ESCUELA POLITÉCNICA

UNIVERSIDAD DE EXTREMADURA

ESCUELA POLITÉCNICA

GRADO EN INGENIERÍA INFORMÁTICA EN INGENIERÍA DEL SOFTWARE

TRABAJO FIN DE GRADO

Aplicación para facilitar la difusión de información del Banco de Sangre de Extremadura

Álvaro Crespo Cardoso

Febrero, 2019

ESCUELA POLITÉCNICA

UNIVERSIDAD DE EXTREMADURA

ESCUELA POLITÉCNICA

GRADO EN INGENIERÍA INFORMÁTICA EN INGENIERÍA DEL SOFTWARE

TRABAJO FIN DE GRADO

Aplicación para facilitar la difusión de información del Banco de Sangre de Extremadura

Autor: Álvaro Crespo Cardoso

Tutor: Pedro José Clemente Martín

Tribunal Calificador:

Presidente: Elena Jurado Málaga

Secretario: Alberto Gómez Mancha

Vocal: Álvaro Prieto Ramos

Resumen

El Banco de Sangre de Extremadura es una Institución Sanitaria dependiente del Servicio Extremeño de Salud que es la encargada de todos los procesos de hemodonación y hemoterapia en Extremadura cuyo objetivo es conseguir el autoabastecimiento de sangre y hemoderivados en la comunidad autónoma gracias a las donaciones voluntarias y desinteresadas de los ciudadanos.
La labor del Banco de Sangre de Extremadura es necesaria para poder realizar cirugías y hemoterapias, entre otros procedimientos, debido a que la sangre es un tejido que no puede ser creado de forma sintética.

Actualmente, la forma en la que se comunica el BSE con los donantes habituales es de forma tradicional, concretamente, mediante carta postal o SMS en caso de necesidad de su grupo sanguíneo. También colaboran con las Hermandades de Sangre para promocionar las colectas en las diferentes localidades. Pero presentan un problema para atraer al público joven, ya que la mayoría de usuarios frecuentes son personas mayores de 50 años y a los mayores de 65 años no se les permite donar, con lo que se reducirá considerablemente el número de usuarios a medio plazo.

Por esta razón, hemos visto necesario crear un sistema software de soporte para el BSE y de promoción entre los voluntarios del BSE. Ello, facilitará la relación entre los voluntarios y el BSE, además se espera un aumento de voluntarios en el sector jóven. Este sistema mantendrá informado de forma pasiva al usuario de las colectas cercanas en todo momento y del estado de las reservas de sangre en el BSE a la vez que se informatizarán procesos internos del BSE.

Abstract

The Banco de Sangre de Extremadura is a health sub-institution of the Servicio Extremeño de Salud, which is in charge of all the processes of hemodonation and hemotherapy in the area of Extremadura, whose objective is to achieve self-supplies of blood related equipment through voluntary and altruistic donations from citizens.
The task of the aforementioned institution is necessary in order to be able to perform surgeries and hemotherapies, among other procedures, due to the fact that blood is a tissue that cannot be produced synthetically.

Nowadays, the way in which the BSE communicates with its usual donors is traditional: through postal letter and via SMS if they need your blood group. Furthermore, they also collaborate with blood brotherhoods to promote collecting campaigns in different locations. However, they usually face problems. When it comes to attract young audiences, since most of the frequent users are older than 50, and those who exceed 65 are not allowed to donate, which results in a decrease on the average number of users.

Due to this reason, it is clear why creating a software system to offer support to the BSE staff and promotion between the volunteers of BSE is needed. Actually, it will ease the relationship and communication between the volunteers and the BSE crew. Furthermore, an increase in the number of young volunteers is expected. This system will keep the user in touch with the organization about any nearby collections and provide information on the blood stockage in the BSE while the processes of the institution are computerized.

Índice general

Índice de tablas

Índice de figuras

Capítulo 1

Introducción

El Banco de Sangre de Extremadura es un centro sanitario constituido como Centro Comunitario de Transfusión Sanguínea que opera en la Comunidad Autónoma de Extremadura [1], es decir, es una Institución Sanitaria que depende del Servicio Extremeño de Salud. Se fundó en 2002 con la finalidad de abastecer a los hospitales de Extremadura de hemocomponentes[1] y hemoderivados[2]. Para obtener la **sangre**, es necesaria la participación de voluntarios que la donen de forma altruista en colectas realizadas por toda la región.

Para atraer a estos voluntarios, el Banco de Sangre de Extremadura publica los datos de las colectas (o donaciones) y el lugar en el que se llevarán a cabo. Esto lo realizan de distintas formas:

- Al **público general** mediante el uso de redes sociales como Facebook o Twitter y el *blog* de esta organización.

- A los **donantes habituales** mediante el envío de cartas postales y SMS si hay una carencia del grupo sanguíneo de estos donantes.

- En los **puntos de donación** mediante la colocación de carteles informativos.

[1]Hemocomponentes: fracción celular o acelular de la sangre obtenido de una unidad de sangre total mediante procesos físicos.

[2]Los *hemoderivados* que se extraen de la unidad de sangre total después de una serie de procesos físicos son las albúminas, globulinas y factores de coagulación

Desde el Banco de Sangre de Extremadura plantean mensualmente los puntos de donación donde se realizarán las colectas. Estos datos los plasman en una hoja de cálculo Excel y lo suben a las redes sociales y al blog mediante una captura de pantalla de esta hoja de cálculo.

Esta es una forma simple pero muy primitiva de realizar esta promoción y de exponer los datos al público general. De ahí, surge la necesidad de mejorar el sistema actual de promoción y difusión de los datos al público.

Este proyecto consiste en el desarrollo de una aplicación corporativa del Banco de Sangre de Extremadura y una aplicación móvil para el público en general, concretamente:

- La **aplicación móvil** dirigida al público general, que accede a los datos mediante una API REST.

- La **aplicación web** dirigida al Banco de Sangre de Extremadura (BSE), que añade, modifica y elimina los datos que serán almacenados en la base de datos y que se expondrán al público general mediante la API REST.

Estos datos están relacionados con las colectas y sus resultados, los lugares de colecta, niveles de sangre para cada grupo, noticias y alertas.

La aplicación móvil será de gran utilidad para los donantes y posibles donantes por las siguientes razones:

- Obtienen de forma constante información sobre el los niveles de sangre disponible en el BSE.

- No es necesario acceder constantemente a la aplicación para saber cuándo hay una colecta cercana, ya que ésta avisa con antelación.

- Informa al usuario sobre el proceso completo de donación, explicando los requisitos para poder donar, la donación en sí y las recomendaciones que deben seguirse después de donar.

Este proyecto, además es muy útil para el Banco de Sangre de Extremadura por las siguientes razones:

- La aplicación móvil está concebida para atraer a un público joven e itinerante, que es justo el sector de la población del que carece actualmente el BSE.
- Se reducen los costes de promoción, como por ejemplo los SMS y las cartas postales en este sector de la población.
- Muchas de las tareas que anteriormente estaban disgregadas, como son la creación de informes de existencias de hematíes y plaquetas, la publicación de la programación de las colectas y las noticias, ahora están aunadas en un único sistema.

Por último, este proyecto es de gran utilidad para la sociedad en general, por las siguientes razones:

- Aumenta el número de usuarios del Banco de Sangre de Extremadura.
- Aumenta los niveles de reservas, incluidos los de *plasma* y *plaquetas* que son los que tienen una vida útil más corta (tan solo cinco días).
- Se conciencia a los usuarios de la necesidad de la donación de sangre.
- Otras aplicaciones pueden utilizar los datos de la aplicación, lo que podría aumentar aún más el número de usuarios.

Capítulo 2

Objetivos

2.1. Objetivo principal

El objetivo principal de este Trabajo de Fin de Grado (TFG) es dar una solución real y sencilla a los problemas presentes en el ámbito de la promoción, sobre todo entre el público joven y la publicación de datos al público en general de las colectas del banco de sangre y de información del stock de hemocomponentes en tiempo real.

2.2. Objetivos secundarios

Teniendo en cuenta este objetivo principal, se han realizado un par de entrevistas con el director del Banco de Sangre de Extremadura para obtener una lista de requisitos con la que empezar a diseñar un primer prototipo. Los requisitos obtenidos son los siguientes:

1. Deberá existir una aplicación corporativa sencilla que facilite la introducción de los datos (de stock, colectas, noticias y emergencias) y otra aplicación móvil que consumirá estos datos.

2. Los datos generados son estructurados.

3. No se podrá recabar ningún dato del usuario. Todo dato recabado para los

distintos filtros (localidad y grupo sanguíneo) deberá permanecer en todo momento en el dispositivo del usuario.

4. Será necesario el uso de la ubicación para notificar a los usuarios las colectas programadas cercanas a su posición.

5. Las emergencias, serán derivadas a los puntos de extracción fijos. Las emergencias son eventos excepcionales que suceden de forma no planificada, como puede ser un accidente con un número elevado de heridos que descendería de forma notoria las reservas de hemocomponentes.

6. El objetivo es llegar a un público joven, ya que actualmente no es así, y lo más adecuado sería utilizar una aplicación móvil, en vez de medios tradicionales como puede ser la carta ordinaria, SMS o el acceso manual a sus redes sociales.

7. Los datos que se generen desde la aplicación corporativa, deberán ser públicos para ser reutilizados por otras aplicaciones, ya que podría aumentar considerablemente el número de usuarios del BSE.

8. El estado actual de las existencias de hemocomponentes deberá ser en tiempo real o actualizado cada poco tiempo, al menos más de una vez al día.

9. Será necesario añadir las colectas programadas en la aplicación corporativa.

10. En las colectas programadas, se deberá poder añadir con posterioridad al evento el número de bolsas conseguidas y el número de nuevos donantes.

11. Se deberá transformar el documento de texto con extensión .docx que contiene el informe de existencias en un PDF con los datos concretos.

12. Se podrá modificar el estado de la colecta (o evento de donación) entre activo y cancelado.

13. Se deberá diferenciar entre una ciudad y un punto de donación.

14. La aplicación móvil deberá hacer uso de los datos proporcionados por la aplicación corporativa del BSE.

15. El proyecto deberá ser lo más económico posible y desarrollado utilizando herramientas y *frameworks* de código abierto.

De estos requisitos obtenidos en la primera reunión hemos generado los siguientes objetivos derivados del primario:

- La creación de una aplicación que sea lo más liviana posible debido a las características de los ordenadores de los que disponen y lo más sencilla posible.

- La creación de una aplicación móvil lo más sencilla posible que utilice los datos generados por la aplicación corporativa.

- La creación de una interfaz sencilla para exponer los datos al público.

Capítulo 3

Estado de la cuestión

En este apartado se van a describir las posibles tecnologías aplicables a este proyecto. Se enumerarán las ventajas e inconvenientes de cada una de ellas y se seleccionarán aquellas más idóneas razonando la respuesta.

En primer lugar veremos las alternativas para el sistema de la organización, continuaremos con las posibles bases de datos que podrán ser utilizadas, la interfaz que expondrá los datos y a continuación, los posibles marcos de desarrollo de aplicaciónes para dispositivos móviles. Terminaremos revisando aplicaciones similares de otros bancos de sangre de España.

3.1. Aplicación corporativa

Para poder satisfacer uno de los objetivos más importantes, que es el de crear una aplicación liviana, hemos tenido en cuenta dos tipos de aplicaciones: las aplicaciones de escritorio por un lado y las aplicaciones web por otro. Ambas aplicaciones deberían tener acceso a la red (ya sea internet o intranet) para recuperar y enviar registros a la base de datos. Además, esa base de datos deberá ser centralizada debido a la concurrencia de usuarios.

3.1.1. Aplicaciones de escritorio

Normalmente utilizamos el término "software de escritorio" para referirnos a sistemas autónomos que pueden ser desplegados de forma independiente en un tipo específico de hardware. Un ejemplo puede ser la aplicación Microsoft Word™. Esta aplicación es descargada e instalada en nuestro equipo. Una vez instalada, se puede usar de forma autónoma. [2]

Ventajas

- Estas aplicaciones normalmente tienen más funcionalidades, ya que aprovechan todo lo que el sistema operativo le ofrece.
- Estas aplicaciones no dependen de otros sistemas y son independientes.
- Por norma general, no es necesaria su conexión a Internet para su funcionamiento.

Inconvenientes

- Es imposible proteger los algoritmo de la ingeniería inversa.
- La escalabilidad depende del hardware del cliente.
- El usuario se deberá hacer cargo de la instalación y actualizaciones del software.
- Es dependiente del sistema operativo.
- Debe ser actualizado en cada equipo de forma individual.

3.1.2. Aplicaciones web

Los sistemas software "basado en la web", al contrario que el software de escritorio, se usan a través del navegador web del usuario, utilizando un hardware remoto mantenido por los propietarios de este software. El ejemplo complementario

al anterior sería entrar en https://office.com y utilizar Microsoft Office Word de forma online sin necesidad de descargar la aplicación. [2]

Ventajas

- Por regla general, es imposible aplicar ingeniería inversa.
- La escalabilidad depende de la infraestructura en la que esta desplegado el sistema.
- No requiere ningún mantenimiento por parte del usuario.
- No son necesarios requisitos especiales para ejecutar el software además de un navegador web actualizado.
- No es necesaria su instalación.
- Es independiente del sistema operativo.

Inconvenientes

- Necesitan conexión a internet (o intranet en el caso de un software corporativo).
- Suelen poseer menos funcionalidades que una aplicación de escritorio.
- Dependen del servidor al que estén conectados para obtener acceso a esa aplicación. Si el servidor deja de funcionar o se queda sin conexión, no se podrá utilizar esa aplicación.

3.1.3. Tipo de aplicación escogida

Para llevar a cabo este proyecto, se ha decidido **escoger una aplicación de ordenador basada en web** por las siguientes razones:

1. Uno de los requisitos es que la aplicación debe ser muy ligera y ejecutable en un entorno con pocos recursos hardware.

2. No es necesario que el usuario final tenga que actualizar la aplicación para las mejoras oportunas o la solución de fallos.

3. Si el usuario cambia de equipo, no tendrá que reinstalar el software.

3.1.4. Frameworks para desarrollar una aplicación web

Para desarrollar una aplicación web, hay muchas tecnologías posibles para ello, aunque se pueden agrupar por su funcionamiento en **aplicaciones del lado del cliente** y en **aplicaciones del lado del servidor**. Se van a mostrar algunas de las más importantes y se seleccionará una de entre todas ellas.

Pila MEAN

La pila MEAN o *MongoDB*, *Express*, *AngularJS* y *Node.js* es un conjunto de herramientas de desarrollo "full-stack" en JavaScript para crear aplicaciones web. La aplicación se ejecutará en el lado del cliente, realizando las operaciones desde éste y tan solo solicitando los datos al servidor cuando sea necesario.

- **MongoDB**: es un sistema de gestión de bases de datos multiplataforma orientado a documentos y de esquema libre.

- **Express**: es un *framework* del lado del servidor para desarrollar aplicaciones web con Node.js.

- **AngularJS**: es un *framework* de JavaScript de código abierto que permite el desarrollo de aplicaciones web en el lado del cliente y utiliza un patrón Modelo-Vista-Controlador.

- **Node.js**: es un intérprete de JavaScript del lado del servidor. Su principal objetivo es que el programador sea capaz de desarrollar una aplicación con alta escalabilidad en una única máquina. [3]

Ventajas

- El uso de JavaScript en todo su desarrollo como único lenguaje.
- Gracias a AngularJS, es muy sencillo crear aplicaciones "Single Page Application" (SPA).

Desventajas

- Curva de aprendizaje lenta
- El Sistema Gestor de Base de Datos (SGBD) es NoSQL, por lo que desarrollos con un SGBD relacional quedan fuera de este ámbito.

Meteor

Es un *framework* OpenSource para desarrollar aplicaciones web y aplicaciones móviles en JavaScript escrito en Node.js. Se integra con las siguiente herramientas:

- **React**: es una librería de JavaScript para crear interfaces de usuario. [4]
- **Vue**: es una librería de JavaScript para crear interfaces de usuario SPA. [5]
- **Angular.js**
- **MongoDB**
- **GraphQL**: es un lenguaje de consulta API, que ejecuta las consultas del lado del servidor usando un sistema de tipos que es definido por el desarrollador. No está vinculado a ninguna base de datos, sino que utiliza datos o bases de datos ya existentes. [6]
- **npm**: es un gestor de paquetes para JavaScript. [7]
- **Apache Cordova**: es un "framework" móvil de código abierto. Permite utilizar las tecnologías estándar web como HTML5, CSS3 y JavaScript para desarrollo multiplataforma, evitando el lenguaje de desarrollo nativo para cada plataformas móviles. [8]

Ventajas

- Está diseñado para hacer la pila MEAN más sencilla.
- Permite mayor flexibilidad que la pila MEAN.

Desventajas

- Se intercambian los aspectos de rapidez y facilidad de uso por el control al utilizar Meteor en vez de MEAN.
- Es nuevo y no tiene una gran comunidad. [9]

Django

Es un *framework* web de alto nivel que permite el desarrollo de aplicaciones web de forma rápida, segura y mantenible. Está basado en Python y que tiene como principio *Don't Repeat Yourself* (DRY). [10]

Ventajas

- Se basa en Python, que es uno de los lenguajes más usados y con mayor comunidad y con un gran numero de *plug-ins*.
- Es muy versátil y con muchas características.

Desventajas

- Es monolítico.
- Requiere un gran conocimiento de Python. [9]

Ruby on Rails

Es un *framework* de desarrollo de aplicaciones web escrito en el lenguaje de programación Ruby. Esta diseñado para hacer las aplicaciones web más fáciles al suponer lo necesario para que todos los desarrolladores comiencen. [11]

Ventajas

- Incluye su propio servidor y una base de datos SQLite, que es fácilmente intercambiable.
- Es excelente para crear aplicaciones rápidamente.

Desventajas

- Tiene una gran pérdida de control en favor de esa rapidez y sencillez de desarrollo.
- Es sencillo escribir código sin saber qué significado tienen esas líneas. [9]

Java + Spring

Spring es un *framework* "full-stack" escrito en Java. Es una alternativa a Java Enterprise Edition (J2EE). Utiliza Programación Orientada a Aspectos (PAO, AOP en inglés). Además promueve la pérdida de acoplamiento haciendo uso de la inyección de dependencias. [12]

Ventajas

- La inyección de dependencias en Spring permite a los desarrolladores separar de una forma limpia sus clases sin perder el acceso a las dependencias.
- No requiere programadores para implementar ninguna interfaz, lo que hace el código mucho más mantenible.

Desventajas

- Tiene más de 2.500 clases y muchas otras herramientas que incrementa la curva de aprendizaje.[9]

3.1.5. Tecnologías seleccionadas

Debido a los requisitos obtenidos en el apartado 2.2, y siendo uno de los más importantes la creación de una aplicación web ligera, descartamos la opción del una aplicación web del lado del cliente (*client-side*) ya que, al depender del hardware del cliente, ejecutar este tipo de aplicaciones puede ralentizar el sistema. Por lo tanto, descartamos la pila MEAN y Meteor.

Además, existe el requisito de la obtención de los puntos de colecta y su ciudad asociada, es decir, es necesario un análisis de datos. En Python existen opciones sencillas para el análisis de datos como puede ser "Pandas".

Existe otro requisito con respecto a la generación de documentos PDF en base a los datos de los stocks. En Python existen varias opciones, entre ellas "ReportLab" y en Java, "Pdf in IText".

Otro requisito es exponer los datos generados al público. En Django, existe "Django REST Framework". En Spring viene incluido dentro de Spring Boot.

Teniendo todo esto en cuenta, se ha decidido optar por Django por las siguientes razones:

- Python es un lenguaje más sencillo y más inteligible.
- Django tiene un patrón de diseño MVT (Modelo-Vista-Plantilla) que simplifica mucho la creación de un proyecto de este tipo.
- Django implementa el principio DRY.
- La serialización de objetos es muy sencilla.
- Provee una sencilla y completa librería para crear una API REST.
- Hay un gran número de librerías en Python para todos los objetivos a cumplir.
- Django está escrito desde cero, mientras que Spring se ha basado en tecnologías anteriores.
- Sencillo mapeo objeto-relacional.

- La documentación de Django y de las librerías de Python, por norma general, tienen un alto grado de detalle.

3.2. Base de datos

En este proyecto, los datos que vamos a insertar en una base de datos, son totalmente estructurados, por lo que utilizaremos una base de datos relacional. Un ejemplo podría ser el de las *Ciudades*, ya que todas deben tener su identificador, su nombre, el nombre de la provincia a la que pertenece, la latitud y longitud y una referencia al área de salud al que pertenece.

En este apartado, compararemos las distintas bases de datos disponibles para Django y elegiremos la más idónea.

Para garantizarnos una mayor fiabilidad, utilizaremos las bases de datos soportadas de forma oficial, que son: **PostgreSQL, MySQL, MariaDB, SQLite y Oracle Database Server**.

Uno de los requisitos es la economicidad del proyecto, por lo que las licencias como las de Oracle Database Server no pueden ser asumidas. Además, si hay que elegir entre MySQL y MariaDB, escogeremos ésta última, ya que es *open-source* y pertenece a la comunidad (MySQL pertenece a Oracle, quien puede cambiar su licencia por una privativa) y además, MariaDB tiene unas ligeras mejoras de rendimiento con respecto a MySQL.

SQLite, no puede ser una buena opción para la etapa de producción de este proyecto, ya que es necesario un mayor nivel de seguridad y, además, en mejoras futuras se deberán de incluir copias de seguridad regulares de los datos.

Por lo tanto, tan solo tenemos dos opciones viables, **PostgreSQL** y **MariaDB**. Entre estas dos se ha elegido MariaDB, ya que:

- Tiene un mayor rendimiento con bases de datos pequeñas pero es menor con bases de datos medianas y grandes.

- Uno de los usos recomendados son el de complemento a un servicio web, como

por ejemplo sucede con WordPress.

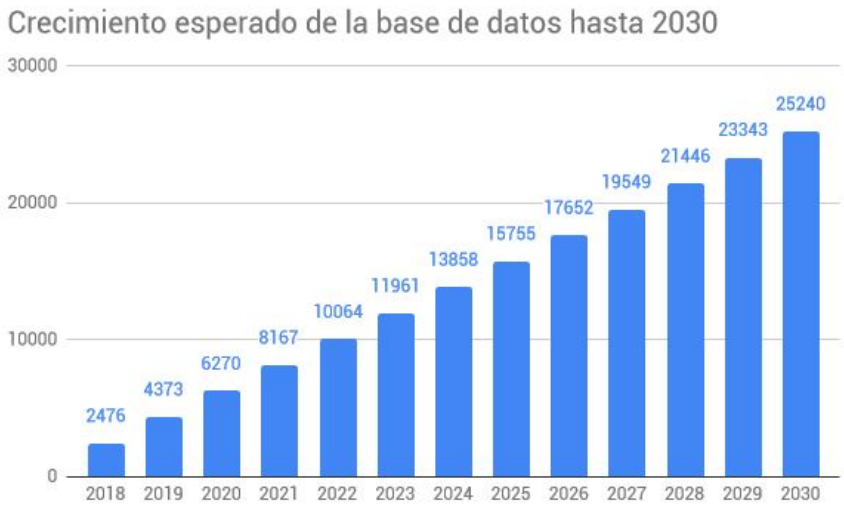


Figura 3.1: Crecimiento esperado de la base de datos desde la actualidad hasta 2030

El crecimiento esperado de los datos de esta base de datos es el descrito en la figura 3.1. Teniendo en cuenta los requisitos más específicos del cliente como es el contenido de cada tabla, podremos tener una base de datos de aproximadamente unos 15,27 MB en 2030, teniendo en cuenta exclusivamente los datos. Para eso nos hemos basado en los datos presentes en la tabla 3.1.

