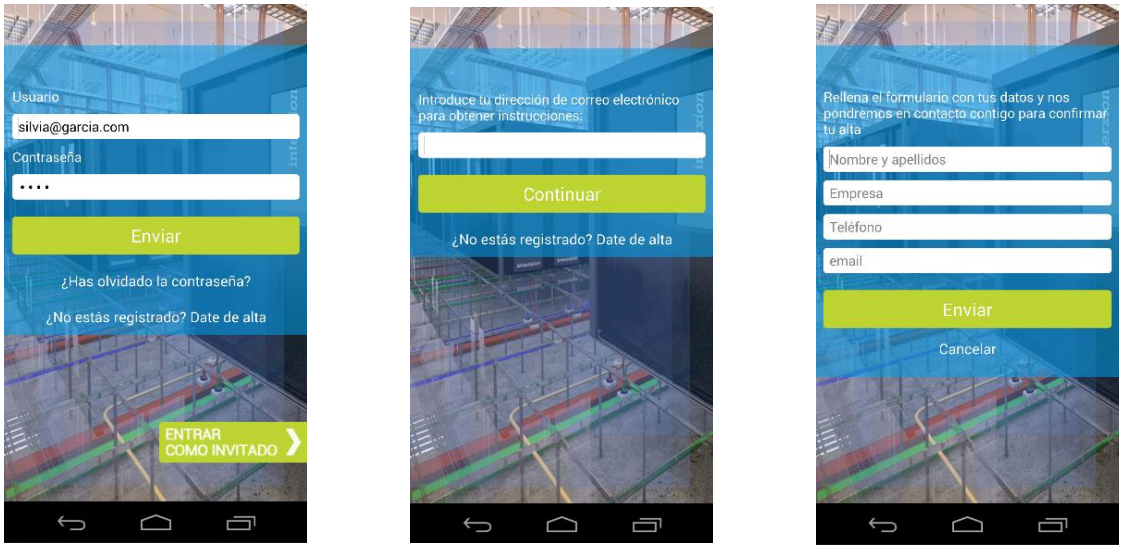


Ilustración 14: Pantallas de login de la App

5.6.3 Módulo NOTICIAS

El módulo en el que se publican las noticias del blog de Norextin consta de dos pantallas. La primera contiene un Grid con una versión mini de cada una de las noticias y la segunda es una vista en detalle de la que se haya pulsado.

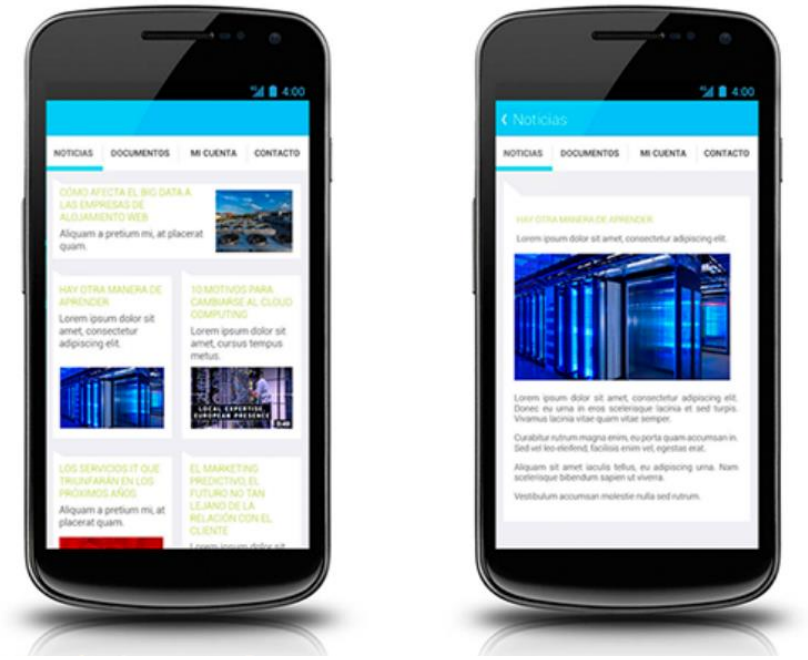


Ilustración 15: Pantallas del módulo 'Noticias' para la App

5.6.3 Módulo DOCUMENTOS

Para la parte de la aplicación dedicada a simular un sistema de archivos donde directorios y ficheros se distribuyen a placer del administrador, es decir, los muestra tal y como estén organizados en la web, habrá tantas pantallas como niveles de directorios tenga el árbol. La última pantalla refleja cómo debe visualizarse un PDF.