Año	Area de salud	Ciudades	Puntos de donación	Eventos fijos	Eventos de donación	Noticias	Emergencias	Stock
2018	8	228	333	10	1095	60	12	730
2019	8	228	333	10	2190	120	24	1460
2020	8	228	333	10	3285	180	36	2190
2021	8	228	333	10	4380	240	48	2920
2022	8	228	333	10	5475	300	60	3650
2023	8	228	333	10	6570	360	72	4380
2024	8	228	333	10	7665	420	84	5110
2025	8	228	333	10	8760	480	96	5840
2026	8	228	333	10	9855	540	108	6570
2027	8	228	333	10	10950	600	120	7300
2028	8	228	333	10	12045	660	132	8030
2029	8	228	333	10	13140	720	144	8760
2030	8	228	333	10	14235	780	156	9490
Tamaño por fila (B)	1004*	1045*	2026*	69	26	16495**	490	110
Tamaño por tabla 2030 (MB)	0,008032	0,23826	0,674658	0,00069	0,37011	12,8661	0,07644	1,0439

(*): Los campos LONGTEXT pueden llegar a ser de 4GB pero se ha estimado que en el peor de los casos serán usados como máximo 1KB.

(**): Los campos LONGTEXT pueden llegar a ser de 4GB pero se ha estimado que en el peor de los casos serán usados como máximo 16KB.

Tabla 3.1: Filas de las tablas de base de datos por año y su tamaño total en 2030.

De todos estos datos estimados, podemos asumir que esta base de datos no es de gran envergadura, por lo que es preferible utilizar MariaDB en vez de PostgreSQL.

Si esta estimación quedase corta, ya sea por un mayor número de tablas en el futuro, un mayor número de filas por año, o un uso más complejo de los datos almacenados o una combinación de ellas, y se considerase cambiar de sistema gestor de base de datos a PostgreSQL, no debería haber ningún inconveniente ya que existen herramientas como *pg_chamaleon* o *pgloader* para poder hacerlo. [13]

3.3. Exposición de los datos

La **exposición de datos se hará en este proyecto mediante el uso de una API REST**. Se ha tomado esta decisión teniendo en cuenta los siguientes datos:

- Django (y Python) nos provee herramientas para desarrollar una API REST de forma sencilla como son Django Rest Framework o Flask.
- En la "Guía de arquitectura de apps" de Android Jetpack (una de las posibles alternativas móviles), nos recomiendan obtener los datos desde una API REST haciendo uso de la librería Retrofit. [14]

La diferencia entre Django Rest Framework y Flask es la funcionalidad que ofrecen. Django Rest Framework nos ofrece, de forma general, opciones de seguridad y de personalización de la API REST, mientras que Flask nos la ofrece de una forma mucho más sencilla y simple, pero mucho menos personalizable.

Por lo tanto, se utilizará **Retrofit** para la aplicación Android y **Django Rest Framework** para la aplicación web desarrollada con Django.

3.4. Aplicación móvil

En este apartado vamos a comparar las tecnologías móviles más importantes. En este caso, hablaremos de desarrollo en las plataformas Android e iOS.

3.4.1. Android

Android es una plataforma de computación popular basado en el sistema operativo Linux. La primera versión comercial de Android fue lanzada al mercado en 2008 en forma de plataforma para teléfonos móviles, siendo BlackBerry el terminal más popular entre la gente de negocios y cuando el iPhone estaba dándose a conocer de forma notoria en todos los sectores. [15]

Cuota de mercado

En sus diez años de historia, Android se ha convertido en el sistema operativo móvil más popular. Su crecimiento ha sido imparable, pasando desde un 3.9 % de cuota de mercado en 2009, a un 88.0 % en el segundo trimestre de 2018. [16][17]

En la figura 3.2 podemos observar una gráfica con su creciente evolución desde 2009 hasta mediados de 2016.

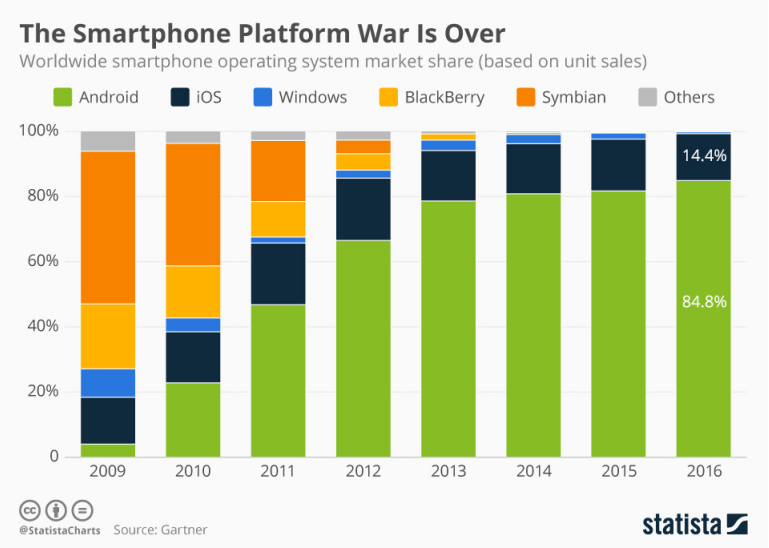


Figura 3.2: Cuota de mercado de las plataformas de smartphone 2009-2016
Fuente: https://www.statista.com/chart/4112/smartphone-platform-market-share/

Coste inicial de desarrollo

Tanto Android Open Source Project como el Integrated Development Environment (IDE) Android Studio (como parte de JetBrains Community Edition) tienen una

licencia Apache 2.0. Sin embargo el SDK que es usado en este IDE para desarrollar aplicaciones es más restrictivo que las anteriores licencias. Todos ellos son de uso gratuito, cumpliendo todas las condiciones de las correspondientes licencias. [18][19][20]

Una vez terminada la aplicación desarrollada, lo más recomendable es añadirla a la tienda de aplicaciones mayoritaria (sin incluir China, debido al "Gran Firewall de China")[21], que es Google Play Store, debido a que es instalada por defecto en los terminales Android. El precio de acceso de Google Play Console (que incluye la opción de añadir aplicaciones a la Play Store) es de un solo pago de 25 USD. [22]

Lenguaje de programación

Java es el lenguaje de programación oficial de Android, aunque Kotlin, ha sido introducido recientemente como el segundo lenguaje oficial. Android también ofrece soporte a C/C++ mediante el uso del Android Native Development Kit (NDK). [23]

Fragmentación

La fragmentación es un problema evidente de Android y desde Google están intentando luchar contra este fenómeno desde la versión Oreo (API 26) con un proyecto llamado Treble. [24]

Pero la realidad es que sigue siendo un problema en la actualidad como se puede observar en la tabla 3.2.

Teniendo en cuenta estos datos, la mayoría de usuarios tiene una versión superior a la 5.0 pero inferior a la 8.1 (77.9 %).

Dos meses antes de esta estadística (06/08/2018) se lanzó la última versión, Android Pie 9.0 y, tras más de dos meses, aún no ha conseguido ni una sola décima porcentual, que es el mínimo para poder aparecer en la tabla 3.2. [25]

Versión	Nombre	API	Distribución
2.3.3-2.3.7	Gingerbread	10	0.2 %
4.0.3-4.0.4	Ice Cream Sandwitch	15	0.3 %
4.1.x	Jelly Bean	16	1.1 %
4.2.x		17	1.5 %
4.3		18	0.4 %
4.4	KitKat	19	7.6 %
5.0	Lollipop	21	3.5 %
5.1		22	14.4 %
6.0	Marshmallow	23	21.3 %
7.0	Nougat	24	18.1 %
7.1		25	10.1 %
8.0	Oreo	26	14.0 %
8.1		27	7.5 %

Fuente: Android Developers. [26]

Tabla 3.2: Distribución de versiones de Android a 26/10/2018

3.4.2. iOS

Anteriormente denominado iPhone OS, es el sistema operativo presente en los iPhone desde su primera versión lanzada en junio de 2007. Salió al mercado mientras Windows Mobile, Palm OS, Symbian y Blackberry OS lideraban el mercado móvil. [27]

Cuota de mercado

Como vimos anteriormente en la Figura 3.2, la cuota de mercado de iOS ha estado entorno al 15 % de forma más o menos estable, estando actualmente en un 11.9 % como

podemos observar en la tabla 3.3.

Sistema operativo	Unidades en 2Q18 (en miles)	Cuota de mercado en 2Q18 (%)	Unidades en 2Q17 (en miles)	Cuota de mercado en 2Q17
Android	329.503,4	88,0	321.848,2	87,8
iOS	44.715,1	11,9	44.314,8	12,1
Otro sistema operativo	112,1	0,0	433,1	0,1
Total	**374,330.6**	**100.0**	**366,596.1**	**100.0**

Fuente: Gartner [17]

Tabla 3.3: Cuota de mercado de los sistemas operativos móviles del segundo trimestre de 2018 en comparación con el mismo trimestre del año anterior.

Coste inicial de desarrollo

A diferencia de Android, para desarrollar aplicaciones en iOS, es necesario utilizar los productos de Apple, incluyendo un dispositivo Mac, ya que el iPhone se puede simular. Esto es así debido a que el IDE de iOS, Xcode, es exclusivo de plataformas con el sistema operativo Mac OS. [28]

Además del coste de estos equipos, al finalizar el desarrollo de la aplicación, es necesario subir esta aplicación a la única tienda de aplicaciones permitida en iOS, la App Store. Esta tienda tiene más de mil millones de usuarios (2016). [29]

Una vez terminada la aplicación y si quiere publicarse en esta tienda, se deberá hacer un pago anual de 99 USD en el caso de licencias personales, en el caso de licencias empresariales pasa a ser un pago anual de 299 USD. [30]

Lenguaje de programación

El lenguaje de programación usado para desarrollas las aplicaciones en iOS es Swift. Swift es un lenguaje de programación OpenSource potente e intuitivo para

desarrollar en macOS, iOS, watchOS y tvOS. Swift también es interoperable con Objective-C. [31]

Fragmentación

De acuerdo a los datos oficiales ofrecidos el 3 de diciembre de 2018, el 70 % de todos los dispositivos utilizan iOS 12. Este dato es significativo teniendo en cuenta que esta versión fue lanzada el 17 de septiembre de 2018 (menos de tres meses desde su lanzamiento). La diferencia con respecto a Android es notable ya que, como se comentó anteriormente, el porcentaje de uso de la nueva versión no alcanza ni el 0,1 %. [32][33]

3.4.3. Plataforma móvil seleccionada

Tras revisar todas las ventajas e inconvenientes de Android e iOS finalmente **la aplicación que se desarrollará será una aplicación Android** por las siguientes razones:

1. Uno de los requisitos es llegar al mayor número de usuarios posibles. Por ello, se selecciona Android, que es el que mayor cuota de mercado posee actualmente (88.0 %).

2. Además, tiene un coste inicial de desarrollo muy bajo ya que no es necesario tener un dispositivo específico para realizar el desarrollo, al contrario de iOS, que es necesario tener un dispositivo Mac con Xcode.

3. Los lenguajes de programación de Android son conocidos y tienen una mayor comunidad, sobre todo Java.

4. La publicación del producto en la tienda de aplicaciones no tiene una cuota anual por mantener la aplicación en dicha tienda, tan solo una cuota. Además con ese único pago se podrán desarrollar tantas aplicaciones como se desee.

5. Es cierto que Android tiene un gran problema con la fragmentación pero haciendo uso de determinados recursos, la aplicación funcionará en todos los dispositivos independientemente de la versión del sistema operativo.

Además, viendo los datos de la tabla 3.2, se ha decidido tener la versión objetivo 27 (Oreo 8.1) del SDK. Sin embargo la mínima se ha optado por utilizar la 21 (Lollipop 5.0) del SDK para abarcar un mayor número de usuarios posibles sin comprometer las funcionalidades de la aplicación, concretamente un 96,5 % del total, como podemos ver en la tabla 3.2.

3.4.4. Frameworks Android seleccionados

Room

Room nos provee una capa de abstracción sobre SQLite para permitir el acceso a la base de datos con fluidez mientras aprovecha todas las características de SQLite. [34]

Retrofit

Es una librería que nos provee una cliente HTTP con tipado seguro para Android y Java. [35]

Ambas librerías también son recomendadas desde la *Guía de arquitectura de apps* de Android Jetpack. [14]

3.5. Aplicaciones Android similares

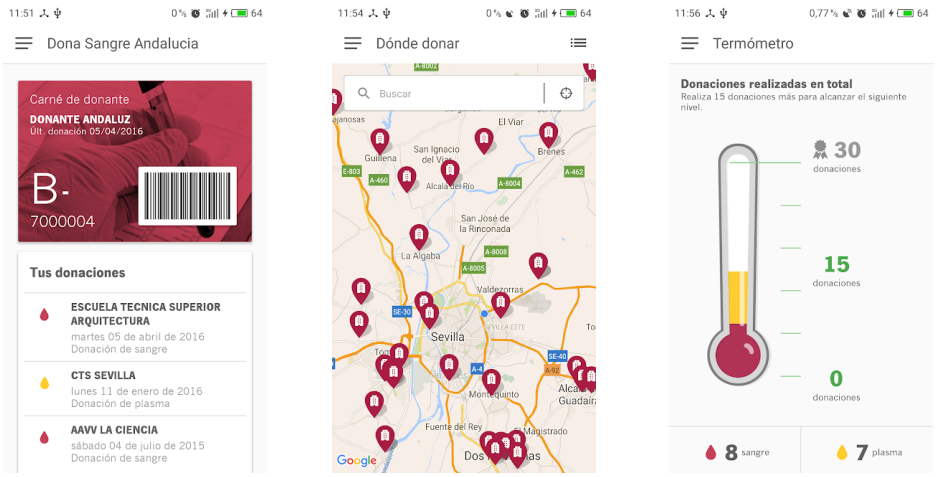
Debido a que el Banco de Sangre de Extremadura nunca ha tenido un sistema como el que se quiere llevar a cabo y, además, el ámbito de los bancos de sangre son autonómicos, no existen competidores. Lo que si existen son aplicaciones móviles similares.

3.5.1. Dona Sangre Andalucía

Esta aplicación Android es la única de este tipo presente en el ámbito nacional. Tiene las siguientes características:

- Tiene un Área del donante, donde se podrán ver un registro de las donaciones que ha realizado esa persona.
- Un "*termómetro*" en el que se contabiliza el número de donaciones de sangre y plasma para alcanzar niveles.
- Un indicador con las donaciones de sangre y plasma realizadas durante el año actual del usuario.
- Es necesario el registro.
- Existe un mapa con las ubicaciones de los centros para donar.
- Incluye información útil sobre la donación.
- Incluye un conjunto de preferencias del donante.

En la figura 3.3 podremos observar algunas de las capturas de Dona Sangre Andalucía.



(a) Carné de donante (b) Mapa de puntos de donación (c) Termómetro de logros

Figura 3.3: Capturas de pantalla de la aplicación "Dona Sangre Andalucía"
Fuente: https://play.google.com/store/apps/details?id=es.ja.csalud.sas.msspa.appdonasangre

3.5.2. Comparativa

En este apartado se compararán las funcionalidades de la aplicación de **Dona Sangre Andalucía** con la aplicación Android propuesta.

Funcionalidades de Dona Sangre Andalucía

- Presenta el carné de donante (si te has identificado como donante de sangre en Andalucía).
- Visualiza el histórico de donaciones.
- Recibe alertas en función de la localización si tu grupo sanguíneo es necesario.
- Muestra tu compromiso con la donación en un "termómetro solidario", indicando el número de donaciones y el reconocimiento asignado.
- Comparte con familia y amigos tu acción solidaria al donar sangre o plasma.
- Muestra un mapa con todos los centros donde se realizan colectas.
- Muestra información útil sobre las donaciones.

Funcionalidades de la aplicación Android propuesta

- Visualiza las reservas actuales de sangre.
- Muestra las próximas colectas en función de la ubicación o todas si no se ha obtenido esta ubicación.
- Muestra las noticias publicadas por el BSE.
- Muestra información útil sobre las donaciones.
- Muestra un mapa con los centros donde se realizan colectas de forma eventual y centros donde se realizan de forma continua.
- Tiene acceso directo a las redes sociales.

Como vemos, ambas tienen funcionalidades similares, como por ejemplo: muestra un mapa con los lugares de donación, avisa sobre las necesidades de tu grupo sanguíneo teniendo en cuenta la ubicación y muestran información útil sobre las donaciones, pero hay grandes diferencias.

La aplicación **Dona Sangre Andalucía** tiene toda su funcionalidad si se introducen los datos facilitados por el Banco de Sangre de Andalucía una vez realizada la primera donación, en caso contrario solo puede acceder al mapa con los centros de colecta (eventuales y fijos). Además, no muestra información sobre el estado actual de las reservas de sangre.

La aplicación propuesta sin embargo no tiene la mayoría de las funcionalidades que tiene Dona Sangre Andalucía, por uno de los requisitos imprescindibles impuestos por el cliente que es la recopilación de datos personales por parte de la aplicación ni el uso de los mismos.

Capítulo 4

Metodología

4.1. Metodología escogida

En este caso hemos seguido una metodología ágil denominada Kanban basada en el desarrollo ágil de software de forma iterativa e incremental.

El término Kanban tiene su origen en las palabras japonesas “kan” que significa visible o visual y “ban” que significa tarjeta. Esta metodología se basa en estos términos, ya que consiste en tener un tablero con tarjetas o pequeños papeles autoadhesivos (ban) y nos permite llevar a cabo las tareas de una forma más flexible y sencilla, ya que de un solo vistazo se pueden observar todas las tareas, su estado y su prioridad (kan), además de otras características como son las siguientes:

- **Calidad garantizada**. La etiqueta no podrá llegar a la fase de “hecho” si no funciona correctamente. Para poder dar por terminada la tarea deberá pasar por todas las fases (pendiente, en preparación, en desarrollo, en pruebas y hecho) de forma satisfactoria.

- **Flexibilidad**. La siguiente tarea se decide del *backlog* (o tareas pendientes acumuladas), priorizando tareas dependiendo de las necesidades.

- **Reducción del desperdicio**. Esta metodología se basa en hacer lo justo y necesario, pero de forma correcta, eliminando todo aquello superficial o

secundario (principio YAGNI). [36]

Además, tiene los siguientes beneficios:

- Al reducir el número de tareas que se realizan de forma simultánea, se reduce el tiempo dedicado a completar cada una de ellas.
- Las tareas que están llevando más tiempo del esperado, bloquean al resto por lo que si, por ejemplo, la columna de *pruebas* está completa y no es posible pasar ninguna de estas a la columna *hecho* mientras que hay tareas de la columna de *desarrollo* que están terminadas y pueden pasar a la de *pruebas*, es evidente que hay que reorganizar el equipo para avanzar rápidamente con las pruebas.
- Se puede calcular el tiempo que ha tardado en ser desarrollada cada tarea como la diferencia entre el tiempo en ser colocado en la columna de *pendiente* y la columna de *hecho* (en nuestro caso).[37]

Las columnas elegidas en el tablero Kanban tampoco son fijas, pero en este caso se han utilizado las siguientes por decisión propia:

- **Pendiente**. Tareas que se pueden comenzar de forma inmediata.
- **En preparación**. Tareas que necesitan de una documentación previa antes de poder ser ejecutadas.
- **En desarrollo**. Tareas que han comenzado ha realizarse.
- **En pruebas**. Una vez que la tarea ha terminado su desarrollo de forma completa, se traslada a esta fase donde comenzarán las pruebas.
- **Hecho**. La tarea ha pasado por todas las anteriores fases de forma satisfactoria por lo que no debería volver atrás. La tarea ha finalizado.

Hay que tener en cuenta que si una tarea no está preparada para ejecutarse en la fase en la que están, deberán ser devueltas a la etapa anterior. [38]

En la figura 4.1 podemos observar los cinco estados de las tareas. En este caso se ha elegido como número máximo de cinco tareas en activo (la suma de los estados pendiente, desarrollo y prueba).

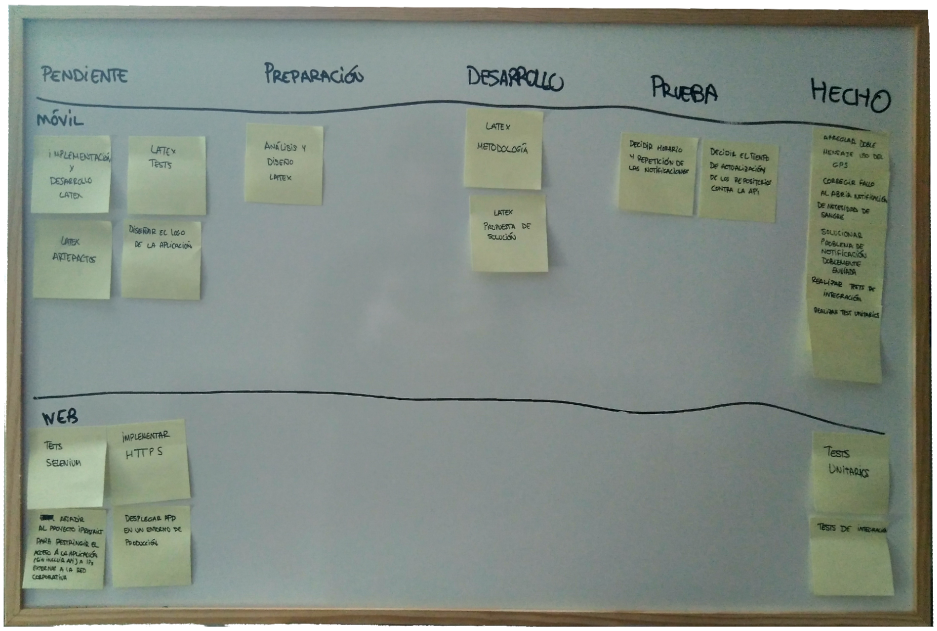


Figura 4.1: Tablero Kanban utilizado

En este caso, se ha utilizado un tablero físico, concretamente el de la figura 4.1 en vez de uno digital como por ejemplo puede ser Trello. Esta decisión se ha tomado por comodidad y simplicidad, ya que en la pared están todas las tareas sin necesidad de acceder a una plataforma digital.

4.2. Planificación

El proyecto tenía un tiempo estimado de desarrollo de unos **nueve meses**. En la figura 4.2 podemos observar el diagrama de Gantt inicial realizado durante la etapa de análisis.