Ilustración 16: Pantallas del módulo 'Documentos' para la App

5.6.3 Módulo MI CUENTA

En la etapa del diseño de la aplicación no se tienen muy claros los datos que va a contener este módulo, pero el cliente sí tiene claro cómo deben mostrarse.

La primera pantalla muestra las salas y los circuitos que el usuario (logado) tenga contratados. A partir de esta pantalla, y según los servicios de los que disponga la cuenta, se muestran las demás.

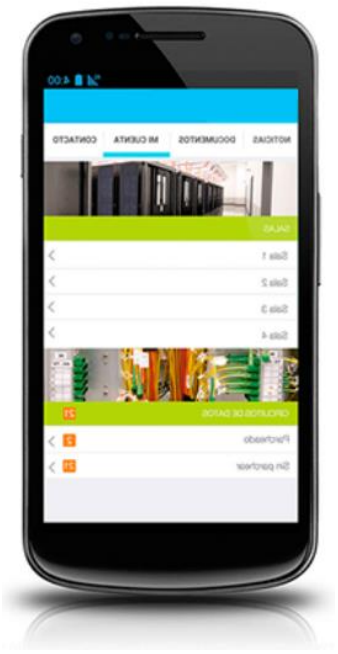


Ilustración 17: Pantalla 1 del módulo 'Mi Cuenta' de la App

A partir de aquí hay dos posibles caminos en la navegación:

- La visualización de una sala concreta al pulsar sobre alguna de ellas, como se indica en la figura 1. Mostrando el mapa de la sala y demás datos.
- La visualización de los circuitos parcheados o sin parchear, como se indica en la figura 2.

Ilustración 18: Pantallas para las Salas del módulo 'Mi Cuenta'

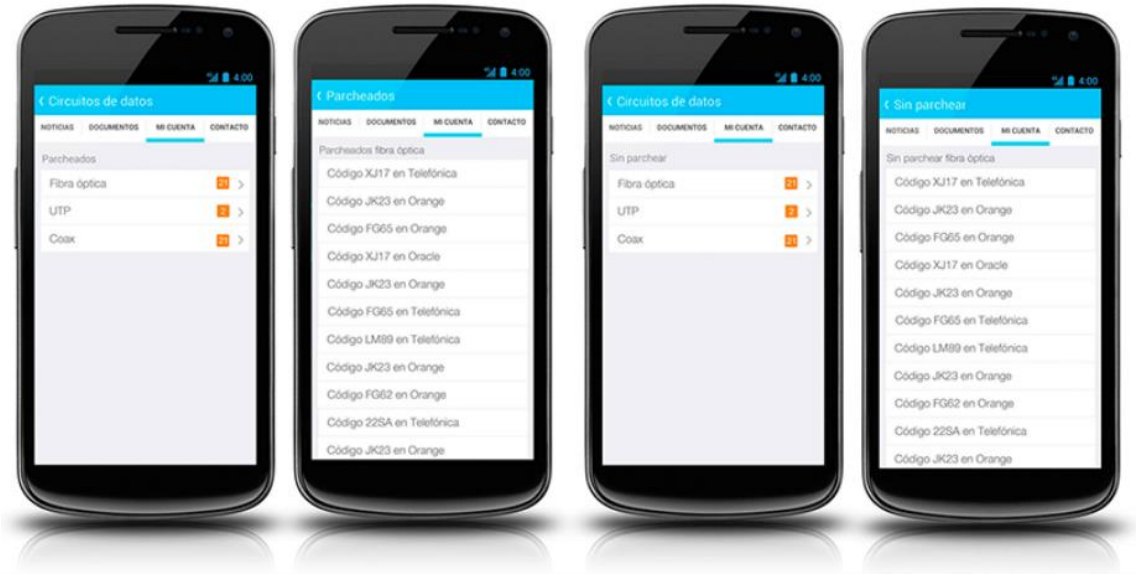


Ilustración 19: Pantallas para los circuitos del módulo 'Mi Cuenta'

5.6.4 Módulo CONTACTOS

Para terminar, se presenta el diseño de los contactos. No es más que un listado con los mismos y una vista de detalle para cada uno de ellos. De forma adicional se han incluido dos iconos que servirán como acceso directo a realizar una llamada, en el caso del teléfono o a enviar un email, desde el sobre.

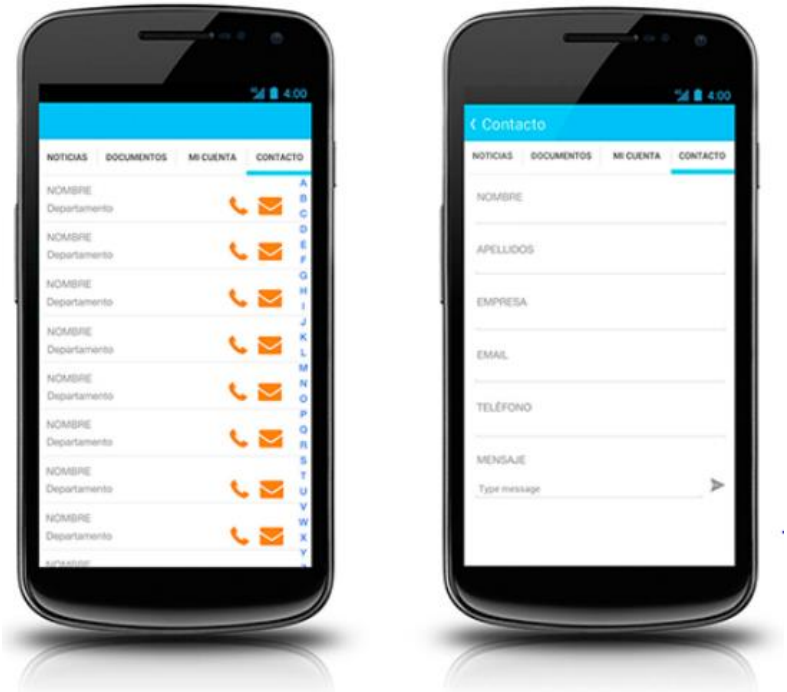


Ilustración 20: Pantallas del módulo 'Contactos' para la App

Capítulo 6

Planificación

Una vez definido el alcance del proyecto y sus requerimientos, la elaboración de la planificación es más realista. Aunque la planificación ha sido elaborada en la fase inicial del proyecto, se incluyen todas las tareas a llevar a cabo para cumplir con éxito todos los objetivos fijados.

En un proyecto software, como en cualquier otro tipo de proyecto, la planificación es fundamental. Pero cuando dicho proyecto tiene fecha estipulada de finalización marcada por un cliente, no sólo se hace más importante la planificación sino que hay que lidiar con el peor de los enemigos: el tiempo.

Éste, es un proyecto diseñado a medida para el cliente Norextin, quien establece dos requisitos primordiales: el tiempo y el dinero. En base a estas premisas se debe trazar un plan de desarrollo que permita llegar a tiempo, habiendo cumplido todos los objetivos y siendo rentable.

En esta sección se va a detallar, mediante un diagrama de Gantt, el tiempo que debe dedicarse a cada tarea con el propósito de llegar a tiempo al hito de final de proyecto. Se intentó seguir la filosofía de las metodologías ágiles, haciendo que cada miembro del equipo tuviera tareas independientes y se fijara objetivos que presentar en las reuniones semanales de seguimiento. Véase a continuación:

Figura 24: Diagrama de Gantt para la planificación del sistema

Duración total del proyecto:

El marco de tiempo en el que debe realizarse el desarrollo completo, estipulado por el cliente, es de 3 meses y 2 semanas.

- **Fecha de inicio**: 26 de junio de 2014
- **Fecha de finalización**: 10 de octubre de 2014

Desglose de intervalos por etapas:

El proceso de desarrollo se ha distribuido en 5 etapas de Ingeniería del Software:

Toma de requisitos: - Reunión con cliente para capturar - Reunión con cliente para validar	2 días
Análisis: - Identificación de subsistemas - Elección de tecnologías - Identificación de riesgos - Casos de uso	4.5 días
Diseño: - Arquitectura de componentes - Arquitectura de despliegue - Modelo de datos - Arquitectura de la aplicación web - Arquitectura de la aplicación móvil - Diseño de interfaz	7 días
Implementación del servidor web: - Preparación de entornos - Desarrollo de todos los módulos - Pruebas unitarias y de integración	22 días
Implementación de la aplicación móvil: - Interfaz REST - Desarrollo de todos los módulos	35 días
Instalación y formación: - Puesta en producción - Formación al usuario final	5 días

Capítulo 7

Conclusiones y desarrollos futuros

7.1 Conclusiones

Como ya se ha precisado a lo largo de la documentación, éste no ha sido sólo un Trabajo Fin de Grado, ha sido un desarrollo software analizado, diseñado e implementado a medida para un cliente real en una empresa real. Ésta ha sido la motivación que ha desencadenado cada una de las decisiones tomadas en los procesos de ciclo de vida del software.