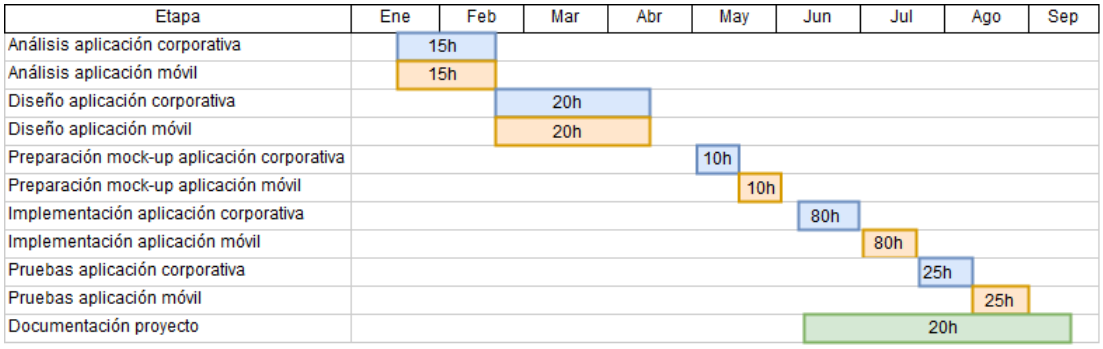


Figura 4.2: Diagrama de Gantt inicial

Sin embargo, debido a diversos contratiempos no contemplados y a circunstancias personales, entre otros, el proyecto se ha alargado más de lo inicialmente previsto. Por lo que, el diagrama de Gantt de la figura 4.3 reflejaría de forma más precisa los tiempos dedicados a cada fase.

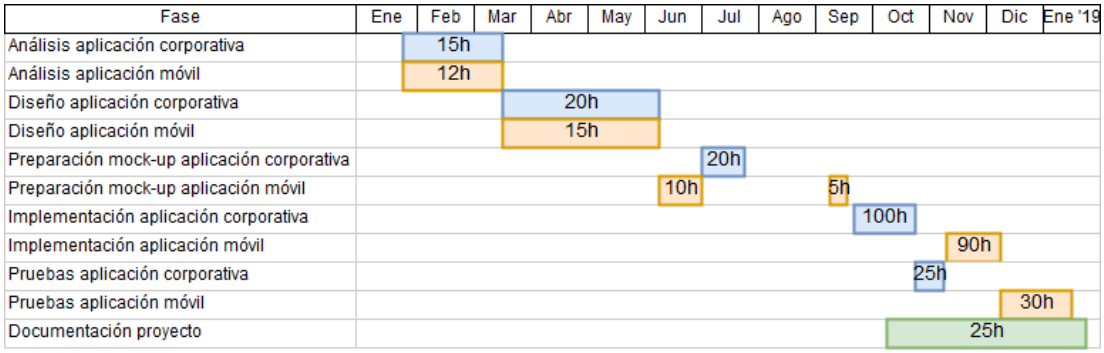


Figura 4.3: Diagrama de Gantt real

Esto tan solo representa una primera fase del proyecto (que es la que se expone en este Trabajo de Fin de Grado). De este proyecto esperamos que evolucione incrementando el número de iteraciones teniendo en cuenta los trabajos futuros.

Como podemos observar en las figuras 4.2 y 4.3 las barras indican el comienzo y fin de una fase y dentro de éstas el tiempo en horas dedicado a cada una de las fases.

4.3. Reuniones

Para poder llevar a cabo este proyecto se han necesitado una serie de reuniones para, por un lado, la obtención necesaria de requisitos y por otro lado, la revisión de

los requisitos que se han llevado a cabo y han sido implementados correctamente.

4.3.1. Toma de contacto

Antes de tener una reunión con el cliente, que en nuestro caso es el director del Banco de Sangre de Extremadura, José María Brull, se preparó un documento con preguntas cortas sobre la organización, sobre los métodos de publicidad y comunicación y el sistema que tienen actualmente.

Tras la revisión de las respuestas dadas por el cliente, se decidió crear una aplicación diferente a la que ellos ya tenían. Esta decisión fue tomada porque el sistema interno de control de la sangre era robusto y seguro, además de que el cliente tenía un contrato de mantenimiento durante bastantes años, por lo que se descartó la opción de crear este sistema.

Pero hay una parte que no cubría el sistema, que era el de la publicidad y comunicación que tiene el Banco de Sangre de Extremadura con los donantes o posibles donantes. Esta primera toma de contacto nos dio pie a crear un sistema para cubrir los apartados de publicidad y comunicación.

4.3.2. Primera reunión

La primera reunión se llevó a cabo el día 22 de febrero de 2018 en la Escuela Politécnica. En ella se expuso la idea de la creación de este sistema de promoción y, al ser aceptada por el cliente, se comenzó con la obtención de los requisitos.

En esta reunión se obtuvieron los requisitos genéricos y posteriormente, mediante envíos de correos electrónicos, se fueron detallando más a fondo.

Los requisitos más genéricos fueron los siguientes:

- Crear una aplicación que pudiera ser ejecutada en los ordenadores de la organización de uso interno.
- Creación de una aplicación móvil que mostrase a los usuarios los lugares de colecta y los niveles de sangre del BSE.

- Ninguna de las aplicaciones podría tener los datos de ningún usuario, ni podría recabarlos.
- Se podrá utilizar técnicas de geolocalización para avisar a los usuarios sobre las colectas. En este caso se utilizarán datos como pueden ser: GPS, dirección IP, datos provenientes del acelerómetro o elementos cercanos (puntos de acceso Wi-Fi, antenas de servicio de telefonía móvil o dispositivos con el Bluetooth activado)
- La aplicación móvil utilizará los datos generados en la aplicación de la organización.

En los posteriores correos se definieron aspectos más específicos de la lógica de negocio, concretamente:

- La forma de almacenar las colectas previstas para el próximo mes.
- La forma de almacenar el stock de hemocomponentes.

4.3.3. Segunda reunión

El trabajo realizado hasta esta segunda reunión fue el siguiente:

- Análisis de los requisitos para la aplicación corporativa.
- Diseño de la aplicación corporativa.
- Creación de los mock-ups de la aplicación corporativa.
- Análisis de los requisitos para la aplicación móvil.
- Diseño de la aplicación móvil.
- Creación de los mock-ups de la aplicación móvil.
- Creación de un vídeo mostrando el funcionamiento y aspecto de la aplicación móvil y otro de la aplicación corporativa.

En esta segunda reunión tan solo se modifican pequeños detalles como son:

- Modificación de la distancia y dimensiones de algunos botones de la aplicación web.
- Modificación del modelo de datos incluyendo área de salud para los eventos en vez de provincia.
- Los colores de las barras correspondientes a los grupos sanguíneos seguirán el estándar internacional. En los marcadores de la aplicación Android también seguirán dichos estándares.

Una vez que tuvimos el visto bueno por parte del director del BSE, se comenzó con el desarrollo de las aplicaciones. En este caso, también se han intercambiado una serie de correos electrónicos con detalles concretos como son:

- Listado completo de puntos de colecta.
- Límites inferior y superior de stock para cada uno de los grupos sanguíneos.
- Creación de grupos y permisos asignados a cada uno de ellos.

Además, se concretaron algunos aspectos relacionados con la interfaz de usuario, como es la información mostrada y la forma en la que se muestra.

Capítulo 5

Propuesta de solución

En este aparatado vamos a detallar la solución propuesta para alcanzar los objetivos del capítulo 2.

5.1. Análisis y diseño

Primero vamos a detallar los requisitos, se hará un análisis de los casos de uso y de las estructuras primarias de datos. Posteriormente pasaremos al diseño de los datos y al diseño arquitectónico.

5.1.1. Análisis de requisitos

En este apartado detallaremos a fondo los requisitos funcionales, no funcionales, de información y restricciones.

Requisitos funcionales

1. La aplicación corporativa deberá tener una página de acceso.

2. La aplicación corporativa deberá tener una página de inicio donde se mostrará un resumen del estado actual del sistema.

3. La aplicación corporativa deberá almacenar los datos del stock, pudiendo añadir, modificar y revisar estos datos.

4. La aplicación corporativa permitirá exportar los datos de stock de forma individual en formato PDF con un formato predeterminado.

5. La aplicación corporativa deberá almacenar los datos de las colectas programadas, pudiendo añadir, modificar, eliminar, cancelar, recuperar estas colectas. Además, deberá permitir añadir, ver y modificar el resultado de éstas.

6. La aplicación corporativa permitirá actualizar los puntos de colecta utilizando un fichero con extensión **xlsx**.

7. La aplicación corporativa deberá almacenar los datos de las emergencias, pudiendo leerlas y añadirlas.

8. La aplicación corporativa deberá almacenar los datos de las noticias, pudiendo publicarlas, eliminarlas, guardarlas como borrador, modificar borrador, eliminar borrador y publicar borrador.

9. La aplicación corporativa permitirá al usuario modificar su contraseña.

10. Los datos generados por la aplicación corporativa, serán expuestos utilizando una interfaz.

11. La aplicación móvil obtendrá los datos necesarios de la interfaz donde están expuestos.

12. La aplicación móvil deberá mostrar los datos de stock en tiempo real, o lo más real posible.

13. La aplicación móvil deberá mostrar las próximas colectas.

14. La aplicación móvil deberá mostrar información sobre las donaciones al usuario.

15. La aplicación móvil deberá mostrar un listado de noticias recientes.

16. La aplicación móvil deberá mostrar un enlace a las redes sociales del BSE.

17. La aplicación móvil deberá mostrar todos los puntos de extracción activos y todos aquellos que sean fijos.

18. La aplicación móvil deberá enviar notificaciones al usuario sobre colectas cercanas (en tiempo y ubicación), la necesidad de sangre de uno o varios grupos sanguíneos, emergencias y noticias.

19. La aplicación móvil podrá filtrar las colectas en base a la ubicación del usuario. Ésta podrá ser automática (por ejemplo, haciendo uso de sistemas de geolocalización) o manual (introducida por el usuario).

20. La aplicación móvil podrá filtrar las notificaciones de sobre la necesidad de grupos sanguíneos basándose en el que el usuario introduzca.

Requisitos no funcionales

1. La aplicación corporativa deberá ser sencilla.

2. La aplicación corporativa deberá ser rápida.

3. La aplicación corporativa deberá ser segura.

4. La aplicación corporativa deberá ser robusta.

5. La interfaz de exposición de los datos deberá ser segura.

6. La aplicación móvil deberá ser sencilla.

Requisitos de información

1. Las ciudades contienen puntos de colecta, por lo que una ciudad puede tener más de un punto de colecta.

2. El stock contendrá los datos de los hematíes irradiados y no irradiados de todos los grupos sanguíneos con sus respectivos Rh (O+, O-, A+, A-, B+, B-, AB+,

AB-). Además incluirán los pooles de plaquetas irradiadas y no irradiadas de todos los grupos sanguíneos excluyendo el Rh (O, A, B, AB).

3. Las noticias tendrán título, subtítulo, imagen destacada y cuerpo de la noticia.

Restricciones

1. Todas las herramientas, frameworks, bases de datos, interfaces, entre otros, deberán ser open-source.

2. No se podrá obtener datos del usuario (email, teléfono, etc.) ni podrán ser transmitidos.

3. La aplicación corporativa podrá ejecutarse en ordenadores con poca potencia de procesamiento.

5.1.2. Casos de uso

A continuación se expondrán los casos de uso. En primer lugar se mostrarán aquellos correspondientes a la aplicación corporativa y en segundo lugar aquellos correspondientes a la aplicación móvil.

Casos de uso de la aplicación corporativa

En la figura 5.1 podemos observar un único actor, que es cualquier empleado del Banco de Sangre de Extremadura.

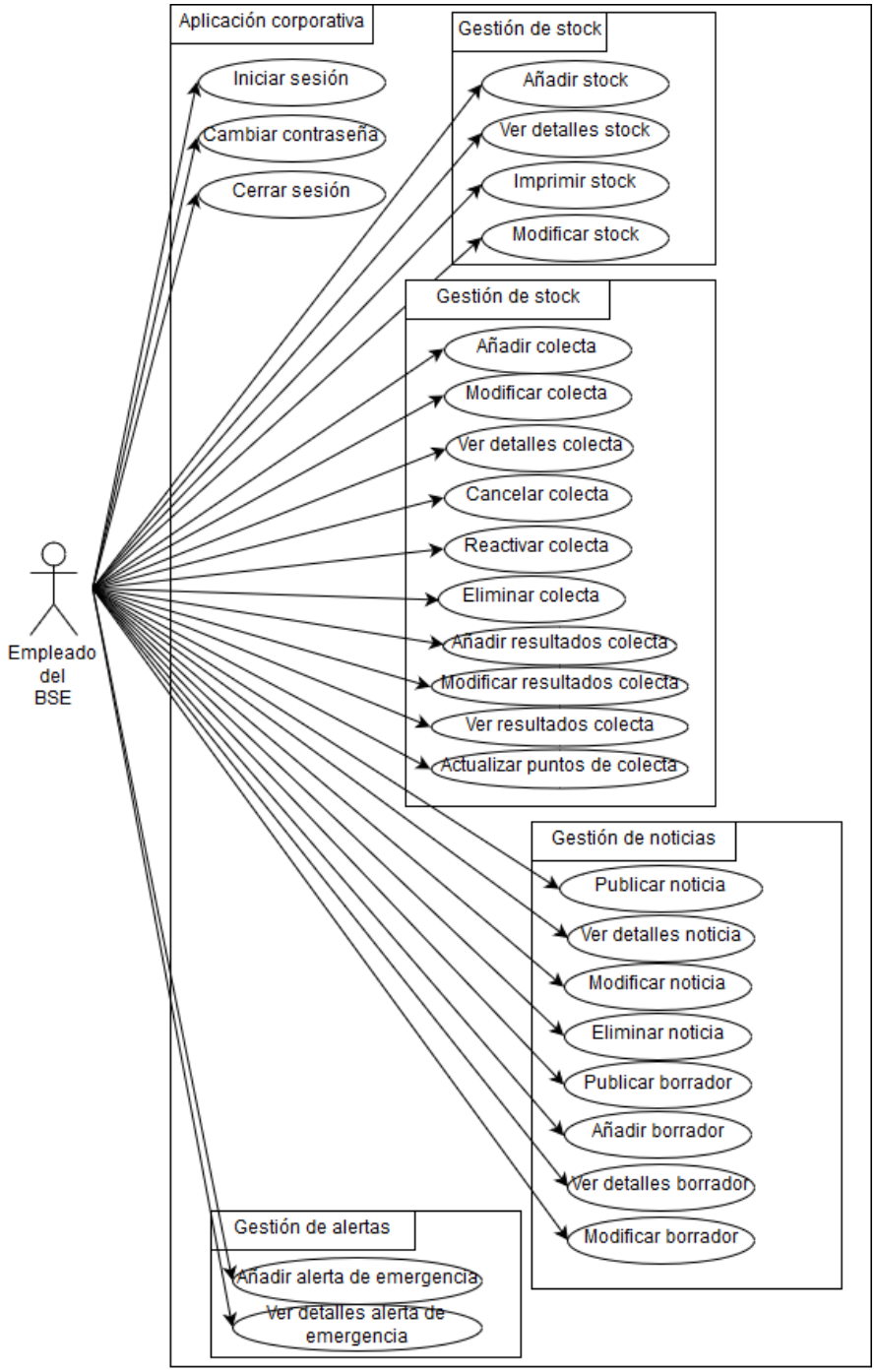


Figura 5.1: Diagrama de casos de uso de la aplicación corporativa

Casos de uso de la aplicación móvil

En la figura 5.2 podemos observar un único actor, que es cualquier potencial usuario del Banco de Sangre de Extremadura.

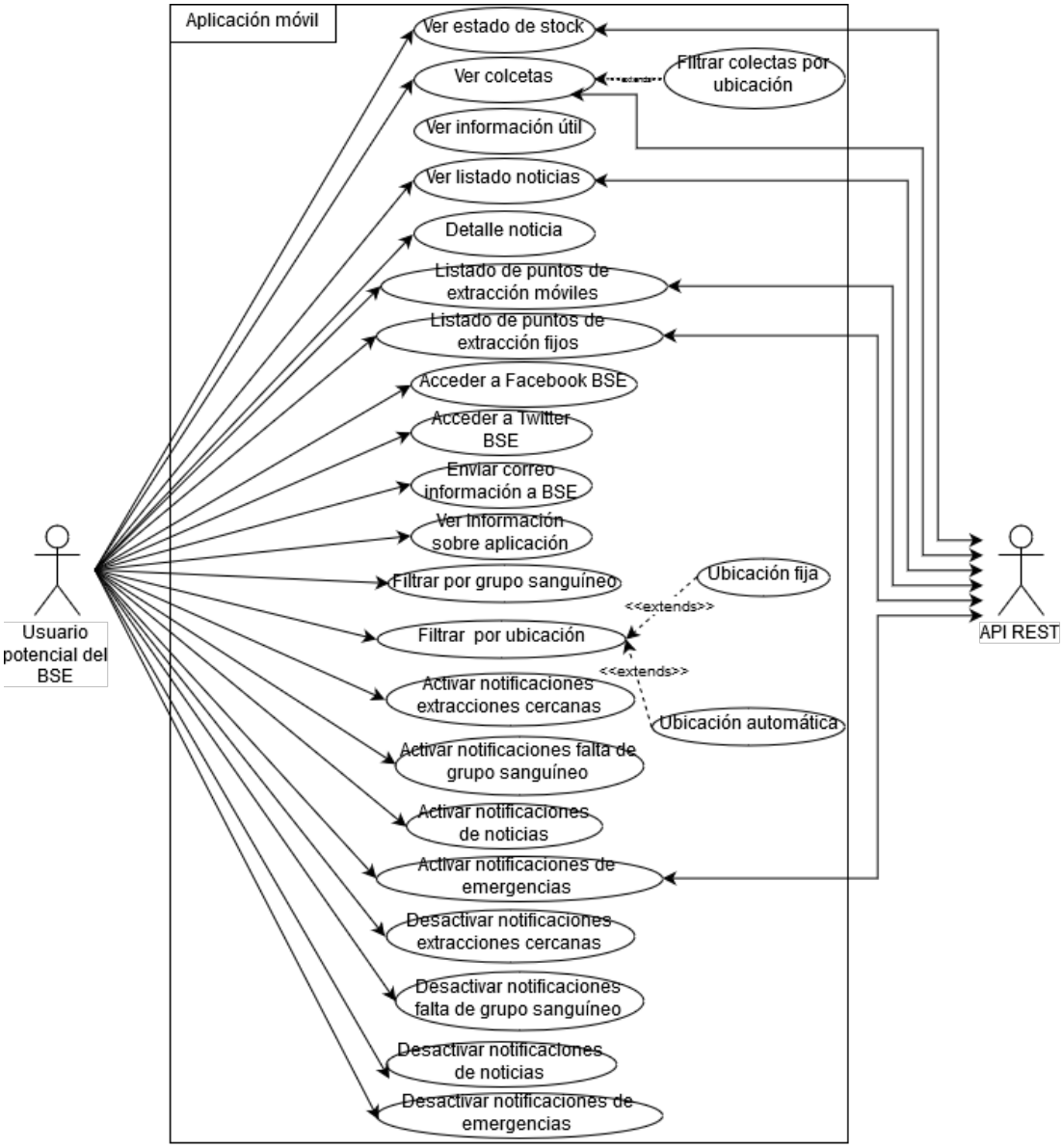


Figura 5.2: Diagrama de casos de uso de la aplicación móvil

5.1.3. Diagramas de arquitectura

En el diagrama de la figura 5.3, que es un diagrama de arquitectura general, podemos observar dos actores distintos, el empleado del BSE y el usuario con la aplicación móvil con los dos sistemas que se implementarán.

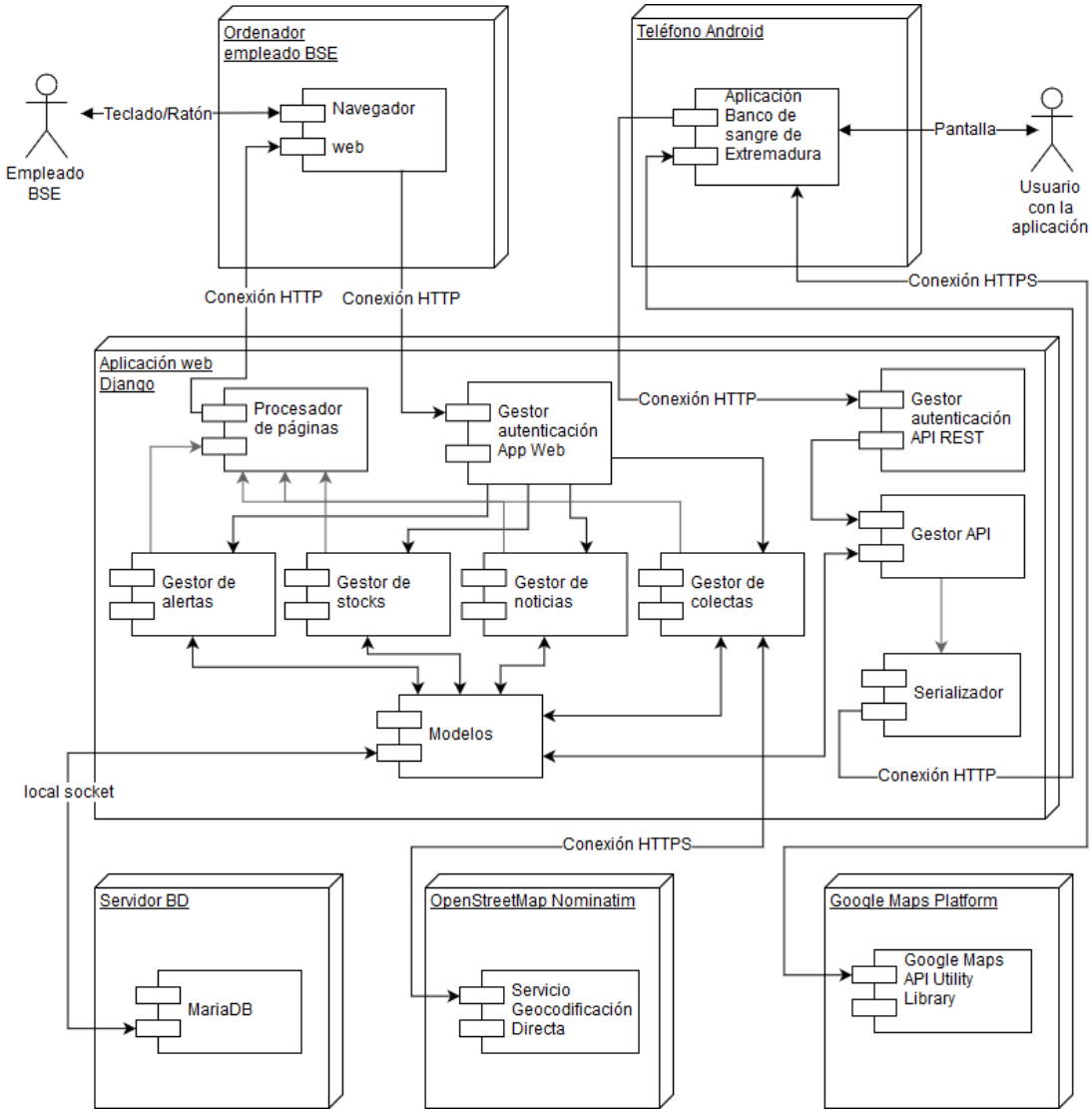


Figura 5.3: Diagrama de arquitectura

El empleado del BSE utilizará un ordenador o un dispositivo móvil (teléfono móvil, tablet, etc.) que tenga instalado un navegador web. A través de éste accederá a la aplicación web Django (mediante una conexión HTTP) que estará desplegada en un servidor web.