La planificación del proyecto, una fase primordial para la gestión de tiempos, actividades, tareas, recursos… se llevó a cabo en la fase inicial. El cliente estipuló el intervalo de tiempo global que podía permitirse de duración del desarrollo y el deseo de revisar una serie de hitos intermedios. En base a esto se llevó a cabo la planificación con sus correspondientes errores de cálculo.

- Como era de esperar, la estimación de los tiempos de algunas tareas, fueron demasiado optimistas. En concreto, las tareas relacionadas con la construcción tanto del módulo web como de la aplicación móvil, sufrieron retrasos por motivos como el desconocimiento de algunos modelos de datos o la escasa experiencia desarrollando en las tecnologías elegidas.

- Uno de los riesgos señalados en el apartado 4.3, se convirtió en un problema real. El cliente no facilitó los datos del módulo 'Mi Cuenta' hasta estar muy avanzado el proyecto. Como plan de contingencia se aplazó el desarrollo de este módulo hasta que

no se pudo más, tomando la determinación de emprender una solución independiente. Gracias a esta decisión el retraso no fue catastrófico, aunque sí tangible.

- En cuanto a los procesos del ciclo de vida del software y su cronología, no han estado tan diferenciados ni ordenados como se ha mostrado en la documentación. La fase de toma de requisitos no sólo duró más de lo esperado, sino que la modificación de alguno de ellos se entremezcló con la fase de análisis y diseño. Y partes del diseño, como el de las interfaces de la aplicación móvil, tuvieron que presentarse al cliente tras la toma de requisitos.

Aunque no todo en este proyecto han sido retrasos. Con el permiso del lector voy a abandonar la tercera persona para exponer que, por circunstancias internas en la empresa donde se ha llevado a cabo el desarrollo de este proyecto, me ocupé personalmente de cubrir el puesto Directora para este Proyecto. Con el apoyo técnico de un miembro experimentado del equipo que no disponía de tiempo para asumir dicho rol, llevé a cabo todas las fases del desarrollo especificadas en este documento. Y, aunque son evidentes los errores, el trabajo salió adelante, el cliente quedó satisfecho y para mí resultó tremendamente enriquecedor como futura profesional.

El mundo de la consultoría informática es un mundo voraz y con mucha competencia en el que a veces no importa tanto el cómo sino el cuándo y por cuánto. Los conocimientos adquiridos en la universidad sirven para saber cómo hay que hacer las cosas y sobre todo: cómo hacerlas bien. Pero la salida al mundo laboral muestra la importancia de la eficacia a la hora de conseguir resultados.

Aunque para el desarrollador, un nuevo proyecto sea una posibilidad de hacer las cosas mejor y corregir errores del pasado, aun sabiendo que ello implicará un poco más de esfuerzo, no podrá hacerlo sin el respaldo de tres recursos imprescindibles: tiempo, dinero y personas cualificadas. Luego lo importante de verdad para poder competir en este mundo es saber aprovechar los recursos disponibles para conseguir los mejores resultados.

7.2 Líneas de trabajo futuro

Para la rentabilidad de cualquier negocio, el hecho de realizar una inversión sobre un producto en principio no necesario, o para la mejora de un servicio ofrecido, supone un riesgo. Norextin quiso asumir ese riesgo, pero cuidando que el impacto económico sobre la empresa fuese mínimo.

Por este motivo y por querer tomar la delantera a la competencia, el desarrollo software que se ha diseñado podría decirse que es low cost. El deber de ajustar el presupuesto al máximo ha llevado a tomar decisiones sabiendo que se podrían mejorar. Pero a veces no vale con saberlo, hay que demostrarlo.

El objetivo de esta primera etapa del sistema (la que engloba este proyecto) era que el cliente probara una primera versión de la aplicación en su negocio y valorara su utilidad y rentabilidad. Para cuando el cliente se dé cuenta de las ventajas que ofrece este desarrollo, ya habrá una segunda fase pensada con las siguientes mejoras:

- Como ya se expuso en el estado del arte, la plataforma iOS con todos sus dispositivos Apple, es el gran (por no decir único) rival de Android. Por ello, desarrollar esta misma aplicación para "la otra mitad del mundo" que utiliza dispositivos Apple, garantiza a Norextin llegar a casi la totalidad del mercado de smartphones que existe en la actualidad.

- Por seguir en el lado de la aplicación móvil del sistema, mencionar la ausencia de una base de datos en el dispositivo (aunque se hace uso de la caché por defecto de Android) para minimizar el tránsito de peticiones al servidor, dar la posibilidad de administrar desde el cliente móvil y ofrecer funcionalidad sin la necesidad de tener que estar conectado a la red.

  En concreto para Android, se ha estudiado la posibilidad de incluir CouchDB como motor de base de datos, el cual ofrece muchas ventajas para sistemas este tipo de sistemas, como su naturaleza distribuida, su tolerancia a pérdidas de conexión o su modelado de datos orientado a documentos.

- En cuanto a la Web, también se podría mejorar la base de datos. SQLite, motor de base de datos actual, almacena los datos en un archivo único, siendo más ligero y rápido, pero en principio no está preparado para un gran volumen de datos.

  Como alternativas podrían estudiarse MySQL, una base de datos muy robusta, que permite backups automáticos y podría hacerse distribuida si el volumen creciese mucho. O PostgreSQL, que también cumple estos requisitos.

  O como opción diferente y favoreciendo el manejo de JSON y comunicación con el móvil, el servidor de base de datos CouchBase. Que entre sus ventajas incluye una interfaz REST desde la que acceder a todos los ítems vía HTTP [9].

- Ahondando un poco en la propia funcionalidad de la aplicación y satisfaciendo una necesidad que el cliente ha expresado tener, la posibilidad de exportar toda la información contenida en la base de datos en formato XML sería una gran ventaja.

- Por último, Norextin es una empresa que opera a nivel europeo con países como Francia, Alemania o Reino Unido. Su página web ofrece soporte para multitud de idiomas, por tanto es esencial que el servicio que brinda este desarrollo también lo haga.

Parte III

Apéndices

Apéndice A

Presupuesto

Este capítulo presenta un análisis económico preliminar, puesto que todavía no se ha realizado la implementación de la propuesta. Es más bien una estimación de costes en base a los tiempos estipulados de planificación para, con ello, realizar una oferta a potenciales clientes.

Para realizar una estimación lo más realista posible es necesario fijarse tanto en costes de personal, como en costes de software y hardware.

Coste de personal

Tomando como base la planificación, el desarrollo a realizar supondría aproximadamente un trabajo de 15 semanas, en las que deben intervenir diferentes perfiles de profesionales. Haciendo una estimación del esfuerzo que debe realizar cada perfil en cada etapa del desarrollo, se ha cuantificado la cantidad de trabajo en horas y costes:

Perfil	Horas	Coste/Hora	Total
Gestor de proyecto	68	50 €	3.400 €
Ingeniero senior	80	35 €	2.800 €
Ingeniero junior	456	20 €	9.120 €
Diseñador gráfico	15	25 €	375 €
		TOTAL	15.695 €

Tabla 35: Tabla de costes de personal

Coste de hardware y materiales

Para la realización de este proyecto se debe disponer de determinados elementos hardware que cumplan unos requisitos mínimos. Por un lado está la fase de desarrollo o construcción del software, que debe llevarse a cabo en un ordenador de gama media-alta, para que el entorno de desarrollo se pueda ejecutar sin problemas y por otro lado está la fase de despliegue, donde residirá el servidor que deberá estar siempre conectado.

Recurso	Precio
Ordenador de sobremesa	450 €
Alquiler de servidor y dominio	150 €
TOTAL	600 €

Tabla 36: Tabla de costes hardware

Coste de software

Los costes de software son mínimos porque la mayoría de las herramientas que se utilizan en este proyecto son de licencia libre. Sólo se tienen en cuenta la licencia del sistema operativo del ordenador de sobremesa y de la herramienta para realizar la documentación. Además, para poder subir la App a Google Play hay que tener una cuenta

Recurso	Precio
Microsoft Windows 7	239 €
Microsoft Office 2013	119 €
Licencia Google Play	23 €
TOTAL	381 €

Tabla 37: Tabla de costes software

Coste total del proyecto

El coste total asumido en la realización de este proyecto sería la suma de los costes desglosados, sin incluir el IVA. Véase en la siguiente tabla cómo queda:

Costes desglosados	Precio
Personal	15.695 €
Hardware	600 €
Software	381 €
I.V.A. (21%)	3.501.96 €
TOTAL	20.177.96 €

Tabla 38: Coste total del proyecto

Apéndice B

Documentación del API REST

La arquitectura REST (Representational State Transfer) es la arquitectura de desarrollo web más estándar para crear APIs para servicios orientados a Internet. Al apoyarse totalmente en el estándar HTTP, permite que cualquier dispositivo o cliente que entienda este protocolo pueda utilizarlo.