La aplicación web se conecta al servidor de base de datos mediante un socket local y a un servicio web de *geocodificación directa*[1] mediante una conexión HTTPS. A su vez, la aplicación web expone los datos a través de una API REST.

[1]Geocodificación directa: consiste en la obtención de coordenadas (latitud y longitud) utilizando la dirección de un punto del mapa.

Por otra parte, el usuario con un terminal Android que instale la aplicación. Accederá a los datos que son expuestos en la API REST de la aplicación web mediante una conexión HTTP y para mostrar las colectas en el mapa, utilizará el servicio de Google Maps para Android mediante una conexión HTTPS.

Arquitectura de una aplicación Django

Como hemos comentado anteriormente, Django tiene un patrón de arquitectura ligeramente diferente al Modelo-Vista-Controlador, que se denomina Modelo-Plantilla-Vista o (*Model-Template-View* en inglés). En la siguiente figura, se explica de una forma sencilla:

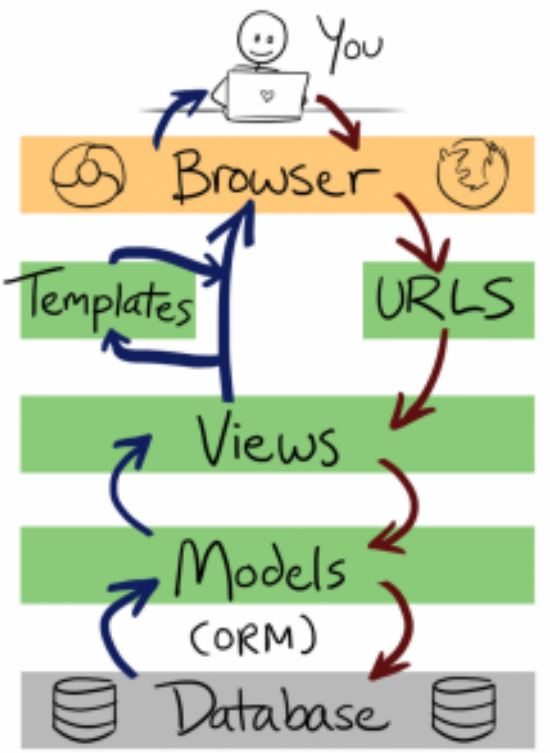


Figura 5.4: Diagrama del patrón de arquitectura Modelo-Plantilla-Vista
Fuente: http://blog.easylearning.guru/implementing-mtv-model-in-python-django/

Como podemos observar en la figura 5.4, en el patrón MTV existen dos flujos diferenciados, el rojo es el recorrido realizado por la petición del cliente hasta la fuente de datos y el azul es el que se realiza desde la fuente de datos hasta el usuario. El

proceso es el siguiente:

1. El usuario accede a la dirección a la que apunta la aplicación.
2. Django comprueba que es una de las URL que tiene registradas para la aplicación, si es así, carga la vista que está asociada a esta URL.
3. La vista, si requiere datos, buscará la entidad o entidades necesarias en el modelo.
4. El modelo, encargado de hacer el mapeo objeto-relacional (ORM), busca en la base de datos la entidad o entidades necesarias y realiza las consultas pertinentes para obtener los datos solicitados por el usuario.
5. La base de datos devuelve los datos al modelo.
6. El modelo transforma los datos para ser interpretados por Django mediante el ORM y devuelve los datos a la vista.
7. La vista se encargará de entregar los datos a la plantilla que los interpretará y mostrará de forma adecuada para, finalmente, devolver al usuario una respuesta adecuada a su solicitud.

La ruta seguida por la petición del usuario, no siempre sigue el mismo itinerario. Por ejemplo, el usuario accede a una dirección que Django no tiene registrada, por lo que lanzará una código de error 404. En este caso no accede a los modelos ni a la base de datos. el proceso sería el siguiente:

1. El usuario accede a la dirección a la que apunta la aplicación y a una sección que no existe.
2. Django no tiene entre las URL de la aplicación esta dirección. Lanza una excepción 404.
3. Django tiene una vista prediseñada para capturar esta excepción. La carga y, a su vez, envía los datos a la plantilla.

4. La plantilla por defecto puede sustituirse por una personalizada, en ese caso cargaría la personalizada, sino cargará la plantilla por defecto.

5. La vista con la plantilla se envía al usuario.

Como vemos, en este caso no se accede al modelo ni se accede a la fuente de datos.

Patrones arquitectónicos más comunes en Android

Vamos a tener en cuenta en primer lugar los patrones de arquitectura que relacionan la interfaz del usuario con la lógica de la aplicación.

Modelo - Vista - Controlador

Es un patrón de arquitectura que divide los datos de la aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos.

- **Modelo**: contiene una representación de los datos del sistema, la lógica de negocio y los mecanismos de persistencia.

- **Vista**: también denominada interfaz de usuario, está compuesta por la información enviada al cliente y los mecanismos de interacción con el mismo.

- **Controlador**: actúa como mediador entre el Modelo y la Vista, gestionando el flujo de información y las transformaciones de los datos necesarios para adaptarlos a cada una de estas capas. [39]

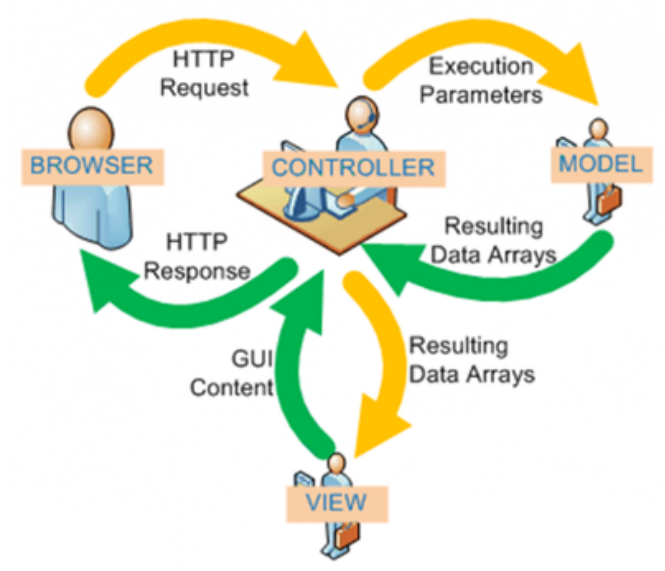


Figura 5.5: Flujo del patrón de arquitectura Modelo-Vista-Controlador
Fuente: Universidad de Alicante [39]

El flujo seguido en la figura 5.5 es el siguiente:

1. El cliente envía una petición al controlador.
2. El controlador solicita los datos al modelo.
3. El modelo devuelve los datos solicitados.
4. El controlador ofrece los datos a una vista determinada, que los incluye.
5. La vista, con los datos actualizados, es devuelta al controlador.
6. El controlador, con la vista ya preparada, se la devuelve al usuario en forma de una respuesta.

Modelo - Vista - Presentador

Es una derivación del anterior patrón de diseño que pretende desacoplar la vista de las responsabilidades del controlador.

- **Modelo**: representa los datos y la lógica de negocio.

- **Vista**: la vista ahora implementará una interfaz para que el presentador pueda eliminar la dependencia entre ambos. La vista no contendrá ninguna lógica de negocio.

- **Presentador**: es igual que el controlador del MVC pero ya no está vinculado a la vista, sino a la interfaz de ésta. Así se consigue un desacoplamiento que permitirá realizar pruebas de una forma mucho más sencilla.

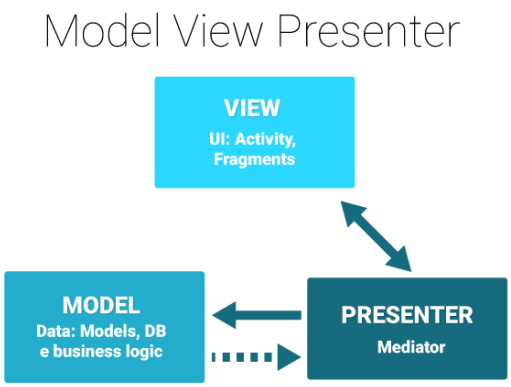


Figura 5.6: Flujo del patrón de arquitectura Modelo-Vista-Presentador
Fuente: https://blog.fossasia.org/introduction-to-mvp-in-susi-android-app/

El flujo seguido en la figura 5.6 es el siguiente:

1. El usuario solicita datos en la vista.

2. La vista traslada los solicitud al presentador.

3. El presentador se encarga de comprobar que la solicitud está permitida y de solicitarlos al modelo.

4. El modelo notifica la obtención de los datos al presentador.

5. El presentador obtiene los datos, comprueban que son correctos y los transforma para ser enviados a la vista.

6. La vista muestra los cambios al usuario.

Modelo - Vista - Modelo de vista

Este patrón de arquitectura es un rediseño del MVC. Permite el desacoplamiento de los tres elementos, es decir, el modelo de vista no conoce a la vista y el modelo no conoce a el modelo de vista, permitiendo, además, una gran reutilización.

- **Modelo**: contiene la lógica de negocio y los datos. No está vinculado ni a la vista ni al modelo de vista, haciéndolo reutilizable.
- **Vista**: Se enlaza a través de observadores y acciones expuestas por el modelo de vista. Es posible que varias vistas se unan a un único modelo de vista.
- **Modelo de vista**: Responsable de obtener los datos necesarios del modelo y preparar los datos observables que necesita la vista. Algo importante a tener en cuenta es que el modelo de vista no está vinculado a la vista.

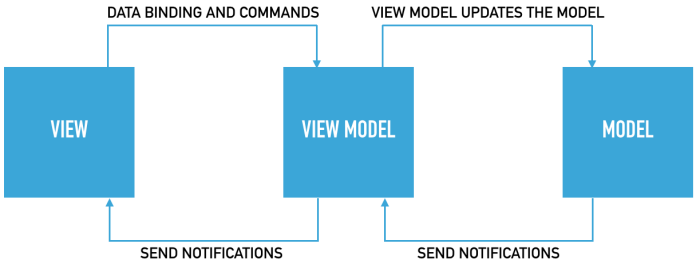


Figura 5.7: Flujo del patrón de arquitectura Modelo-Vista-Modelo de vista
Fuente: Medium.com [40]

El flujo seguido en la figura 5.7 es el siguiente:

1. El usuario solicita datos en la vista.
2. La vista traslada los solicitud al modelo de vista.
3. El presentador se encarga de comprobar que la solicitud está permitida y de solicitarlos al modelo.
4. El modelo notifica la obtención de los datos al presentador.

5. El modelo de vista obtiene los datos, comprueban que son correctos y notifica a la vista.

6. La vista muestra los cambios al usuario.

Patrón de diseño escogido

En nuestro caso, hemos decidido utilizar el patrón de diseño Modelo - Vista - Vista de Modelo. Además, desde Android Jetpack también nos lo recomiendan en su *Guía de arquitectura de apps*. [14] Además, también incluiremos el patrón **Repositorio**.

Este patrón nos facilitará la independencia de la fuente de datos, es decir, el **Repositorio** obtendrá los datos del **Modelo** o de una fuente externa de datos, como puede ser una **API REST**.

Esquema general de arquitectura de la aplicación Android

Este es el esquema general de arquitectura utilizado en la aplicación Android:

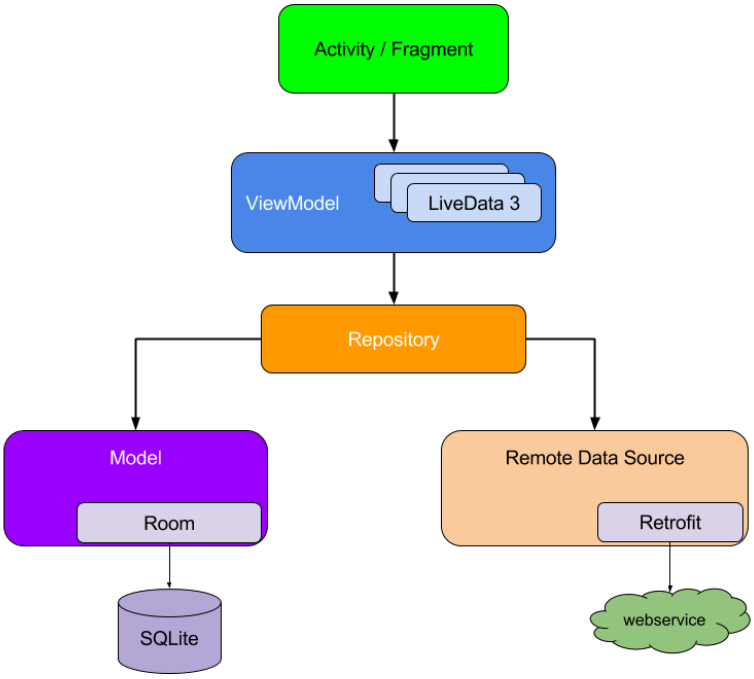


Figura 5.8: Arquitectura final de la aplicación Android
Fuente: Android developer [14]

En el apartado de implementación definiremos de forma individual cada uno de los componentes de la figura 5.8, como son las actividades y fragmentos, los modelos de vista, los repositorios, los modelos, la base de datos y la API REST.

5.1.4. Modelo de datos

Modelo de datos de la aplicación corporativa

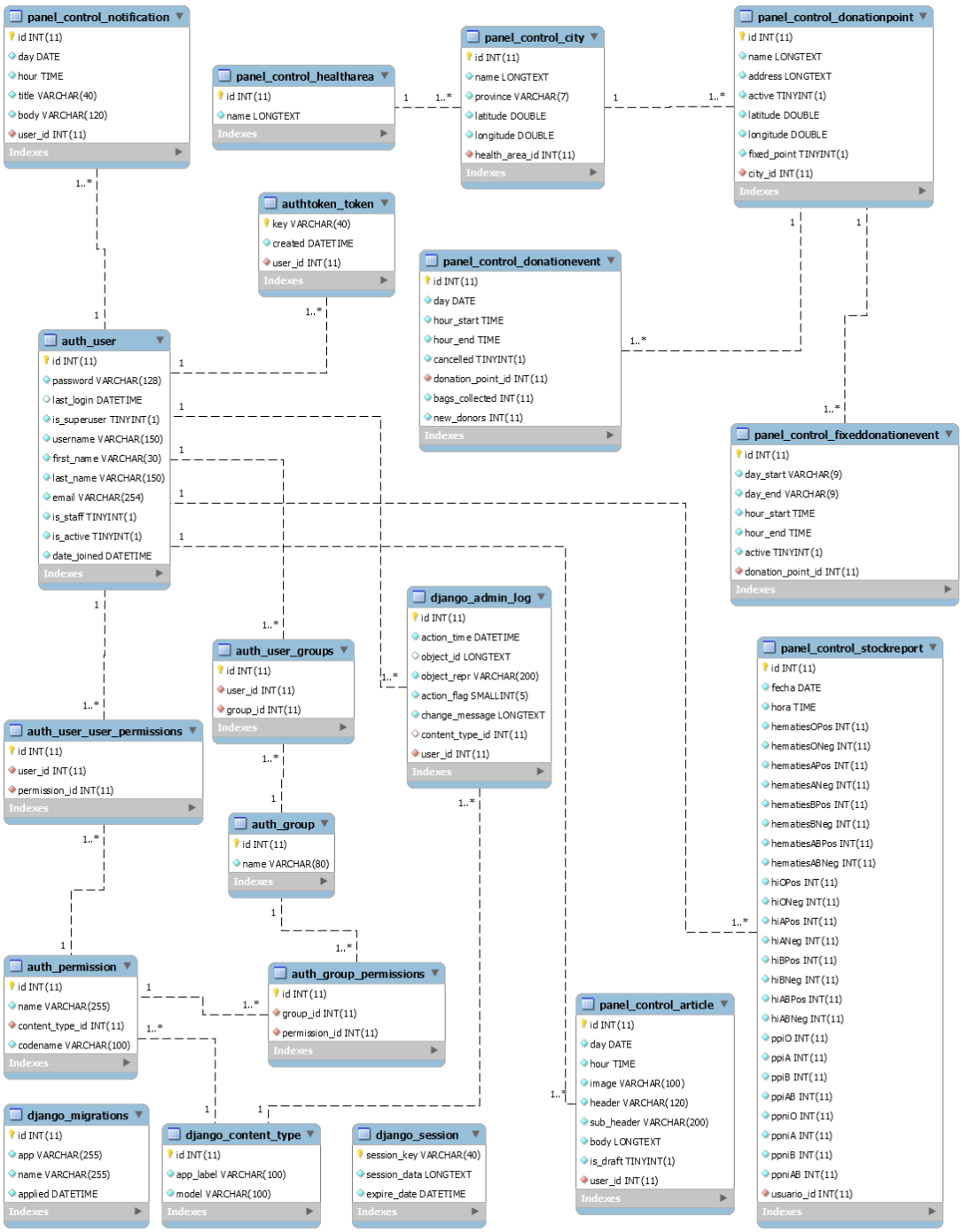


Figura 5.9: Diagrama de la base de datos de la aplicación corporativa

En la figura 5.9 podemos observar el modelo de datos de la aplicación corporativa detallado con las siguientes entidades:

- **Auth_user**: entidad perteneciente al módulo de administración de Django. Es una entidad en la que se almacenarán los datos de los usuarios.
- **Auth_user_user_permissions**: entidad perteneciente al módulo de administración de Django. Es una entidad que contiene el conjunto de permisos que tiene cada usuario.
- **Auth_permission**: entidad perteneciente al módulo de administración de Django. Es una entidad que contiene todos los permisos disponibles.
- **Django_content_type**: entidad perteneciente al módulo de administración de Django. Contiene información de las entidades que existen en la base de datos.
- **Django_admin_log**: entidad perteneciente al módulo de administración de Django. Contiene información de las acciones llevadas a cabo por el administrador.
- **Auth_group_permissions**: entidad perteneciente al módulo de administración de Django. Contiene el conjunto de permisos aplicables a grupos.
- **Auth_group**: entidad perteneciente al módulo de administración de Django. Esta entidad almacena el conjunto de grupos de usuarios.
- **Auth_user_group**: entidad perteneciente al módulo de administración de Django. Esta entidad intermedia almacena el conjunto de usuarios perteneciente a uno o varios grupos.
- **Django_migrations**: entidad perteneciente al módulo de administración de Django. Incluye cada una de las migraciones que se han realizado en el proyecto Django. Las migraciones son la forma en la que Django propaga los cambios a los modelos de la base de datos.

- **Django_session**: entidad perteneciente al módulo de administración de Django. Incluye cada una de las sesiones activas en la aplicación web.
- **Auth_token**: entidad perteneciente al módulo Django REST Framework que almacena los token generados por la aplicación para la API REST.
- **Panel_control_notification**: entidad que almacenará las alarmas de emergencia.
- **Panel_control_article**: entidad que almacenará las noticias.
- **Panel_control_stockreport**: entidad que almacenará los informes de stock.
- **Panel_control_healtharea**: entidad que almacenará las áreas de salud.
- **Panel_control_city**: entidad que almacenará las ciudades.
- **Panel_control_donationpoint**: entidad que almacenará los puntos de donación (o puntos de colecta).
- **Panel_control_donationevent**: entidad que almacenará los eventos de donación o colectas.
- **Panel_control_fixeddonationevent**: entidad que almacenará los eventos de donación o colectas fijas.

Modelo de datos de la aplicación móvil

En el diagrama de la figura 5.10 podemos observar las siguientes entidades:

- **Emergency**: esta entidad almacenará las alertas de emergencia notificadas al usuario.
- **AccessToken**: esta entidad almacenará los *tokens* que se obtengan periódicamente.
- **Article**: esta entidad almacenará las noticias obtenidas.
- **Stock**: esta entidad almacenará los datos de stock obtenidos.

- **City**: esta entidad almacenará los datos de las ciudades obtenidas.
- **DonationPoint**: esta entidad almacenará los datos de los puntos de donación o puntos de colecta obtenidos.
- **DonationEvent**: esta entidad almacenará los datos de los eventos de donación o colectas obtenidas.
- **FixedDonationEvent**: esta entidad almacenará los datos de los eventos de donación o colectas fijas obtenidas.

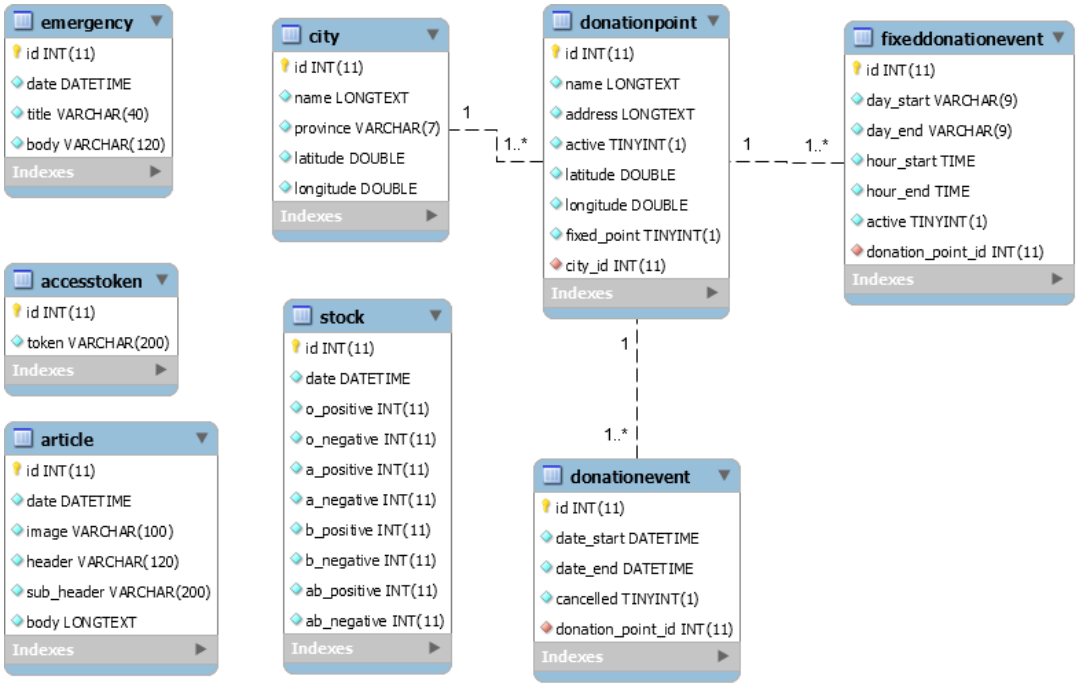


Figura 5.10: Diagrama de la base de datos de la aplicación móvil

5.2. Implementación

5.2.1. Aplicación web Django

El proyecto web está dividido en tres grandes partes, los modelos, las vistas y las plantillas, aunque también hay que tener en cuenta las URL.