La forma de utilizarla es muy sencilla. El servidor que implementa el servicio pone a disposición de los clientes una serie de recursos a los que se podrá acceder mediante URIs (Uniform Resource Identifier), identificadores unívocos de los recursos que no sólo permiten localizarlos sino compartir su ubicación.

Para manipular los recursos, HTTP proporciona una serie de métodos, pero en este proyecto sólo se van a utilizar los métodos:

- GET: Para consultar recursos.
- POST: Para crear recursos.

Para que un cliente HTTP pueda consultar un recurso del servidor, basta con acceder a la URI específica y recibirá el recurso en formato JSON.

Django, proporciona un conjunto de herramientas de gran alcance y flexibilidad, django-rest-framework, que hace que sea muy fácil construir APIs Web [11].

A continuación se detalla cómo en muy pocos pasos se ha conseguido crear una API que preste servicio Web a la aplicación móvil:

1. Lo primero es instalar en el proyecto Django la herramienta con el sistema de gestión de paquetes usado en Python: pip. Con sólo un comando está listo.

```
>> pip install djangorestframework
```

Y a continuación hacérselo saber al archivo de configuración: settings.py

```
INSTALLED_APPS = (
    ...
    'rest_framework',
)
REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': (
        'rest_framework.authentication.BasicAuthentication',
    ),
    'DEFAULT_RENDERER_CLASSES': (
        'rest_framework.renderers.JSONRenderer',
        'rest_framework.renderers.BrowsableAPIRenderer',
    )
}
```

2. Una vez instalado se puede empezar a trabajar.

   En primer lugar hay que definir algunos Serializers. Los Serializadores permiten converir los datos complejos como QuerySets e instancias de Model a los tipos de datos nativos de Python que luego pueden ser fácilmente transmitidos en JSON. Los Serializadores también proporcionan deserialización, permitiendo que los datos analiza para convertirse de nuevo en tipos complejos, después de validar primero los datos entrantes [13].

   Para definirlos, se crea un nuevo módulo en el proyecto llamado serializers.py que se usará en las representaciones de datos. A continuación se muestra el de 'Noticia' como ejemplo:

```
class NoticiaSerializer(serializers.HyperlinkedModelSerializer):
    class Meta:
        model = Noticia
        fields = Noticia.__all__
```

3. Una vez hecho esto, se crea la vista correspondiente.

   ViewSet es una clase propia de Django que agrupa todo el comportamiento común de los datos facilitando el uso. Ordena en una vista predefinida con dos simples líneas la vista para la API.

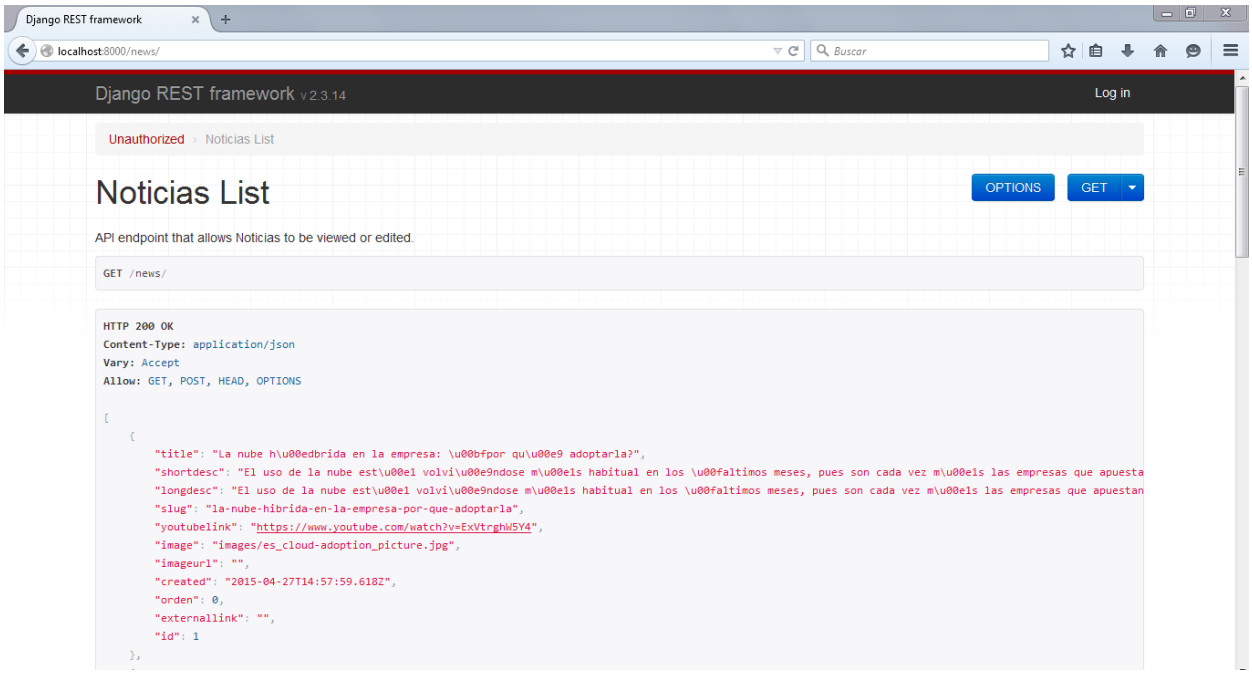
```
class NoticiasViewSet(viewsets.ModelViewSet):

    queryset = Noticia.objects.order_by('orden','-created')[:15]
    serializer_class = NoticiaSerializer
```

4. Al usar ViewSets en lugar de crear Views específicas, se puede generar automáticamente la URL para la API, simplemente al registrar los ViewSets con una clase de router.

```
router = routers.DefaultRouter()
router.register(r'news', newsview.NoticiasViewSet)
```

Y con estos sencillos pasos ya se puede acceder a las noticias del sistema, desde el navegador o desde cualquier otro dispositivo, introduciendo la ip y el puerto del servidor donde se haya hecho el despliegue más la URI específica:

La respuesta, en este caso para un navegador Web, será similar a la siguiente:



Apéndice C

Instalación en producción del servidor

En este manual se detallan los pasos necesarios para la puesta en producción de la aplicación servidora desarrollada en Python y Django. El manual se dividirá en dos partes: identificación de requisitos hardware y software e instalación de la solución.

Requisitos hardware

Los requisitos mínimos recomendados para la correcta ejecución de la aplicación son los siguientes:

Memoria RAM	Número de procecsadores	Cabina de almacenamiento	Conectividad
8 GB	2 cores x 2.1 GHz	100 GB / 7.200RPM	Ethernet 1GB

Tabla 39: Requisitos hardware mínimos para despliegue

Estos requisitos garantizan la correcta ejecución de la aplicación bajo una carga media-baja de utilización. Para una mayor garantía de reacción ante situaciones de carga puntuales se recomienda una configuración algo más potente, principalmente en la utilización de memoria RAM.

Memoria RAM	Número de procecsadores	Cabina de almacenamiento	Conectividad
16 GB	4 cores x 2.1 GHz	100 GB / 7.200RPM	Ethernet 1GB

Tabla 40: Requisitos hardware deseables para despliegue

Este caso se va a garantizar tener disponible en memoria gran parte de las transacciones con la base de datos, agilizando el tiempo de respuesta del API a los clientes que hagan uso de ella. El aumento de cores es recomendable para garantizar además la inmediata respuesta del portal web de administración, más que suficiente para cubrir el volumen de usuarios inicialmente planteados por Norextin.

Requisitos software

Es necesario instalar el sistema operativo CentOS 6 x86_64 versión *minimal* en la máquina donde se situará el servidor de la aplicación Norextin. Será la propia empresa Norextin la que se encargue de este paso, no siendo necesaria la intervención por parte del equipo de instalación.

El siguiente paso será definir el usuario de administración de la plataforma. Para ello el instalador debe ejecutar los siguientes comandos con usuario root.

```
useradd norextin
password norextin 1nt3rx10n
```

A continuación se debe instalar el software base que permitirá ejecutar la aplicación servidora, python 3.3 y django 1.6. El instalador debe ejecutar el siguiente comando con usuario norextin.

```
sudo yum install python python-django
```

Este comando instalará la última versión disponible de python y el framework django, necesario para la ejecución del servidor.

Instalación del servidor

El siguiente paso tras la configuración del sistema operativo y el software base es la puesta en marcha de la aplicación servidora. Para ello ejecute los siguientes comandos con el usuario norextin.

```
mkdir /norextin
cp <install_resources> / norextin / norextin
```

El directorio <install_resources> será la unidad con el software original que debe entregarse al cliente. Llegados a este puto debe existir en el servidor un directorio /norextin con el software de la aplicación.

A continuación se verificará el correcto funcionamiento del servicio arrancando la aplicación de forma manual.

```
python manage <server_ip><port_server>
```

El instalador debe verificar que la aplicación arranca con normalidad, tras lo cual debe detenerla.