Modelos

Los modelos de la figura 5.9 son los que hemos implementado en este apartado. Vamos a detallar cada uno de los atributos:

- **Article**:
  - **id**: identificador del artículo o noticia.
  - **day**: día que ha sido publicado el artículo (o guardado en el caso de ser un borrador).
  - **hour**: hora en la que ha sido publicado el artículo (o guardado en el caso de ser un borrador).
  - **user**: identificador del usuario que ha publicado la noticia (o ha modificado por última vez, en el caso de ser un borrador).
  - **image**: ruta de la imagen que será la cabecera de la noticia.
  - **header**: título de la noticia. Tendrá un tamaño máximo de 120 caracteres.
  - **subheader**: subtítulo de la noticia. Tendrá un tamaño máximo de 200 caracteres.
  - **body**: cuerpo de la noticia.
  - **is_draft**: indica si es una noticia (false) o si es un borrador (true).
- **HealthArea**:
  - **id**: identificador del área de salud.
  - **name**: nombre del área de salud.
- **City**:
  - **id**: identificador de la ciudad.
  - **name**: nombre de la ciudad.
  - **province**: nombre de la provincia a la que pertenece la ciudad.

- **health_area_id**: identificador del área de salud al que pertence la ciudad.
- **latitude**: latitud de la ciudad.
- **longitude**: longitud de la ciudad.

- **DonationPoint**:
  - **id**: identificador del punto de donación o punto de colecta.
  - **city_id**: identificador de la ciudad a la que pertenece este punto de donación.
  - **address**: dirección del punto de donación. Tendrá una forma similar a la siguiente: [Vía], [número], [código postal], [localidad], [provincia].
  - **active**: indica si este punto de donación está activo o se ha desactivado temporalmente.
  - **latitude**: latitud del punto de donación.
  - **longitude**: longitud del punto de donación.
  - **fixed_point**: indica si es un punto fijo (true) o si es un púnto móvil (false).

- **DonationEvent**:
  - **donation_point_id**: identificador del punto de donación.
  - **day**: día del evento de donación o colecta.
  - **hour_start**: hora de comienzo de la colecta.
  - **hour_end**: hora de finalización de la colecta.
  - **cancelled**: indica si el evento de donación o colecta ha sido cancelada.
  - **bags_collected**: indica el número de bolsas que se han conseguido obtener.
  - **new_donors**: indica el número de nuevos donantes.

- **FixedDonationEvent**:
  - **donation_point_id**: identificador del punto de donación o punto de colecta.

- **day_start**: día de comienzo de la colecta. Ejemplo: *Lunes*.
- **day_end**: día de finalización de la colecta. Ejemplo: *Viernes*.
- **hour_start**: hora de comienzo de la colecta.
- **hour_end**: hora de finalización de la colecta.
- **active**: indica si el evento de donación fijo o colecta fija está activo (true) o no (false).

- **StockReport**:
  - **fecha**: día en el que se ha creado o modificado por última vez un informe de stock.
  - **hora**: hora en la que se ha creado o modificado por última vez un informe de stock.
  - **usuario_id**: identificador del usuario que ha creado o modificado por última vez el informe de stock.
  - **hematiesOPos**: unidades de hematíes no irradiados O+.
  - **hematiesONeg**: unidades de hematíes no irradiados O-.
  - **hematiesAPos**: unidades de hematíes no irradiados A+.
  - **hematiesANeg**: unidades de hematíes no irradiados A-.
  - **hematiesBPos**: unidades de hematíes no irradiados B+.
  - **hematiesBNeg**: unidades de hematíes no irradiados B-.
  - **hematiesABPos**: unidades de hematíes no irradiados AB+.
  - **hematiesABNeg**: unidades de hematíes no irradiados AB-.
  - **hiOPos**: unidades de hematíes irradiados O+.
  - **hiONeg**: unidades de hematíes irradiados O-.
  - **hiAPos**: unidades de hematíes irradiados A+.
  - **hiANeg**: unidades de hematíes irradiados A-.

- **hiBPos**: unidades de hematíes irradiados B+.
- **hiBNeg**: unidades de hematíes irradiados B-.
- **hiABPos**: unidades de hematíes irradiados AB+.
- **hiABNeg**: unidades de hematíes irradiados AB-.
- **ppiO**: pooles de plaquetas irradiadas O.
- **ppiA**: pooles de plaquetas irradiadas A.
- **ppiB**: pooles de plaquetas irradiadas B.
- **ppiAB**: pooles de plaquetas irradiadas AB.
- **ppniO**: pooles de plaquetas no irradiadas O.
- **ppniA**: pooles de plaquetas no irradiadas A.
- **ppniB**: pooles de plaquetas no irradiadas B.
- **ppniAB**: pooles de plaquetas no irradiadas AB.

Para clarificar la relación entre estas entidades, vamos a mostrar un ejemplo:

En Extremadura hay ocho áreas de salud, una de ellas es **Badajoz**. Dentro de esta área de salud hay multitud de localidades, entre ellas **Alburquerque**. A su vez, en este municipio existen dos puntos de donación, el **I.E.S Castillo de Luna** y el **Centro de Salud**. Los eventos de donación son cada una de las colectas o extracciones que se programan en cada uno de estos puntos de donación.

Con respecto a los eventos de donación fijos, son aquellos que se llevan a cabo en puntos de donación fijos, en este caso, los hospitales. Los eventos no tienen un día numérico concreto de inicio ni fin, sino que tienen un día de comienzo (por ejemplo, lunes) y un día de fin. Un punto fijo sería el **Hospital Universitario de Badajoz** y un evento fijo sería **lunes a viernes de 10:00-14:00**.

Vistas de la aplicación corporativa

Las vistas son la lógica detrás de las pantallas que se muestran como resultado al cargar las páginas. En este caso tenemos las siguientes vistas:

Login

En esta vista se mostrará un formulario con un campo para el nombre de usuario y otro para la contraseña. No se podrá acceder en los siguientes casos:

- El usuario ha introducido mal el usuario, la contraseña o ambos.
- El usuario no está autorizado para entrar en el sitio web.
- El usuario ya no está activo.

Además, si el usuario lleva más de cuatro horas con la sesión iniciada, la sesión expirará, por lo que será necesario que vuelva a iniciar sesión. Esta vista utiliza la plantilla *login.html*

Home

Mostrará un mapa con los eventos programados para el día actual y el próximo día. Además contiene dos gráficos, uno con los hematíes[2] y otro con los pooles de plaquetas[3] disponibles. Los datos se obtienen del último *stock* guardado.

- **Mapa**: el mapa utiliza una librería para JavaScript llamada Leaflet, que a su vez utiliza los mapas de OpenStreetMap, disponibles de forma gratuita. Dentro de este mapa, aparecen con forma de chincheta roja los eventos programados para el día actual y de azul los que están programados para el próximo día.
- **Gráfico de hematíes**: cada barra es consecuencia de la suma de los hematíes no irradiados y de los hematíes irradiados para cada grupo sanguíneo teniendo en cuenta los anticuerpos (O+,O-, A+, etc.)
- **Gráfico de pooles de plaquetas**: cada barra es consecuencia de la suma de los pooles de plaquetas no irradiadas e irradiadas para cada grupo sanguíneo, sin tener en cuenta los anticuerpos (O, A, B y AB).

Esta vista utiliza la plantilla *home.html* y *body.html*

[2]Hematíes: eritrocitos o también llamados glóbulos rojos son, entre otros, hemocomponentes

[3]Pool de plaquetas: plaquetas procedentes de diversas unidades de sangre total, es decir, la mezcla de las plaquetas.

Stock Hemocomponentes

Mostrará una lista con el histórico de existencias de hemocomponentes, con las opciones de añadir uno nuevo, verlo en detalle, y modificarlo. Esta vista utiliza la plantilla *stock.html* y *body.html*

Añadir nuevo registro

Dará la opción al usuario de añadir un stock. Los datos añadidos en esta vista, serán guardados en el modelo *StockReport* de base de datos. Esta vista utiliza la plantilla *home.html* y *add_stock.html*

Ver stock

Dará la opción al usuario de mostrar en detalle un stock concreto. Además le dará la opción de imprimirlo o guardarlo como PDF. Esta vista utiliza la plantilla *home.html* y *review_stock.html*

Modificar stock

Dará la opción al usuario de modificar en detalle un stock concreto. Esta opción será dada si no ha pasado el tiempo límite de **tres días**. Esta vista utiliza la plantilla *home.html* y *modify_stock.html*

Programación

Mostrará una lista con los eventos programados que hayan sido insertados previamente. Dará la opción al usuario de añadir uno nuevo, de ver los eventos cancelados, de actualizar los puntos de donación y de añadir el resultado de los eventos finalizados. Además dará la opción de modificar el evento, de cancelarlo y de eliminarlo. Esta vista utiliza la plantilla *home.html* y *schedule.html*

Añadir evento de donación I

En esta primera vista se cargarán las áreas de salud, de las que el usuario seleccionará una y, previsiblemente, pasará a la siguiente vista. Esta vista utiliza la plantilla *home.html* y *add_collect_step1.html*

Añadir evento de donación II

En esta segunda vista se cargarán las ciudades pertenecientes al área de salud seleccionada en la vista anterior. El usuario podrá volver a la vista anterior por si se ha equivocado en la selección del área de salud o por el contrario podrá pasar a la siguiente vista. Esta vista utiliza la plantilla *home.html* y *add_collect_step2.html*

Añadir evento de donación III

En esta tercera y última vista se cargarán los puntos de donación pertenecientes a la ciudad previamente seleccionada. Además se deberán de introducir la fecha del evento, la hora de comienzo y la hora de fin. Esta vista tiene dos restricciones además de las evidentes (tener que introducir los datos):

- La fecha deberá ser mayor o igual que la actual.
- La hora de comienzo deberá ser menor que la hora de finalización.

Esta vista utiliza la plantilla *home.html* y *add_collect_step3.html*

Reactivar colectas canceladas

En esta vista, se mostrará al usuario un listado con los eventos pendientes de suceder (fecha mayor o igual a la del día actual) que se encuentran cancelados. Además se dará la opción de reactivarlos. Esta vista utiliza la plantilla *home.html* y *cancelled_collections.html*

Reactivar colecta cancelada

En esta vista, se mostrará al usuario el evento cancelado seleccionado para ser reactivado. Al usuario le aparecerá toda la información de ese evento y la posibilidad de reactivarlo o de volver a la lista de eventos cancelados. Esta vista utiliza la plantilla *home.html* y *recover_cancelled_collect.html*

Actualizar puntos de colecta

En esta vista, se mostrará al usuario un campo para poder introducir una hoja de cálculo de Microsoft Office para actualizar los puntos de donación. Esta vista utiliza la plantilla *home.html* y *update_donation_points.html*. Esto se ha realizado de esta forma por ser requisito imprescindible del cliente.

La hoja de cálculo tendrá dos hojas. La primera se llamará *Extremadura completo* y tendrá la siguiente estructura correspondiente a los puntos de donación móviles:

1. **Localidad**: el nombre de la localidad, por ejemplo: *Montijo*.

2. **Sitio de colecta**: el nombre identificativo del punto de donación, por ejemplo: *Centro de Salud*.

3. **Dirección**: la dirección exacta, o lo más precisa posible, del punto de donación, por ejemplo: *Avenida Puebla de la Calzada, 2, 06480 Montijo, Badajoz*

4. **Área de Salud**: área de salud a la que pertenece la localidad anterior, por ejemplo: *Badajoz*

5. **KM**: este campo no es relevante para el sistema, ya que solo es una pequeña orientación para el equipo del cliente.

6. **Estado**: nos indicará si este punto de donación es válido o no. Será valido si tiene una dirección válida.

7. **Provincia**: es la provincia a la que pertenece esta localidad, por ejemplo: *Badajoz*.

La segunda hoja se llamará *Puntos fijos* y contendrá todos los puntos que son fijos:

1. **Localidad**: el nombre de la localidad, por ejemplo: *Badajoz*.

2. **Sitio de colecta**: el nombre identificativo del punto de donación, por ejemplo: *Hospital Universitario de Badajoz*.

3. **Dirección**: la dirección exacta o lo más precisa posible del punto de donación fijo, por ejemplo: *Avenida de Elvas, 06080 Badajoz*

4. **Área de Salud**: área de salud a la que pertenece la ciudad a la que pertenece el punto de donación fijo, por ejemplo: *Badajoz*,

5. **KM**: este campo no es relevante para el sistema, ya que solo es una pequeña orientación para el equipo del cliente.

6. **Estado**: nos indicará si este punto de donación fijo es válido o no. Será valido si tiene una dirección válida.

7. **Provincia**: es la provincia a la que pertenece esta localidad, por ejemplo: *Badajoz*.

8. **Días**: este campo contiene el rango de días, en el que el punto de donación está abierto al público. Tiene la siguiente estructura "A-B", donde A es el día de comienzo por su inicial (L, M, X, J, V, S o D) y B es el día de finalización por su inicial. Ambos estarán incluidos en el rango.

9. **Horario**: será el horario de apertura al público. Tiene el siguiente formato "aa:bb-cc:dd", donde:

   - *aa*: hora de comienzo con formato 24h.
   - *bb*: minuto de comienzo.
   - *cc*: hora de finalización con formato 24h.
   - *dd*: minuto de finalización.

Por ejemplo, 09:00-13:00. El formato "a:bb-c:dd" también estará permitido, pero siempre teniendo en cuenta que las horas deberán estar con el formato 24h.

Una vez que el usuario pulse el botón, se actualizarán todos los puntos de colecta (tanto fijos como móviles) y los eventos fijos de donación, siguiendo el algoritmo del anexo C.

Resultados de colecta

En esta vista se mostrará un listado con los eventos de donación (o colectas) que han terminado o van a terminar en el día actual. En este listado, se podrá:

- **Ver el resultado**: siempre y cuando haya sido añadido previamente.
- **Añadir resultado**: se podrá añadir si no ha expirado el tiempo límite para insertarlo, que es de tres días.
- **Editar resultado**: se puede modificar si se ha añadido previamente y no ha expirado el tiempo para modificarlo, que es de tres días.

Esta vista utiliza la plantilla *home.html* y *result_collections.html*

Ver resultado de la colecta

En esta vista se mostrarán todos los datos del evento de donación (área de salud, localidad, punto de donación, fecha, hora de comienzo y hora de finalización, bolsas obtenidas y número de nuevos donantes). Solo se podrán ver estos datos si se han insertado previamente los resultados de esa colecta. Esta vista utiliza la plantilla *home.html* y *review_result_collect.html*

Añadir resultado de la colecta

En esta vista se mostrarán los datos del evento de donación (área de salud, localidad, punto de donación, fecha, hora de comienzo y hora de finalización) y se podrán insertar el número de bolsas obtenidas y el número de nuevos donantes, siempre

y cuando no haya expirado el periodo máximo de inserción de resultados, que es de tres días. Esta vista utiliza la plantilla *home.html* y *add_result_collect.html*

Modificar resultado de la colecta

En esta vista se mostrarán los datos del evento de donación (área de salud, localidad, punto de donación, fecha, hora de comienzo y hora de finalización) y se podrán modificar el número de bolsas obtenidas y el número de nuevos donantes, siempre y cuando no haya expirado el tiempo máximo de modificación, que es de tres días. Esta vista utiliza la plantilla *home.html* y *modify_result_collect.html*

Modificar evento de donación

En esta vista se mostrarán los datos del evento de donación y el usuario podrá modificar, tan solo la fecha, hora de comienzo y hora de finalización. Esta vista utiliza la plantilla *home.html* y *modify_collect.html*

Cancelar evento de donación

En esta vista se podrá cambiar de estado activo a cancelado de un evento de donación. Sólo se muestran los datos del evento de donación pero no se pueden modificar, solo cancelarlo. Esta vista utiliza la plantilla *home.html* y *cancel_collect.html*

Eliminar evento de donación

En esta vista se podrá eliminar un evento de donación . Se mostrarán los datos del evento y se le dará la opción de eliminarlo. Los eventos eliminados no pueden ser recuperados. Esta vista utiliza la plantilla *home.html* y *delete_collect.html*

Alertas de emergencia

En esta vista se mostrará una lista de emergencias. Estas emergencias son eventos excepcionales que hacen necesaria la donación de forma urgente, como puede ser un

accidente con muchos heridos. Las alertas tendrán dos estados posibles, enviadas o caducadas:

- **Enviada**: estado que tienen aquellas alertas que han sido enviadas y que aún no han expirado.
- **Caducada**: estado que tienen aquellas alertas que han expirado, es decir, ha pasado más de dos horas desde que han sido enviadas.

Esta vista utiliza la plantilla *home.html* y *alerts.html*

Añadir nueva alerta de emergencia

En esta vista se mostrarán los campos para añadir una alerta de emergencia. Hay campos que ya están rellenos de forma automática, que son *día*, *hora* y *creador de la alerta*. El usuario que crea la alerta tiene que rellenar los siguientes campos:

- **Título**: el título tendrá un tamaño máximo de 40 caracteres. Esto es así para no saturar la interfaz de las notificaciones de la aplicación Android.
- **Cuerpo**: el cuerpo tiene un tamaño de 120 caracteres.

Esta vista utiliza la plantilla *home.html* y *add_alert.html*

Noticias

En esta vista se mostrará una lista de noticias creadas. Las noticias creadas tienen dos estados, en **borrador** o **publicada**. Además se da la opción de ver la noticia en detalle, eliminar la noticia concreta, añadir una noticia y ver la lista de borradores. Esta vista utiliza la plantilla *home.html* y *news.html*

Añadir noticia

En esta vista aparecen campos precargados como son el *día*, la *hora* y el creador de la noticia. El usuario tendrá que rellenar los siguientes campos:

- **Título de la noticia**: el título tendrá un tamaño máximo de 120 caracteres.
- **Subtitulo de la noticia**: el subtítulo tendrá un tamaño máximo de 200 caracteres.
- **Imagen de la noticia**: imagen que será la cabecera de la noticia. Deberá tener un formado *jpg*, *png* o *gif*.
- **Cuerpo de la noticia**: el cuerpo de la noticia tiene formatos negrita, cursiva y subrayado. También tiene listas ordenadas y no ordenadas. Se podrá alinear el texto a la derecha, en el centro, a la izquierda y justificado. Se podrán añadir imágenes externas con enlaces. Se podrán añadir enlaces. Por último se podrán añadir y quitar.

La noticia se podrá publicar o se podrá guardar como borrador. Esta vista utiliza la plantilla *home.html* y *add_news.html*

Borradores

En esta vista aparece una lista de las noticias que están en borradores. Se da la opción de publicar, modificar y eliminar. Esta vista utiliza la plantilla *home.html* y *drafts.html*

Publicar borrador

En esta vista se muestra el borrador de la noticia con todos los campos de solo lectura. El usuario tan solo podrá publicar la noticia. Esta vista utiliza la plantilla *home.html* y *publish_news.html*

Modificar borrador

En esta vista se muestra el borrador de la noticia con los campos *título*, *subtítulo*, *imagen* y *cuerpo* modificables y los campos *día*, *hora* y *creador* siguen siendo de solo lectura. El usuario podrá modificar todos los campos modificables. Los no modificables se actualizarán. La noticia podrá ser publicada o actualizada como borrador. Esta vista utiliza la plantilla *home.html* y *modify_news.html*

Eliminar noticia

En esta vista se muestran todos los campos de la noticia antes de ser eliminada por el usuario. Las noticias y borradores eliminados no podrán ser recuperados. Esta vista utiliza la plantilla *home.html* y *delete_news.html*

Leer noticia

En esta vista se muestran todos los campos de la noticia o borrador sin poder ser modificados por el usuario. Esta vista utiliza la plantilla *home.html* y *review_news.html*

Perfil

En esta vista se muestran el nombre, apellidos y el tipo de usuario de solo lectura. Además se podrá actualizar la contraseña, que es recomendable cambiarla cada tres meses. Esta vista utiliza la plantilla *home.html* y *profile.html*

Plantillas

Las plantillas están divididas por conjunto de vistas, en concreto:

- **Stock**: contiene las plantillas: añadir stock (*add_stock.html*), modificar stock (*modify_stock.html*), revisar stock (*review_stock.html*) y listado de stock (*stock.html*).

- **Programación**: contiene las plantillas: actualizar los puntos de colecta (*update_donation_points.html*), añadir colecta paso 1 (*add_collect_step1.html*), añadir colecta paso 2 (*add_collect_step2.html*), añadir colecta paso 3 (*add_collect.html*), añadir resultado de la colecta (*add_result_collect.html*), cancelar colecta (*cancel_collect.html*), colectas canceladas (*cancelled_collections.html*), eliminar colectas (*delete_collect.html*), modificar colectas (*modify_collect.html*), modificar resultados de la colectas (*modify_result_collect.html*), listado de programación (*schedule.html*), recuperar colectas canceladas (*recover_cancelled_collect.html*), listado de los resultados

de las colectas (*results_collection.html*), revisar el resultado de las colectas (*review_result_collect.html*).

- **Noticias**: contiene las plantillas: añadir noticia (*add_news.html*), eliminar noticia (*delete_news.html*), borradores (*drafts.html*), modificar noticia (*modify_news.html*), listado de noticias (*news.html*), publicar noticia (*publish_news.html*) y revisar noticia (*review_news.html*).
- **Alertas**: contiene las plantillas: añadir alertas (*add_alert.html*) y listado de alertas (*alerts.html*).
- **Errores**: contiene las plantillas de los códigos de estado de HTTP: 400 (*400.html*), 403 (*403.html*), 404 (*404.html*) y 500 (*500.html*) personalizados.
- **Perfil**: plantilla de la vista perfil (*profile.html*).
- **Login**: plantilla de la vista de login (*login.html*).
- **Home**: plantilla de la vista home (*home.html*).
- **Body**: plantilla del cuerpo de HTML importado por el resto de plantillas (*body.html*) excepto la de acceso (*login.html*), cumpliendo el principio DRY.

URL

Las URL correspondientes a la aplicación, sin contar aquellas de la API REST son las siguientes:

URL	Vista asociada
/	Login
/home/	Home
/profile/	Perfil
/stock/	Stock Hemocomponentes
/stock/add/	Añadir nuevo registro

URL	Vista asociada
/stock/review/(id_stock)/	Ver stock
/stock/modify/(id_stock)/	Modificar stock
/schedule/	Programación
/schedule/add-step1/	Añadir evento de donación I
/schedule/add-step2/	Añadir evento de donación II
/schedule/add-step3/	Añadir evento de donación III
/schedule/recover/	Rectivar colectas canceladas
/schedule/recover/(id_evento)	Reactivar colecta cancelada
/schedule/update/	Actualizar puntos de colecta
/schedule/results/	Resultados de colecta
/schedule/results/review/(id_evento)/	Ver resultado de la colecta
/schedule/results/add/(id_evento)/	Añadir resultado de la colecta
/schedule/results/modify/(id_evento)/	Modificar resultado de la colecta
/schedule/modify/(id_evento)/	Modificar evento de donación
/schedule/cancel/(id_evento)/	Cancelar evento de donación
/schedule/delete/(id_evento)/	Eliminar evento de donación
/app/alerts/	Alertas de emergencia
/app/alerts/add/	Añadir nueva alerta de emergencia
/app/news/	Noticias
/app/news/add/	Añadir noticia
/app/news/drafts/	Borradores
/app/news/publish/(id_noticia)/	Publicar borrador
/app/news/modify/(id_noticia)/	Modificar borrador
/app/news/delete/(id_noticia)/	Eliminar noticia
/app/news/review/(id_noticia)/	Leer noticia

Tabla 5.1: Conjunto de URL disponibles en la aplicación web

Complementos de terceros utilizados

HTML, CSS y JS

- **Leaflet**: es una librería de JavaScript de código fuente abierto para mapas interactivos. Se utiliza en la vista **Home**. [41]
- **Chart.js**: es una librería de JavaScript de código fuente abierto para mostrar gráficos en HTML5. Se utiliza en la vista **Home**. [42]
- **Bootstrap**: es un conjunto de herramientas de código fuente abierto para desarrollar con HTML, CSS y JS. Se utiliza en todas las vistas de la aplicación excepto en las pertenecientes a la API REST y aquellas que vienen por defecto como el sitio web administrador de Django. [43]
- **JQuery**: es una librería JavaScript que simplifica la forma de interactuar con los documentos HTML, manipulación del DOM, manejar eventos, animaciones y Ajax que funciona en la mayoría de navegadores. Se utiliza de forma conjunta a Bootstrap ya que muchos de los componentes de éste hacen uso de JQuery. [44][45]
- **Popper.js**: es una librería de JavaScript utilizada para gestionar las descripciones emergentes. Es usado de forma conjunta a Bootstrap, por la misma razón que JQuery. [46]

Fuentes

- **Google Fonts. Material Icons**: es un conjunto de iconos que son utilizados en el sistema operativo Android y que sigue una línea de diseño denominada *Material design*. Se utiliza en toda la aplicación.

Python

- **Pandas**: es una librería de Python de código fuente abierto, con licencia BSD que provee estructuras de datos y herramientas para análisis de datos de fácil uso

y con un gran rendimiento. En este proyecto ha sido utilizado para extraer los datos de una hoja de cálculos de Microsoft Excel™y manipularlos. [47]

- **Xlrd**: es una librería de para extraer datos de hojas de cálculo de Microsoft Excel™. [48]
- **Geopy**: es un cliente para varios servicios web de geocodificación populares, entre ellos **OpenStreetMap Nominatim**. Es usado en la aplicación con este servicio web para hacer geocodificación directa y obtener la ubicación de los puntos de donación en base a su dirección. [49]
- **ReportLab**: es una librería de código fuente abierto que permite la creación de documentos de forma dinámica. En la aplicación web es utilizado para generar los informes del *stock de hemocomponentes*.
- **Django-Jchart**: es un paquete para Django que es usado para mostrar gráficas utilizando la librería **Chart.js**.
- **Django-CKEditor**: es un paquete para Django utilizado para utilizar de forma sencilla CKEditor4. CKEditor4 es un editor HTML WYSIWYG [4] de código fuente abierto. [50][51]
- **Django Rest Framework**: es un potente conjunto de herramientas para construir APIs Web para Django.
- **Django-IPRestrict**: es un paquete para Django y un middleware para restringir el acceso a todo o partes de una aplicación Django mediante el uso de rangos de IP del cliente. [52]

5.2.2. API REST con Django REST Framework

En este caso hemos apostado por el uso de Django REST Framework, ya que nos permite añadir algo más de funcionalidad, en concreto, para añadir más seguridad, se

[4] What You See Is What You Get: normalmente referido a editores de texto que lo muestran como si fuese el documento final.

añadirá autenticación por *Token*.

Vistas

Las llamadas a la API REST también tienen vistas asociadas, pero no tienen *plantillas*, a no ser que queramos añadirla o utilizar la que nos provee por defecto. Las vistas son las siguientes:

Login y obtención del Token

En esta vista se obtendrá autenticándose con usuario y contraseña un *Token* temporal para obtener los datos de la API. El proceso es el siguiente:

1. El usuario deberá enviar una petición POST que en la cabecera de la petición HTTP incluirá "Content-Type: application/json" y en el cuerpo deberá suministrar las credenciales (usuario y contraseña) de la siguiente forma:

```
{
        "username": "usuario",
        "password": "contraseña"
}
```

2. El usuario recibirá el token, siempre y cuando este autenticado correctamente y sea autorizado.

3. Para realizar la consulta de los datos en cualquiera de las URIs que se describirán posteriormente, se deberá incluir el siguiente campo en la cabecera de la petición HTTP GET:

   "Authorization: Token (token obtenido en el paso anterior)"

Listado de ciudades

Esta vista se encarga de obtener todas las ciudades almacenadas. La respuesta será la siguiente:

```
[{
        "id": [Long: identificador de la ciudad],
        "name": [String: nombre de la ciudad],
        "province": [String: nombre de la provincia],
        "latitude": [Float: latitud de la ciudad],
        "longitude": [Float: longitud de la ciudad]
}]
```

Además, se pueden utilizar los siguientes filtros: *Obtención de una ciudad por nombre concreto*

```
/city?name=Mérida
```

Obtención de las ciudades pertenecientes a una provincia

```
/city?province=Badajoz
```

Ciudad individual

Esta vista se encarga de obtener la ciudad con el identificador que suministra el cliente. Si existe la respuesta será la siguiente:

```
{
        "id": [Long: identificador de la ciudad],
        "name": [String: nombre de la ciudad],
        "province": [String: nombre de la provincia],
        "latitude": [Float: latitud de la ciudad],
        "longitude": [Float: longitud de la ciudad]
}
```

En caso contrario se obtendrá un error 404 - No encontrado con la siguiente respuesta:

```
{
    "detail": "No encontrado."
```

```
}
```

Listado de puntos de donación

Esta vista se encarga de mostrar todos los puntos de donación. Tiene la siguiente estructura:

```
[{
        "id": [Long: identificador del punto de donación],
        "city": [Long: identificador de la ciudad],
        "name": [String: nombre del punto de donación],
        "fixed_point": [Boolean: indicador de punto fijo o móvil],
        "latitude": [Float: latitud del punto de donación],
        "longitude": [Float: longitud del punto de donación]
}]
```

Además, se pueden utilizar los siguientes filtros: *Obtención de los puntos de colecta filtrados por el identificador de la ciudad*

```
/point?city=121
```

Obtención de los puntos de colecta filtrados por nombre

```
/city?name=Centro de Salud Mérida Norte
```

Obtención de los puntos de colecta filtrados por ser o no punto fijo

```
/city?fixed_point=true
```

Punto de donación individual

Esta vista se encarga de mostrar un punto de donación individual con el identificador proporcionado por el cliente. La estructura es la siguiente:

```
{
```

```
        "id": [Long: identificador del punto de donación],
        "city": [Long: identificador de la ciudad],
        "name": [String: nombre del punto de donación],
        "fixed_point": [Boolean: indicador de punto fijo o móvil],
        "latitude": [Float: latitud del punto de donación],
        "longitude": [Float: longitud del punto de donación]
}
```

Si este punto no se encuentra por que no existe, se obtendrá un código de error del cliente 404 - No encontrado y el siguiente mensaje:

```
{
    "detail": "No encontrado."
}
```

Listado de eventos móviles

Esta vista se encarga de mostrar todos los eventos de donación que tienen una fecha igual o superior a la actual. Tiene la siguiente estructura:

```
[{
        "id": [Long: identificador del evento],
        "donation_point": [Long: id. del punto de donación],
        "start_date": [Datetime: fecha de comienzo del evento],
        "end_date": [Datetime: fecha de finalización del evento],
        "cancelled": [Boolean: indicado si el evento esta activo]
}]
```

Además, se pueden utilizar los siguientes filtros: *Obtención de las colectas filtrados por la fecha de hoy*

```
/event?day=today
```

Obtención de las colectas filtrados por la fecha de mañana

```
/event?day=tomorrow
```

Obtención de las colectas filtrados por una fecha concreta

```
/event?day=2019-01-01
```

Obtención de las colectas filtrados un rango de días. Este por ejemplo mostrará los que hay en esta semana.

```
/event?range=7
```

Evento móvil individual

Esta vista se encarga de mostrar un evento de donación individual con el identificador proporcionado por el cliente. La estructura es la siguiente:

```
{
        "id": [Long: identificador del evento],
        "donation_point": [Long: id. del punto de donación],
        "start_date": [Datetime: fecha de comienzo del evento],
        "end_date": [Datetime: fecha de finalización del evento],
        "cancelled": [Boolean: indicado si el evento esta activo]
}
```

Si este evento no se encuentra por que no existe o por estar caducado, se obtendrá un código de error del cliente 404 - No encontrado y el siguiente mensaje:

```
{
    "detail": "No encontrado."
}
```

Listado de eventos fijos

Esta vista se encarga de mostrar todos los eventos de donación fijos activos. Tiene la siguiente estructura:

```
[{
        "id": [Long: identificador del evento fijo],
        "donation_point": [Long: id. del punto de donación],
        "day_start": [String: nombre del día de comienzo],
        "day_end": [String: nombre del día de finalización],
        "hour_start": [Time: hora de comienzo],
        "hour_end": [Time: hora de finalización]
}]
```

Evento fijo individual

Esta vista se encarga de mostrar un evento fijo de donación individual con el identificador proporcionado por el cliente. La estructura es la siguiente:

```
{
        "id": [Long: identificador del evento fijo],
        "donation_point": [Long: id. del punto de donación],
        "day_start": [String: nombre del día de comienzo],
        "day_end": [String: nombre del día de finalización],
        "hour_start": [Time: hora de comienzo],
        "hour_end": [Time: hora de finalización]
}
```

Si este evento no se encuentra por que no existe o por no estar activo, se obtendrá un código de error del cliente 404 - No encontrado y el siguiente mensaje:

```
{
    "detail": "No encontrado."
}
```

Listado de noticias

Esta vista se encarga de mostrar todas las noticias que han sido publicadas (no son borradores). Tiene la siguiente estructura:

```
[{
        "id": [Long: identificador de la noticia],
        "date": [Datetime: fecha de la noticia],
        "image_url": [String: URL de la imagen de la noticia],
        "header": [String: título de la noticia],
        "sub_header": [String: subtítulo de la noticia],
        "body": [String: cuerpo de la noticia en HTML]
}]
```

Noticia individual

Esta vista se encarga de mostrar una noticia individual con el identificador proporcionado por el cliente. La estructura es la siguiente:

```
{
        "date": [Datetime: fecha de la noticia],
        "image_url": [String: URL de la imagen de la noticia],
        "header": [String: título de la noticia],
        "sub_header": [String: subtítulo de la noticia],
        "body": [String: cuerpo de la noticia en HTML]
}
```

Si esta noticia no se encuentra por que no existe o por no estar publicada, se obtendrá un código de error del cliente 404 - No encontrado y el siguiente mensaje:

```
{
    "detail": "No encontrado."
}
```

Listado de emergencias

Esta vista se encarga de mostrar todas las emergencias que han sido publicadas. Tiene la siguiente estructura:

```
[{
        "id": [Long: identificador de la emergencia],
        "date": [Datetime: fecha de la emergencia],
        "title": [String: título de la emergencia],
        "body": [String: cuerpo de la emergencia]
}]
```

Emergencia individual

Esta vista se encarga de mostrar una emergencia individual con el identificador proporcionado por el cliente. La estructura es la siguiente:

```
{
        "id": [Long: identificador de la emergencia],
        "date": [Datetime: fecha de la emergencia],
        "title": [String: título de la emergencia],
        "body": [String: cuerpo de la emergencia]
}
```

Si esta emergencia no se encuentra por que no existe, se obtendrá un código de error del cliente 404 - No encontrado y el siguiente mensaje:

```
{
    "detail": "No encontrado."
}
```

Stock actual

Esta vista se encarga de mostrar el stock actual de hemocomponentes. La estructura es la siguiente:

```
{
        "id": [Long: identificador del stock],
        "date": [Datetime: fecha del stock],
        "o_positive": [Long: unidades de hematíes O+],
        "o_negative": [Long: unidades de hematíes O-],
        "a_positive": [Long: unidades de hematíes A+],
        "a_negative": [Long: unidades de hematíes A-],
        "b_positive": [Long: unidades de hematíes B+],
        "b_negative": [Long: unidades de hematíes B-],
        "ab_positive": [Long: unidades de hematíes AB+],
        "ab_negative": [Long: unidades de hematíes AB-],
}
```

URL

Las URL correspondientes a la API REST son las siguientes:

URL	Vista asociada
/api/v1/get-token	Login y obtención del *Token*
/api/v1/city	Listado de ciudades
/api/v1/city/(city_id)	Ciudad individual
/api/v1/point	Listado de puntos de donación
/api/v1/point/(point_id)	Punto de donación individual
/api/v1/event	Listado de eventos móviles
/api/v1/event/(id_event)	Evento móvil individual
/api/v1/event/fixed	Listado de eventos fijos

URL	Vista asociada
/api/v1/event/fixed/(id_event)	Evento fijo individual
/api/v1/news	Listado de noticias
/api/v1/news/(id_news)	Noticia individual
/api/v1/emergency	Listado de emergencias
/api/v1/emergency/(id_emergency)	Emergencia individual
/api/v1/stock	Stock actual

Tabla 5.2: Conjunto de URIs disponibles en la API REST

Modificaciones de la autenticación

Como hemos descrito anteriormente, una de las razones por las que se ha utilizado este *framework* es la posibilidad de usar autenticación basada en *Token*. Para implementarlo tan solo es necesario incluir en la lista de las aplicaciones instaladas de Django *rest_framework.authtoken* y crear la vista que gestione la obtención de *Tokens*.

Pero en nuestro caso hemos querido ofrecer algo más de seguridad, haciendo que los *tokens* tengan una fecha de uso limitada a 4 semanas. Por esto, una vez que transcurra este periodo, se deberá obtener un nuevo *token*. Se advertirá en las siguientes situaciones:

- El *token* ha caducado.
- El *token* no es válido: si no existe o es incorrecto.
- El usuario no está activo o ha sido eliminado.

5.2.3. Aplicación Android

Modelo

El modelo tendrá las clases que se podrán obtener mediante la API REST.

Article

Esta entidad se relaciona con la entidad remota **Article**. Contiene los siguientes atributos:

- **id**: identificador de la noticia.
- **date**: fecha y hora de la noticia.
- **imageURL**: URL de la imagen de cabecera.
- **header**: título de la noticia.
- **sub_header**: subtítulo de la noticia.
- **body**: cuerpo de la noticia.

City

Esta entidad se relaciona con la entidad remota **City**. Contiene los siguientes atributos:

- **id**: identificador de la localidad.
- **name**: nombre de la localidad.
- **province**: nombre de la provincia a la que pertenece la localidad.
- **latitude**: latitud de la localidad.
- **longitude**: longitud de la localidad.

DonationPoint

Esta entidad se relaciona con la entidad remota **DonationPoint**. Contiene los siguientes atributos:

- **id**: identificador del punto de donación.

- **city**: identificador de la ciudad a la que pertenece el punto de donación.
- **name**: nombre identificativo del punto de donación.
- **fixed_point**: valor que indica si es un punto fijo o un punto móvil.
- **latitude**: latitud del punto de donación.
- **longitude**: longitud del punto de donación.
- *city*: referencia a la ciudad del punto de donación.

DonationEvent

Esta entidad se relaciona con la entidad remota **DonationEvent**. Contiene los siguientes atributos:

- **id**: identificador del evento de donación.
- **donation_point**: identificador del punto de donación al que pertenece el evento de donación.
- **start_date**: fecha y hora de comienzo del evento de donación.
- **end_date**: fecha y hora de finalización del evento de donación.
- **cancelled**: valor que indicará si el evento de donación ha sido cancelado.
- *donationPoint*: referencia al punto de donación del evento de donación.

FixedDonationEvent

Esta entidad se relaciona con la entidad remota **FixedDonationEvent**. Contiene los siguientes atributos:

- **id**: identificador del evento fijo de donación.
- **donation_point**: identificador del punto de donación al que pertenece el evento fijo de donación.

- **start_date**: día de comienzo del evento fijo de donación.
- **end_date**: día de finalización del evento fijo de donación.
- **hour_start**: hora de comienzo del evento fijo de donación.
- **hour_end**: hora de finalización del evento fijo de donación.
- *donationPoint*: referencia al punto de donación del evento fijo de donación.

Emergency

Esta entidad se relaciona con la entidad remota **Emergency**. Contiene los siguientes atributos:

- **id**: identificador de la emergencia.
- **date**: fecha y hora de la publicación de la emergencia.
- **title**: título de la emergencia.
- **body**: cuerpo de la emergencia.

Stock

Esta entidad se relaciona con la entidad remota **Stock**. Contiene los siguientes atributos:

- **id**: identificador del stock.
- **date**: fecha y hora de la publicación del stock.
- **o_positive**: unidades de O+.
- **o_negative**: unidades de O-.
- **a_positive**: unidades de A+.
- **a_negative**: unidades de A-.

- **b_positive**: unidades de B+.
- **b_negative**: unidades de B-.
- **ab_positive**: unidades de AB+.
- **ab_negetive**: unidades de AB-.

AccessToken

Esta entidad de base de datos tiene una finalidad diferente a las anteriores y por ello no aparece en el diagrama anterior. Esta entidad se utiliza para obtener de forma temporal el valor del *token* que permite acceder a los datos del resto de las entidades. Tiene los siguientes atributos:

- **id**: identificador del token autogenerado.
- **token**: cadena alfanumérica que permitirá acceder a los contenidos (el resto de entidades).

API REST con Retrofit

RetrofitRestClient

La creación del cliente con Retrofit es muy sencillo. Tan solo es necesario darle algunos parámetros como por ejemplo la URL base de la API REST y el *usuario* y *contraseña* para acceder poniéndolos en el cuerpo de la petición y una vez obtenido el *token* y guardado en una *SharedPreference*, pasará a realizar una petición con el *token* en la cabecera de la petición.

RestInterface

Interfaz utilizada por el cliente Retrofit para acceder a las diferentes URLs correspondiente a distintos datos definidas en la tabla 5.2.

Repositorios

En nuestro caso hemos creado un repositorio por entidad creada.

NewsRepository

Repositorio encargado de obtener los datos de las **noticias**. Estas noticias se obtendrán:

1. Se obtiene de la base de datos.
2. Se comprueba si ha pasado más de 10 minutos desde la última consulta a la API REST. Si es así, se vuelve a consultar y la respuesta obtenida se insertará en base de datos y se recuperará de la misma y será actualizado el observador con este valor. Si no, se pondrá el obtenido previamente de la base de datos en el observador.

Estos 10 minutos están pensados para no saturar de peticiones al servidor web ya que los cambios en los datos externos no cambian con tanta rapidez.

CityRepository

Repositorio encargado de obtener los datos de las **localidades**. Estas localidades se obtendrán:

1. Se obtiene de la base de datos.
2. Se comprueba si ha pasado más de 10 minutos desde la última consulta a la API REST. Si es así, se vuelve a consultar y la respuesta obtenida se insertará en base de datos y se recuperará de la misma y será actualizado el observador con este valor. Si no, se pondrá el obtenido previamente de la base de datos en el observador.

Estos 10 minutos están pensados para no saturar de peticiones al servidor web ya que los cambios en los datos externos no cambian con tanta rapidez.

DonationPointRepository

Repositorio encargado de obtener los datos de los **puntos de donación**. Estos puntos de donación se obtendrán:

1. Se obtiene de la base de datos.

2. Se comprueba si ha pasado más de 10 minutos desde la última consulta a la API REST. Si es así, se vuelve a consultar y la respuesta obtenida se insertará en base de datos y se recuperará de la misma y será actualizado el observador con este valor. Si no, se pondrá el obtenido previamente de la base de datos en el observador.

Estos 10 minutos están pensados para no saturar de peticiones al servidor web ya que los cambios en los datos externos no cambian con tanta rapidez.

DonationEventRepository

Repositorio encargado de obtener los datos de los **eventos de donación**. Estos eventos de donación se obtendrán:

1. Se obtiene de la base de datos.

2. Se comprueba si ha pasado más de 10 minutos desde la última consulta a la API REST. Si es así, se vuelve a consultar y la respuesta obtenida se insertará en base de datos.

3. Se enlaza cada evento con su punto y a su vez con su ciudad, para tener la estructura completa.

4. Estos eventos, se hayan obtenido de un origen u otro, se enviarán al observador.

Estos 10 minutos están pensados para no saturar de peticiones al servidor web ya que los cambios en los datos externos no cambian con tanta rapidez.

FixedDonationEventRepository

Repositorio encargado de obtener los datos de los **eventos fijos de donación**. Estos eventos fijos de donación se obtendrán:

1. Se obtiene de la base de datos.
2. Se comprueba si ha pasado más de 10 minutos desde la última consulta a la API REST. Si es así, se vuelve a consultar y la respuesta obtenida se insertará en base de datos.
3. Se enlaza cada evento fijo con su punto y a su vez con su ciudad, para tener la estructura completa.
4. Estos eventos, se hayan obtenido de un origen u otro, se enviarán al observador.

Estos 10 minutos están pensados para no saturar de peticiones al servidor web ya que los cambios en los datos externos no cambian con tanta rapidez.

EmergencyRepository

Repositorio encargado de obtener los datos de las **emergencias**. Estas emergencias se obtendránde la API REST. Se guarda en base de datos y se pasa al observador.

StockRepository

Repositorio encargado de obtener los datos del último **stock**. Este stock se obtendrá:

1. Se obtiene de la base de datos.
2. Se comprueba si ha pasado más de 2 horas desde la última consulta a la API REST. Si es así, se vuelve a consultar y la respuesta obtenida se insertará en base de datos y se recuperará de la misma y será actualizado el observador con este valor. Si no, se pondrá el obtenido previamente de la base de datos en el observador.

Estas 2 horas están pensados para no saturar de peticiones al servidor web ya que los cambios en los stocks son tan solo de dos veces al día.

Modelos de vistas

Se ha decidido utilizar MutableLiveData en vez de LiveData, ya que es más sencillo modificar los valores y además, ésta hereda de LiveData.

DonationsViewModel

Es el encargado de obtener los datos del repositorio de las ciudades, los puntos de donación y los eventos de donación.

FixedDonationsViewModel

Es el encargado de obtener los datos del repositorio de las ciudades, los puntos de donación (fijos) y los eventos fijos de donación.

NewsViewModel

Es el encargado de obtener los datos del repositorio de las noticias.

StockViewModel

Es el encargado de obtener los datos del repositorio de los stocks.

Actividades y fragmentos

Main Activity

Encargada de mostrar los datos del stock de hemocomponentes y un listado de las donaciones (filtradas si se ha activado la ubicación o todas en caso contrario). Esta actividad observará el **StockViewModel** y el **DonationsViewModel**. Si se hace clic en cualquier evento de este listado, se abrirá la actividad **SingleLocationActivity** con los datos de este evento de donación en el mapa.

SingleLocationActivity

Encargada de mostrar un mapa con los datos del evento y punto de donación y ciudad seleccionada en la anterior Actividad. El indicador será de color rojo si está activo y azul si no lo está. Sin pulsar el indicador aparecerán todos los datos del evento (lugar, localidad, día y hora de inicio y finalizacion).

NewsActivity

Encargada de mostrar un listado de noticias con la imagen de cabecera, el título y el subtítulo de la noticia. Al hacer clic en alguna de estas noticias, se abrirá la **SingleNewsActivity** con todos los datos de ésta.

SingleNewsActivity

Encargada de mostrar todos los datos de la noticia que ha sido seleccionada. Contiene una imagen de cabecera, el título, el subtítulo y el cuerpo de la noticia en HTML. El cuerpo de la noticia interpretará el formato del texto dado en la aplicación web como también mostrará las imágenes que se hayan introducido en ese HTML.