Creación de la base de datos

La aplicación contiene los modelos de base de datos que deben crearse por primera vez para el correcto funcionamiento del servidor. Para la creación automática de todo el modelo de datos debe ejecutar la siguiente instrucción con usuario norextin.

```
python manage syncdb
```

NOTA: Este comando sólo debe ejecutarse una vez antes de la puesta en marcha de la aplicación.

Creación de servicios

La forma correcta de ejecutar la aplicación será a través de un servicio instalado en el sistema operativo. El primer paso es crear el script de arranque del servidor con usuario norextin.

```
cd /norextin
vim starup.sh
```

Copie el contenido del siguiente script para el arranque del servidor

```
#!bin/sh
python manage runserver <server_ip><server_port>
```

Almacene el fichero y proporcione permisos de ejecución.

```
chmod +x startup.sh
```

A continuación debe crearse el fichero de parada controlada de la aplicación. Para ello ejecute los siguientes comandos con usuario norextin.

```
cd /norextin
vim shutdown.sh
```

Copie el contenido del siguiente script para la parada del servidor.

```
kil ‘ps aux | grep python | grep –v grep | awk ‘{`print $2}’’
```

Almacene en el fichero y proporcione permisos de ejecución.

```
chmod +x shutdown.sh
```

Por ultimo se debe crear el servicio que controlará los scripts antes definidos. Para ello inicie sesión con usuario root y teclee los siguientes comandos.

```
cd /etc/init.d
vim norextin
```

Esto creará un fichero que será el servicio de sistema operativo para arrancar la aplicación. Dentro de este fichero copie el siguiente script.

```
#!/bin/sh
# chkconfig: 345 99 10
# description: Norextin auto start-stop script.
#

OWNER=norextin
HOME=/norextin
case "$1" in
    'start')
        # Start the django server:
        # Remove "&" if you don't want startup as a background process.
        su $OWNER -c nohup “$HOME/norextin/startup.sh >> $HOME/norextin/server.log 2>&1" &

        ;;
    'stop')
        # Stop the django server:
        su $OWNER -c nohup "$HOME/norextin/shutdown.sh >> nohup $HOME/norextin/server.log 2>&1" &
        ;;
esac
```

Guarde el fichero y proporcione permisos de ejecución.

```
chmod +x norextin
```

Llegados a este punto ya está disponible el servicio del sistema operativo para la ejecución de la aplicación. Para conseguir que se ejecute de forma automática durante el arranque teclee el siguiente comando.

```
chkconfig norextin on
```

Ya está en disposición de probar el servicio con los comandos.

```
service norextin start/stop
```

Configuración del firewall

Se va a emplear iptables como firewall por defecto del sistema operativo. Se debe abrir una regla que permita el acceso al servidor desde cualquier dispositivo. Con usuario root teclee el siguiente comando.

```
iptables –A INPUT –p tcp –dport 8000 –j ACCEPT
```

Esto permitirá las conexiones TCP por el puerto 8000 desde fuera del servidor. Si cambia el puerto de arranque en el fichero startup.sh debe cambiar también la regla del firewall asociada.

Parte IV

Bibliografía

Bibliografía

[1] Intellego: Aplicaciones móviles, un mercado de gran crecimiento. http://www.intellego.com.mx/es/noticias/aplicaciones-moviles-un-mercado-de-gran-crecimiento-en-mexico-y-en-el-mundo, 2014.

[2] Yeeply: Economía App: hábitos y uso de aplicaciones móviles. https://www.yeeply.com/blog/economia-app-habitos-y-uso-de-aplicaciones-moviles/, 2014.

[3] MovilZona: Datos de mercado en el primer trimestre de 2015. http://www.movilzona.es/2015/05/06/datos-mercado-primer-trimestre-2015-ios-windows-phone-android-kantar-worldpanel/, 2015.

[4] HotFrameworks: Find your new favorite web framework. http://hotframeworks.com/, 2015.

[5] Django Documentation. https://docs.djangoproject.com/en/1.8/, 2015.

[6] Gironés, Jesús Tomas. El gran libro de Android. Marcombo, 2013.

[7] Gironés, Jesús Tomás. El gran libro de Android Avanzado. Marcombo, 2ª Edición, 2013.

[8] Android Documentation. http://developer.android.com/guide/index.html, 2015.

[9] CouchDB: La guía definitiva. http://guide.couchdb.org/editions/1/es/why.html.

[10] AsierMarques: Conceptos sobre APIs REST. http://asiermarques.com/2013/conceptos-sobre-apis-rest/, 2013.

[11] Django REST framework: http://www.django-rest-framework.org, 2015.

[12] Django Serialization: http://www.django-rest-framework.org/tutorial/1-serialization/#creating-a-seria