InfoActivity

Encargada de mostrar información del Banco de Sangre de Extremadura, como es:

- Información sobre la donación de sangre
- Requisitos para donar
- Recomendaciones para el momento después de donar.

También se ha añadido un botón para que el usuario pueda enviar a un correo facilitado para ello por parte del Banco de Sangre de Extremadura.

MapsActivity

Encargada de mostrar un mapa de Extremadura en el que habrá marcadores con todas las donaciones en puntos móviles y en puntos fijos. Se distinguirá por color si ese evento está activo (rojo) o cancelado (azul). En este caso se ha decidido utilizar la API de Google Maps ya que es la más sencilla para usar en Android.

SettingsActivity

Encargada de mostrar los ajustes de la aplicación, en concreto:

- **Grupo sanguíneo**: grupo sanguíneo del paciente. Es voluntario introducirlo y será utilizado para filtrar las notificaciones. Estos datos permanecen en la aplicación y no son mandados al servidor.
- **Usar ubicación automática**: se utilizará la ubicación, utilizando el método que esté disponible (por triangulación de las antenas de telefonía, por el punto Wi-Fi al que esté conectado el dispositivo, por el GPS) que será seleccionada por la Fused Location Provider API. La configuración está puesta para que sea actualizada diariamente para no drenar la batería del usuario.
- **Usar ubicación manual**: podrá seleccionar en un listado. Esta lista está formada por todas las ciudades obtenidas del **DonationsViewModel**.

Además se podrá seleccionar las notificaciones que se quieren recibir. Por defecto están activadas las de *necesidad de tu grupo sanguíneo* y *urgencias*.

- **Extracciones cercanas**: si se ha seleccionado una ciudad mediante la ubicación manual o está activada la ubicación automática se podrá activar esta opción. Enviará una notificación al usuario si el punto de extracción está a menos de 15 Km de la ubicación del usuario y hay un evento en el día actual, en el día siguiente o a en una semana. Si el evento ha sido cancelado, también lo advertirá.
- **Necesidad de tu grupo sanguíneo**: se enviará una notificación si el stock de uno o varios grupos sanguíneos están por debajo del 10 % o 100 unidades.

Si se ha seleccionado un grupo sanguíneo, solo aparecerán las notificaciones pertenecientes a ese grupo sanguíneo.

- **Urgencias**: se enviará una notificación al usuario si se recibe una emergencia.

- **Noticias**: se enviará una notificación al usuario cuando se publique una nueva noticia.

ListDonationsFragment

Fragmento destinado a mostrar la lista de donaciones móviles disponibles (filtradas o no, dependiendo de los ajustes) y que podrá ser reutilizado.

ListNewsFragment

Fragmento destinado a mostrar la lista de noticias disponibles y que podrá ser reutilizado.

Diagrama de actividades

En el siguiente figura mostraremos lo que puede realizar el usuario interactuando con la interfaz:

Figura 5.11: Diagrama de las actividades de la aplicación Android

Alarmas

Las alarmas son creadas para mandar las notificaciones. Esta es una de las pocas formas viables para llevar a cabo las notificaciones, ya que, uno de los requisitos imprescindibles es la no obtención de los datos de los usuarios y por lo tanto, no tener que elegir a un responsable de la política de protección de datos de la aplicación.

En herramientas como **Firebase Cloud Messaging** es necesario nombrar a este responsable, al igual que con **Pusher Beams**, debido a que es un organismo público el

que gestionaría estos datos [53], por lo que hemos decidido obtener los datos activando una alarma de forma periódica del repositorio y si hay algún evento notificable de los anteriormente descritos, se notificará.

Esta no es la mejor forma de hacerlo, pero, debido a los requisitos impuestos por el cliente, esta es la mejor entre ellas.

Todas las alarmas son *elapsed*, es decir, se activan teniendo en cuenta el tiempo que el dispositivo lleva encendido y no por la hora del reloj. La otra forma de hacerlo sería utilizando *RTC*, que se basa en las horas del reloj, por lo que todas las alarmas se ejecutarán a la vez. Se ha elegido hacerlo *elapsed* para no saturar el servidor y tener picos de peticiones (cosa que sucede con *RTC*) que pueden derivar en una denegación del servicio.

BloodNecessityAlarmReceiver

Alarma activada cada 3 horas (en horario 8:00 - 22:00). Si existe un déficit de uno o varios tipos de sangre se mostrará la siguiente notificación:

> *Se necesitan donantes del grupo {grupo o grupos}. Selecciona la opción "Fijos" en el mapa para ver el punto más cercano a tu localidad para poder realizar la donación.*

Esta alarma tiene una repetición inexacta, es decir, se repite cada tres horas de forma aproximada. Además, si el dispositivo Android esta en modo *Doze*[5], la alarma no será lanzada hasta que salga de este modo. Por último, al tener la opción *ElapsedRealtime* activada, esperará a que haya un conjunto de notificaciones con la misma opción para lanzarlas todas a la vez, ahorrando así batería. Si se hace clic en esta notificación, se lanzará la actividad *MapsActivity*.

EmergencyAlarmReceiver

Alarma activada cada hora. Si existe una emergencia mostrará el título de la emergencia más el cuerpo. Si se hace clic en esta notificación, no pasará nada. Tan

[5] El modo *doze* permite que el dispositivo Android, al no ser usado, entre en un estado de hibernación.

solo desaparecerá si se elimina.

Esta alarma tiene una repetición inexacta, es decir, se repite cada tres horas de forma aproximada. Además, si el dispositivo Android esta en modo *Doze*, la alarma no será lanzada hasta que salga de este modo. Por último, al tener la opción *ElapsedRealtimeWakeup* activada, se lanzará. Esto se ha decidido hacer así debido a su caracter urgente.

Las notificaciones de alarmas lanzadas en horario 8:00 - 22:00 encenderá la luz de notificación y vibrará. En el horario contrario (22:01-7:59) se lanzará la notificación pero no vibrará ni se encenderá el led de notificación.

NearExtractionsAlarmReceiver

Alarma activada cada 3 horas (en horario 8:00 - 22:00). Si existe un evento de donación que su hora de comienzo esté en las próximas tres horas y el evento es en el día actual, el día siguiente o en la semana siguiente se mostrará la siguiente notificación:

> *Hay una donación en {nombre de la localidad} de {nombre del punto de donación} {hoy, mañana o en una semana}*

Esto es si el evento está activo, si por el contrario está cancelado mostrará la siguiente notificación:

> *Había una donación en {nombre de la localidad} de {nombre del punto de donación} {hoy, mañana o en una semana} pero ha sido cancelada*

Esta alarma tiene una repetición inexacta, es decir, se repite cada tres horas de forma aproximada. Además, si el dispositivo Android esta en modo *Doze*, la alarma no será lanzada hasta que salga de este modo. Por último, al tener la opción *ElapsedRealtime* activada, esperará a que haya un conjunto de notificaciones con la misma opción para lanzarlas todas a la vez, ahorrando así batería. Si se hace clic en esta notificación, se lanzará la actividad *MainActivity*.

NewsAlarmReceiver

Alarma activada cada 12 horas (en horario 8:00 - 22:00). Si existe una nueva noticia se lanzará una notificación que contendrá el título y el subtítulo de la noticia. En el caso de haber varias noticias, aparecerá el siguiente texto:

¡Hay {número de noticias nuevas} noticias nuevas!

Esta alarma tiene una repetición inexacta, es decir, se repite cada tres horas de forma aproximada. Además, si el dispositivo Android esta en modo *Doze*, la alarma no será lanzada hasta que salga de este modo. Por último, al tener la opción *ElapsedRealtime* activada, esperará a que haya un conjunto de notificaciones con la misma opción para lanzarlas todas a la vez, ahorrando así batería. Si se hace clic en esta notificación, se lanzará la actividad *NewsActivity*.

Receptores de difusión

BootReceiver

Es un *Broadcast Receiver* (o receptor de difusión) que se encarga de recibir el evento *boot completed*. Este evento se lanza cuando se ha reiniciado el dispositivo o se ha encendido y ha terminado el arranque del sistema operativo.

En este caso, este receptor lo utilizaremos para volver a reactivar las alarmas que envían las notificaciones, ya que si se apaga el dispositivo, las alarmas se desactivan.

Ejemplos de flujo de datos

En los siguientes ejemplos, seguiremos el diagrama de la figura 5.8 para ver el recorrido que realizan los datos desde que se solicitan hasta que se reciben. Comenzaremos por la *activity* y terminaremos con la misma.

Obtener stock

El diagrama de la figura 5.12 puede ser desglosado en los siguientes pasos:

1. El usuario abre la aplicación o vuelve a la **actividad principal**.

2. La actividad principal solicita al **modelo de la vista del stock** los datos del último stock y queda observando la respuesta.

3. El modelo de la vista solicita al **repositorio de stocks** los datos y observa la respuesta.

4. El repositorio comprueba si tiene los datos actualizados. Si es así le devuelve el último stock de base de datos.

5. En caso contrario, lo solicita a la *API REST*, lo guarda en base de datos.

6. Al obtener los datos del stock de una de las dos fuentes de datos, y lo introduce en la variable observada por el *modelo de la vista del stock*

7. El modelo de la vista del stock, al obtenerlo, lo introduce en la variable observada por la *actividad principal*.

8. La actividad principal lo obtiene de la variable observada y actualiza la vista con el stock obtenido.

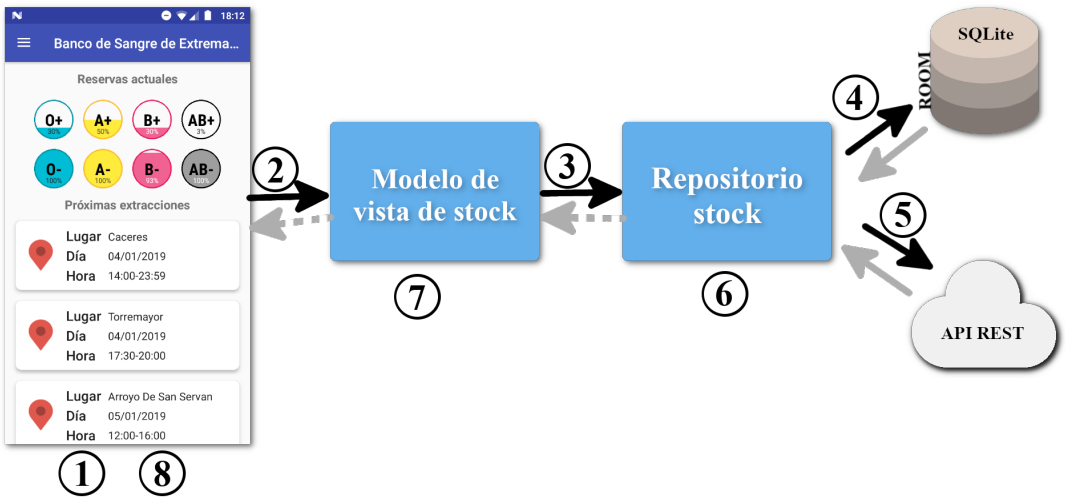


Figura 5.12: Diagrama del flujo de datos del stock

Obtener donaciones

1. El usuario abre la aplicación, vuelve a la **actividad principal** o a la **actividad de mapas**.

2. l,solicita al **modelo de la vista de donaciones** los datos de todas las donaciones o de las cercanas (dependiendo de los filtros) y observa la respuesta.

3. El **modelo de la vista de donaciones** solicita al **repositorio de las ciudades** todas ellas y observa la respuesta.

4. El **repositorio de las ciudades** comprueba que las ciudades están actualizadas. Si lo están, las obtendrá de la base de datos y las introducirá en la variable observada.

5. En caso contrario las solicitará en la *API REST*, las actualizará en base de datos y las introducirá en la variable observada.

6. El **modelo de la vista de las donaciones**, al recibir el conjunto de ciudades, solicita todos los puntos de donación al **repositorio de puntos de donación** y observa la respuesta.

7. El **repositorio de puntos de donación** comprueba que los puntos están actualizados. Si lo están, los obtendrá de la base de datos y los introducirá en la variable observada.

8. En caso contrario las solicitará en la *API REST*, los actualizará en base de datos y los introducirá en la variable observada.

9. El **modelo de la vista de las donaciones**, al recibir el conjunto de puntos de donación, solicita todos los eventos de donación al **repositorio de eventos de donación** y observa la respuesta.

10. El **repositorio de eventos de donación** comprueba que los eventos están actualizados. Si lo están, los obtendrá de la base de datos.

11. En caso contrario las solicitará en la *API REST*, los actualizará en base de datos.

12. El **repositorio de eventos de donación**, enlaza cada evento con su punto de donación, y éste con su ciudad y, a continuación, los introducirá en la variable observada.

13. El **modelo de la vista de donaciones** al tener ya todos los datos, se los pasa a la vista.

14. La **vista** lo obtiene de la variable observada y actualiza la vista con los datos obtenidos.

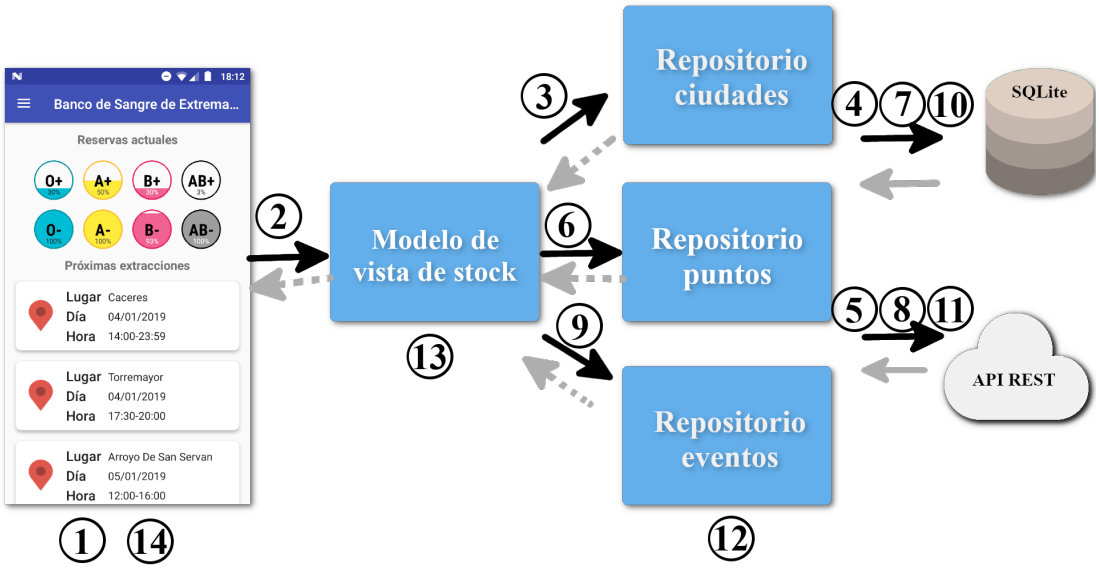


Figura 5.13: Diagrama del flujo de datos de las donaciones

5.3. Pruebas realizadas

5.3.1. Aplicación web corporativa

Para comprobar que la aplicación web va a tener un correcto funcionamiento, se ha realizado la siguiente batería de pruebas:

Vistas

- **Login**: se ha comprobado que es posible acceder a la página, recibiendo un código de estado 200; que, tras haber introducido usuario y contraseña correcto, se autentica, y es redirigido a la página **Home**; que ya habiendo sido autenticado anteriormente y accediendo a esta página, es redirigido a **Home** sin necesidad de volver a introducir las credenciales; que al intentar acceder con unas credenciales incorrectas, no se redirige a la página **Home** y por último, que la plantilla utilizada es *login.html*.

- **Home**: se ha comprobado que si se intenta acceder a esta página sin haberse autenticado, es redirigido automáticamente a la página de **Login**; que si está autenticado sí que es posible acceder a esta página; que si existen datos de stock y programación de colectas, éstos aparecen tanto en los respectivos gráficos como en el mapa (respectivamente) y que la plantilla utilizada es *home.html* y *body.html*

- **Stock Hemocomponentes**: se ha comprobado que el usuario no autenticado no tiene acceso a esta página y es redirigido a la página de **Login**; que si está autenticado sí que es posible acceder a esta página; que si no tiene el permiso "*view_stocks*" asignados para poder acceder a esta página, no podrá acceder y aparecerá un código de error 403 (Prohibido) y que se han cargado las plantillas correspondientes a esta vista *stock/stock.html* y *body.html*.

- **Añadir nuevo registro**: se ha comprobado que el usuario no autenticado no tiene acceso a esa página y es redirigido a la página de **Login**; que si está autenticado sí que es posible acceder a esta página; que si no tiene el permiso "*add_stocks*" no podrá acceder a esta página, apareciendo un código de error 403 (Prohibido); que se han cargado las plantillas correspondientes a esta vista *stock/add_stock.html* y *body.html*; que al añadir los datos de un stock de forma correcta, somos redirigidos a la página **Stock Hemocomponentes** y que la base de datos ahora contiene un nuevo stock y que en caso contrario, no se guardará

ningún stock ni será redirigido a la página **Stock Hemocomponentes**.

- **Ver Stock**: se ha comprobado que el usuario no autenticado no tiene acceso a esa página y es redirigido a la página de **Login**; que si está autenticado sí que es posible acceder a esta página; que si el stock no existe, no se accederá a la página, apareciendo un código de error 404 (No encontrado); que si pulsamos en el botón de imprimir, generará un PDF con los datos de ese stock; que si el usuario no tiene el permiso "*review_stock*" no podrá acceder a esta página, apareciendo un código de error 403 (Prohibido) y que se han utilizado las plantillas *stock/review_stock.html* y *body.html*.

- **Modificar Stock**: se ha comprobado que el usuario no autenticado no tiene acceso a esa página y es redirigido a la página de **Login**; que si está autenticado sí que es posible acceder a esta página; que si el stock no existe, no se accederá a la página, apareciendo un código de error 404 (No encontrado); que si el stock ha expirado, no podrá ser modificado, apareciendo un código de error 404; que si el usuario no tiene el permiso "*modify_stock*" no podrá acceder a esta página, apareciendo un código de error 403 (Prohibido); que al ser modificado introduciendo todos los datos correctamente, se ha actualizado el stock y se ha redirigido a la página **Stock Hemocomponentes**; que en el caso de introducir de forma incorrecta los valores del stock, no se actualizará el stock ni se redirigirá a la página de **Stock Hemocomponentes** y que se han utilizado las plantillas *stock/modify_stock.html* y *body.html*.

- **Programación**: se ha comprobado que el usuario no autenticado no tiene acceso a esa página y es redirigido a la página de **Login**; que si está autenticado sí que es posible acceder a esta página; que si no tiene el permiso "*view_collects*" no podrá acceder a esta página, apareciendo un código de error 403 (Prohibido) y que se han utilizado las plantillas *schedule/schedule.html* y *body.html*.

- **Añadir evento de donación I**: se ha comprobado que el usuario no autenticado no tiene acceso a esa página y es redirigido a la página de **Login**; que

si está autenticado sí que es posible acceder a esta página; que si no tiene el permiso "*add_collects*" no podrá acceder a esta página, apareciendo un código de error 403 (Prohibido); que se han utilizado las plantillas *schedule/add_collect_step1.html* y *body.html*; que si se pulsa el botón *Siguiente* habiendo introducido los datos correctos, avanzará a la siguiente página (Añadir evento de donación II) y que en caso contrario, no se avanzará a la siguiente página.

- **Añadir evento de donación II**: se ha comprobado que el usuario no autenticado no tiene acceso a esa página y es redirigido a la página de **Login**; que si está autenticado sí que es posible acceder a esta página; que si no tiene el permiso "*add_collects*" no podrá acceder a esta página, apareciendo un código de error 403 (Prohibido); que se han utilizado las plantillas *schedule/add_collect_step2.html* y *body.html* y que si se pulsa el botón *Siguiente* habiendo introducido los datos correctos, avanzará a la siguiente página (Añadir evento de donación III) y que en caso contrario, no se avanzará a la siguiente página.

- **Añadir evento de donación III**: se ha comprobado que el usuario no autenticado no tiene acceso a esa página y es redirigido a la página de **Login**; que si está autenticado sí que es posible acceder a esta página; que si no tiene el permiso "*add_collects*" no podrá acceder a esta página, apareciendo un código de error 403 (Prohibido); que se han utilizado las plantillas *schedule/add_collect_step3.html* y *body.html*; que si se pulsa el botón *Guardado* habiendo introducido los datos correctos, redirigirá a la página **Programación** y que en caso contrario, no se avanzará a la siguiente página y mostrará los errores por los que no se puede guardar el evento.

- **Reactivar colectas canceladas**: se ha comprobado que el usuario no autenticado no tiene acceso a esa página y es redirigido a la página de **Login**; que si está autenticado sí que es posible acceder a esta página; que si no

tiene el permiso "*view_cancelled_collects*" no podrá acceder a esta página, apareciendo un código de error 403 (Prohibido); que se han utilizado las plantillas *schedule/cancelled_collections.html* y *body.html* y habiendo cancelado previamente una colecta, aparece en el listado de colectas canceladas.

- **Reactivar colecta cancelada**: se ha comprobado que el usuario no autenticado no tiene acceso a esa página y es redirigido a la página de **Login**; que si está autenticado sí que es posible acceder a esta página; que si no tiene el permiso "*recover_collects*" no podrá acceder a esta página, apareciendo un código de error 403 (Prohibido); que si la colecta no existe o no está cancelada, no se accederá a la página, apareciendo un código de error 404 (No encontrado) y que se han utilizado las plantillas *schedule/recover_collects.html* y *body.html*.

- **Actualizar puntos de colecta**: se ha comprobado que el usuario no autenticado no tiene acceso a esa página y es redirigido a la página de **Login**; que si está autenticado sí que es posible acceder a esta página; que si no tiene el permiso "*update_donation_points*" no podrá acceder a esta página, apareciendo un código de error 403 (Prohibido) y que se han utilizado las plantillas *update_donation_points.html* y *body.html*.

- **Resultados de la colecta**: se ha comprobado que el usuario no autenticado no tiene acceso a esa página y es redirigido a la página de **Login**; que si está autenticado sí que es posible acceder a esta página; que si no tiene el permiso "*view_collect_result*" no podrá acceder a esta página, apareciendo un código de error 403 (Prohibido); que se han utilizado las plantillas *schedule/results_collections.html* y *body.html* y que si previamente hemos insertado los resultados, al acceder aparecen.

- **Ver resultado de la colecta**: se ha comprobado que el usuario no autenticado no tiene acceso a esa página y es redirigido a la página de **Login**; que si está autenticado sí que es posible acceder a esta página; que si no tiene el permiso "*review_collect_result*" no podrá acceder a esta página, apareciendo

un código de error 403 (Prohibido); que se han utilizado las plantillas *schedule/review_result_collect.html* y *body.html* y que si la colecta no tiene ningún resultado o no existe, no se mostrará la página sino que aparecerá un error 404 (No encontrado).

- **Añadir resultado de la colecta**: se ha comprobado que el usuario no autenticado no tiene acceso a esa página y es redirigido a la página de **Login**; que si está autenticado sí que es posible acceder a esta página; que si no tiene el permiso "*add_collect_result*" no podrá acceder a esta página, apareciendo un código de error 403 (Prohibido); que se han utilizado las plantillas *schedule/add_result_collect.html* y *body.html*; que si la colecta no existe, no se mostrará la página sino que aparecerá un error 404 (No encontrado); que si los datos no se escriben correctamente no será redirigido a la página **Resultados de la colecta** ni se guardarán los datos y que, si por el contrario todos los datos son correctos y se pulsa el botón *Guardar*, se guardarán los datos de la colecta y se redirigirá a la página **Resultados de la colecta**.

- **Modificar resultado de la colecta**: se ha comprobado que el usuario no autenticado no tiene acceso a esa página y es redirigido a la página de **Login**; que si está autenticado sí que es posible acceder a esta página; que si no tiene el permiso "*modify_collect_result*" no podrá acceder a esta página, apareciendo un código de error 403 (Prohibido); que se han utilizado las plantillas *schedule/modify_result_collect.html* y *body.html*; que si la colecta no existe o no ha sido insertado el resultado de la colecta, no se mostrará la página sino que aparecerá un error 404 (No encontrado); que si los datos no se escriben correctamente no será redirigido a la página **Resultados de la colecta** ni se actualizarán los datos y que, si por el contrario todos los datos son correctos y se pulsa el botón *Actualizar*, se guardarán los datos de la colecta y se redirigirá a la página **Resultados de la colecta**.

- **Modificar evento de donación**: se ha comprobado que el usuario no autenticado

no tiene acceso a esa página y es redirigido a la página de **Login**; que si está autenticado sí que es posible acceder a esta página; que si no tiene el permiso "*modify_collects*" no podrá acceder a esta página, apareciendo un código de error 403 (Prohibido); que se han utilizado las plantillas *schedule/modify_collect.html* y *body.html*; que si se pulsa el botón *Actualizar* habiendo introducido los datos correctos, redirigirá a la página **Programación** y que en caso contrario, no se redirigirá a la página **Programación** y mostrará los errores por los que no se puede modificar el evento.

- **Cancelar evento de donación**: se ha comprobado que el usuario no autenticado no tiene acceso a esa página y es redirigido a la página de **Login**; que si está autenticado sí que es posible acceder a esta página; que si no tiene el permiso "*cancel_collects*" no podrá acceder a esta página, apareciendo un código de error 403 (Prohibido); que se han utilizado las plantillas *schedule/cancel_collect.html* y *body.html*; que si se pulsa el botón *Cancelar* se redirigirá a la página **Programación** y que en el caso de no existir el evento que quiere ser cancelado, no se accederá a la página, sino que aparecerá un código de error 404 (No encontrado).

- **Eliminar evento de donación**: se ha comprobado que el usuario no autenticado no tiene acceso a esa página y es redirigido a la página de **Login**; que si está autenticado sí que es posible acceder a esta página; que si no tiene el permiso "*delete_collects*" no podrá acceder a esta página, apareciendo un código de error 403 (Prohibido); que se han utilizado las plantillas *schedule/modify_collect.html* y *body.html*; que si se pulsa el botón *Eliminado*, se eliminará el registro y se redirigirá a la página **Programación** y que en el caso de no existir el evento que se quiere eliminar, no se accederá a la página, sino que aparecerá un código de error 404 (No encontrado).

- **Alertas de emergencia**: se ha comprobado que el usuario no autenticado no tiene acceso a esa página y es redirigido a la página de **Login**; que si está

autenticado sí que es posible acceder a esta página; que si no tiene el permiso "*view_alerts*" no podrá acceder a esta página, apareciendo un código de error 403 (Prohibido); que se han utilizado las plantillas *alerts/alerts.html* y *body.html* y que habiendo introducido una alerta previamente, aparece en el listado de alertas.

- **Añadir nueva alerta de emergencia**: se ha comprobado que el usuario no autenticado no tiene acceso a esa página y es redirigido a la página de **Login**; que si está autenticado sí que es posible acceder a esta página; que si no tiene el permiso "*add_alert*" no podrá acceder a esta página, apareciendo un código de error 403 (Prohibido); que se han utilizado las plantillas *alerts/add_alert.html* y *body.html*; que al introducir los datos correctamente para la creación de la alerta y habiendo pulsado el botón *Enviar alerta* se crea la alerta y se redirige a la página **Alertas de emergencia** y que en caso contrario no se creará una alerta ni se redirigirá a la página **Alertas de emergencia**.

- **Noticias**: se ha comprobado que el usuario no autenticado no tiene acceso a esa página y es redirigido a la página de **Login**; que si está autenticado sí que es posible acceder a esta página; que si no tiene el permiso "*view_news*" no podrá acceder a esta página, apareciendo un código de error 403 (Prohibido); que se han utilizado las plantillas *news/news.html* y *body.html* y que habiendo introducido una noticia previamente, aparece en el listado de noticias.

- **Añadir noticia**: se ha comprobado que el usuario no autenticado no tiene acceso a esa página y es redirigido a la página de **Login**; que si está autenticado sí que es posible acceder a esta página; que si no tiene el permiso "*add_news*" no podrá acceder a esta página, apareciendo un código de error 403 (Prohibido); que se han utilizado las plantillas *news/add_news.html* y *body.html*; que al intentar crear una noticia con campos en blanco o incorrectamente rellenados, no se crea la noticia ni se redirige a la página **Noticias**; que al rellenar los campos correctamente y habiendo pulsado el botón *Publicar noticia*, se crea la noticia y se redirigirá a la página **Noticias** y que al rellenar los campos correctamente

y habiendo pulsado el botón *Guardar como borrador*, se crea el borrador y se redirigirá a la página **Noticias**.

- **Borradores**: se ha comprobado que el usuario no autenticado no tiene acceso a esa página y es redirigido a la página de **Login**; que si está autenticado sí que es posible acceder a esta página; que si no tiene el permiso "*view_drafts*" no podrá acceder a esta página, apareciendo un código de error 403 (Prohibido); que se han utilizado las plantillas *news/drafts.html* y *body.html* y que habiendo introducido un borrador previamente, aparece en el listado de borradores.

- **Publicar borrador**: se ha comprobado que el usuario no autenticado no tiene acceso a esa página y es redirigido a la página de **Login**; que si está autenticado sí que es posible acceder a esta página; que si no tiene el permiso "*publish_drafts*" no podrá acceder a esta página, apareciendo un código de error 403 (Prohibido); que se han utilizado las plantillas *news/publish_news.html* y *body.html*; que al intentar publicar una noticia que existe se mostrarán la página con los datos; que si se pulsa el botón *Publicar borrador*, la noticia será publicada y ya no aparecerá en los borradores y que en el caso de no existir o estar publicada la noticia, no se mostrará la página y aparecerá un código de error 404 (No encontrado).

- **Modificar borrador**: se ha comprobado que el usuario no autenticado no tiene acceso a esa página y es redirigido a la página de **Login**; que si está autenticado sí que es posible acceder a esta página; que si no tiene el permiso "*modify_drafts*" no podrá acceder a esta página, apareciendo un código de error 403 (Prohibido); que se han utilizado las plantillas *news/modify_news.html* y *body.html*; que al intentar modificar una noticia con campos en blanco o incorrectamente rellenados, no se modifica la noticia ni se redirige a la página **Noticias**; que al rellenar los campos correctamente y habiendo pulsado el botón *Publicar noticia*, se modifica la el borrador, se publica la noticia y se redirigirá a la página **Noticias**; que al rellenar los campos correctamente y habiendo pulsado el botón *Actualizar borrador*, se modifica el borrador y se redirigirá a la página **Noticias**

y que si se intenta modificar una noticia que no existe o que está publicada, no se mostrará la página y aparecerá un código de error 404 (No encontrado).

- **Eliminar noticia**: se ha comprobado que el usuario no autenticado no tiene acceso a esa página y es redirigido a la página de **Login**; que si está autenticado sí que es posible acceder a esta página; que si no tiene el permiso "*delete_news*" no podrá acceder a esta página, apareciendo un código de error 403 (Prohibido); que se han utilizado las plantillas *news/delete_news.html* y *body.html*; que al intentar eliminar una noticia que existe se mostrarán la página con los datos; que si se pulsa el botón *Eliminar noticia*, la noticia será eliminada y ya no aparecerá en los borradores ni en las noticias y que en el caso de no existir o de haber sido eliminada previamente, no se mostrará la página y aparecerá un código de error 404 (No encontrado).

- **Leer noticia**: se ha comprobado que el usuario no autenticado no tiene acceso a esa página y es redirigido a la página de **Login**; que si está autenticado sí que es posible acceder a esta página; que si no tiene el permiso "*review_news*" no podrá acceder a esta página, apareciendo un código de error 403 (Prohibido); que se han utilizado las plantillas *news/review_news.html* y *body.html*; que al intentar leer una noticia que existe se mostrarán la página con los datos y que en el caso de no existir, no se mostrará la página y aparecerá un código de error 404 (No encontrado).

- **Perfil**: se ha comprobado que el usuario no autenticado no tiene acceso a esa página y es redirigido a la página de **Login**; que si está autenticado sí que es posible acceder a esta página; que se han utilizado las plantillas *profile.html* y *body.html*; que al intentar cambiar la contraseña cumpliendo todas las condiciones impuestas y pulsando el botón *Actualizar contraseña*, ésta se actualiza y que si por el contrario no se cumplen todas las restricciones, no se cambiará la contraseña.

- **Logout**: se ha comprobado que el usuario no autenticado no tiene acceso a esa

página y es redirigido a la página de **Login** y que si está autenticado sí que es posible acceder a esta vista.

Modelos

- **Artículo**: se ha comprobado que la representación de la entidad en una cadena de caracteres es correcta "Título (subtítulo)", que el nombre en singular de la entidad es correct (artículo) y también para el nombre en plural (artículos).

- **Área de salud**: se ha comprobado que la representación de la entidad en una cadena de caracteres es correcta "nombre", que el nombre en singular de la entidad es correcto (área de salud) y también para el nombre en plural (áreas de salud).

- **Ciudad**: se ha comprobado que la representación de la entidad en una cadena de caracteres es correcto "nombre", que el nombre en singular de la entidad es correcta (ciudad) y también para el nombre en plural (ciudades).

- **Punto de donación**: se ha comprobado que la representación de la entidad en una cadena de caracteres es correcta "nombre", que el nombre en singular de la entidad es correcto (punto de donación) y también para el nombre en plural (puntos de donación).

- **Evento de donación**: se ha comprobado que la representación de la entidad en una cadena de caracteres es correcta "nombre del punto de donación (día) hora_comienzo - hora_fin", que el nombre en singular de la entidad es correcto (evento de donación) y también para el nombre en plural (eventos de donación). También se ha comprobado si ha expirado o si no lo ha hecho y si han sido o no añadidos los resultados de la colecta.

- **Evento fijo de donación**: se ha comprobado que la representación de la entidad en una cadena de caracteres es correcta "nombre del punto de donación (día_comienzo - día_fin) hora_comienzo - hora_fin", que el nombre en singular

de la entidad es correcto (evento fijo de donación) y también para el nombre en plural (eventos fijos de donación). También se ha comprobado si ha expirado o si no lo ha hecho y si han sido o no añadidos los resultados de la colecta.

- **Stock**: se ha comprobado que la representación de la entidad en una cadena de caracteres es correcta "Informe: identificador_stock. Nombre_usuario Apellidos_usuario fecha (hora)", que el nombre en singular de la entidad es correcto (informe de stock) y también para el nombre en plural (informes de stock). También se ha comprobado si ha expirado o si no lo ha hecho.

- **Notificación**: se ha comprobado que la representación de la entidad en una cadena de caracteres es correcta "Título: cuerpo", que el nombre en singular de la entidad es correcto (alerta de emergencia) y también para el nombre en plural (alertas de emergencia). También se ha comprobado si ha expirado o si no lo ha hecho.

Habiendo realizado estas pruebas, se han obtenido los siguientes resultados de cobertura, que también pueden ser observados en la figura 5.14:

- **Cobertura total del proyecto**: El proyecto tiene una cobertura del 92 %.

- **Cobertura de las vistas**: Existen 193 tests de las vistas (incluyendo las de la API REST), con una cobertura del 94 %.

- **Cobertura de los modelos**: Existen 32 tests de los modelos, con una cobertura del 100 %.

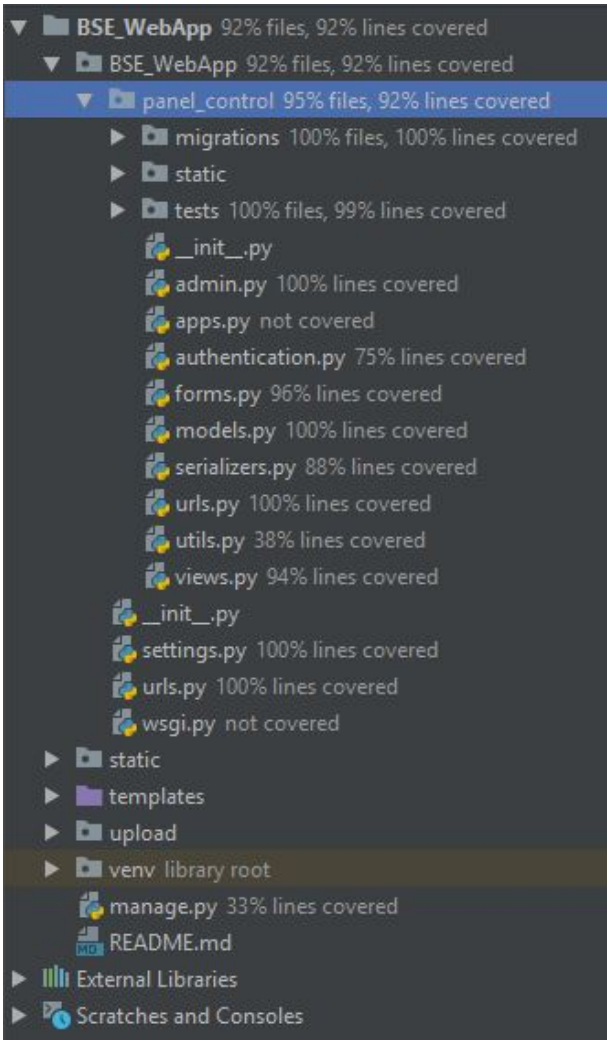


Figura 5.14: Cobertura del proyecto web con Django

La cobertura nos indica el número de líneas de código que han sido abarcadas por los tests, por lo que hay algunas líneas que no son probadas, pero en su mayoría son métodos creados por defecto, como pueden ser muchos de los incluidos en la API REST, que tan solo es necesario indicarles el conjunto de datos y el serializador.

5.3.2. API REST con Django REST Framework

Las pruebas realizadas a la API REST se han enfocado en aquellas que clases a las que le han añadido mayor funcionalidad, ya que heredan de *generics.ListApiView* aquellas que muestran una lista de elementos y de *generics.RetrieveApiView* aquellas

que muestran tan solo uno de ellos.

Concretamente, las clases que han sido probadas son las siguientes:

- **ObtainExpiringAuthTokenAPIViewTest**: Vista encargada de obtener los Token para acceder a los datos. De esta clase se ha probado que el método de autenticación es correcto y funcional.

- **CityListAPIViewTest**: Vista encargada de mostrar el listado de ciudades. De esta clase se ha probado que sólo es posible acceder mediante un Token válido y activo, que se obtienen los datos de las ciudades y que se pueden aplicar los filtros de nombre de la ciudad y nombre de la provincia.

- **DonationPointListAPIViewTest**: Vista encargada de mostrar el listado de puntos de colectas. De esta clase se ha probado que sólo es posible acceder mediante un Token válido y activo, que se obtienen los datos de los puntos de donación y que se pueden aplicar los filtros de nombre de la ciudad, nombre del punto de donación y seleccionando los puntos fijos o móviles.

- **DonationEventListAPIViewTest**: Vista encargada de mostrar el listado de las colectas (o eventos de donación). De esta clase se ha probado que sólo es posible acceder mediante un Token válido y activo, que se obtienen los datos de los eventos de donación y que se pueden aplicar los filtros seleccionando los eventos del día actual, del día siguiente, de un día concreto o de un rango de días.

- **StockDetailAPIViewTest**: Vista encargada de mostrar el último stock. De esta clase se ha probado que sólo es posible acceder mediante un Token válido y activo, que se obtienen los datos del stock si existe alguno y en el caso de no existir alguno, que no muestre ningún dato.

5.3.3. Aplicación Android

Pruebas de aceptación

Para comprobar que la aplicación es útil y funcional para los usuarios, se han realizado pruebas de aceptación de caja negra. Las pruebas han consistido en instalar la aplicación en el terminal Android de los usuarios que la probarán.

Los terminales de los usuarios son los siguientes:

bq Aquaris U Plus

- **Versión de Android**: 7.1.1
- **Memoria RAM**: 3GB
- **Memoria ROM**: 32GB
- **Tamaño de pantalla**: 1280 x 720
- **Densidad de pantalla**: 293 ppp
- **Procesador**: Qualcomm Snapdragon 430

Vodafone Smart Prime 6

- **Versión de Android**: 5.0.2
- **Memoria RAM**: 1GB
- **Memoria ROM**: 8GB
- **Tamaño de pantalla**: 1280 x 720
- **Densidad de pantalla**: 294 ppp
- **Procesador**: Qualcomm MSM8916 Snapdragon 410

Innjoo Halo X 3G

- **Versión de Android**: 6.0.1
- **Memoria RAM**: 2GB
- **Memoria ROM**: 16GB
- **Tamaño de pantalla**: 1280 x 720
- **Densidad de pantalla**: 267 ppp
- **Procesador**: Spreadtrum SC7731G

Los tres usuarios han utilizado todas las funcionalidades sin indicarles el funcionamiento de ninguna de ellas. Con esto hemos asegurado que el usuario es capaz de conseguir utilizarla y obtener un beneficio de ella de forma autónoma.

Los usuarios que han utilizado esta aplicación han comprobado que todos los requisitos expuestos en el apartado del análisis y diseño se han cumplido y se han plasmado en esta aplicación.

Estas pruebas han conseguido solucionar pequeños detalles que para el usuario eran incómodos o poco intuitivos, como por ejemplo, añadir en el apartado de *puntos de extracción* una leyenda indicando el color de las *chinchetas*.

Capítulo 6

Conclusiones y trabajos futuros

6.1. Conclusiones

En líneas generales, se puede decir que ha sido un trabajo laborioso llegar a este resultado final, pero a la vez ha sido igualmente satisfactorio. Se ha proporcionado un sistema completo que contempla todos los requisitos impuestos por el cliente y que en líneas generales es sencillo de utilizar tanto para los empleados del Banco de Sangre de Extremadura como para los usuarios de la aplicación móvil.

Este proyecto se ha basado en un problema real que presenta esta entidad, y en base a ese problema, se ha propuesto esta solución, que será de utilidad para el conjunto de la sociedad extremeña. Su despliegue y uso funcional por parte de esta organización está aún pendiente por parte del Servicio Extremeño de Salud, que tiene que revisar todos los aspectos técnicos de este proyecto.

La aplicación web está desplegada en la dirección web `http://158.49.112.144/` y la aplicación Android puede descargarse desde la siguiente dirección: `https://drive.google.com/file/d/1Qo2hvKJNGXlHG14VCWRzQHiXj6DVuw8s`.

Por último, en la tabla 6.1 vamos a ver una relación con los objetivos que se han completado.

Objetivo	Objetivo cumplido
Deberá existir una aplicación corporativa sencilla que generará los datos y otra aplicación móvil que los consumirá	Sí
Los datos generados serán estructurados	Sí
No se recabará ningún dato del usuario	Sí
Se usará la ubicación para notificar a los usuarios las colectas programadas cercanas a su posición	Sí
Las emergencias serán derivadas a los puntos de extracción fijos	Sí
Los datos generados en la aplicación corporativa deberán ser públicos	Sí
El estado actual de las existencias de hemocomponentes deberá ser en tiempo real o actualizado cada poco tiempo	Sí
Será necesario añadir las colectas programadas en la aplicación corporativa	Sí
En las colectas programadas, se deberá añadir con posterioridad al evento las bolsa conseguidas y el número de nuevos donantes	Sí
Se deberá transformar la plantilla del documento de texto (docx) que contiene el informe de existencias en un PDF con los datos concretos	Sí
No se podrá sobrecargar de funcionalidades la aplicación móvil	Sí
Se podrá modificar el estado de la colecta entre activo y cancelado	Sí
Se deberá diferenciar entre ciudad y punto de donación	Sí
La aplicación móvil deberá hacer uso de los datos proporcionados por la aplicación corporativa del BSE	Sí
El proyecto deberá ser lo más económico posible y desarrollado utilizando herramientas y frameworks de código abierto	Sí
La aplicación corporativa deberá ser sencilla y rápida	Sí

Objetivo	**Objetivo cumplido**
La aplicación corporativa deberá ser segura y robusta	Parcialmente
La interfaz de exposición de datos deberá ser segura	Sí

Tabla 6.1: Relación de objetivos cumplidos

6.2. Trabajos futuros

6.2.1. Creación de un sistema de predicción de lugares de colectas

Uno de los objetivos de este proyecto es la exposición de los datos y su uso de forma pública. Uno de los trabajos futuros sería hacer uso de estos datos, entre otros, para poder desarrollar un sistema de inteligencia artificial que permita predecir las fechas de los lugares de colectas para obtener el máximo número de unidades de sangre de uno o varios grupos sanguíneos que tengan deficiencias en ese momento o se esperen tener.

6.2.2. Mejora de la obtención de Tokens para la API REST

Actualmente solo existe un usuario que hace uso de la API REST, que es utilizado por la aplicación móvil de forma privada. Si un usuario cualquiera que quiera acceder a la API REST, deberá solicitar un usuario y contraseña al administrador del sistema para obtener su API-KEY. Se le facilita un usuario y contraseña, ya que la API-KEY se actualiza cada 4 semanas por seguridad.
Este trabajo futuro consistiría en automatizar este proceso de una forma controlada y sin necesidad de esperar una respuesta del administrador del sistema.

6.2.3. Creación de aplicaciones en otras plataformas de desarrollo

Actualmente tan solo se ha desarrollado la aplicación móvil en la plataforma de desarrollo de Android, pero plataformas como iOS han quedado excluidas. Sería recomendable crear una aplicación para esta plataforma.

6.2.4. Modificación de la aplicación

Actualmente se modifican las áreas de salud, las ciudades, los puntos de donación (fijos y móviles) y los eventos (fijos y móviles) desde una hoja de cálculo Excel por requisito del cliente. En una segunda fase del proyecto se eliminará esta opción y se integrará la creación, modificación y eliminación de estos elementos con el sistema actual, sin necesidad de utilizar ningún fichero externo.

6.2.5. Servidor HTTPS

Actualmente la aplicación web se ha desplegado en un servidor HTTP y no se ha implementado el cifrado de la conexión. Para un entorno de producción, sería recomendable obtener un certificado válido e implementar la conexión HTTPS.

Anexos

Apéndice A

Manual de usuario. Aplicación corporativa

Al acceder a la dirección web de la aplicación proporcionada por el administrador del sistema, por ejemplo *http://bse.ses.es*, aparecerá la página de acceso, que tiene el aspecto de la figura A.1.

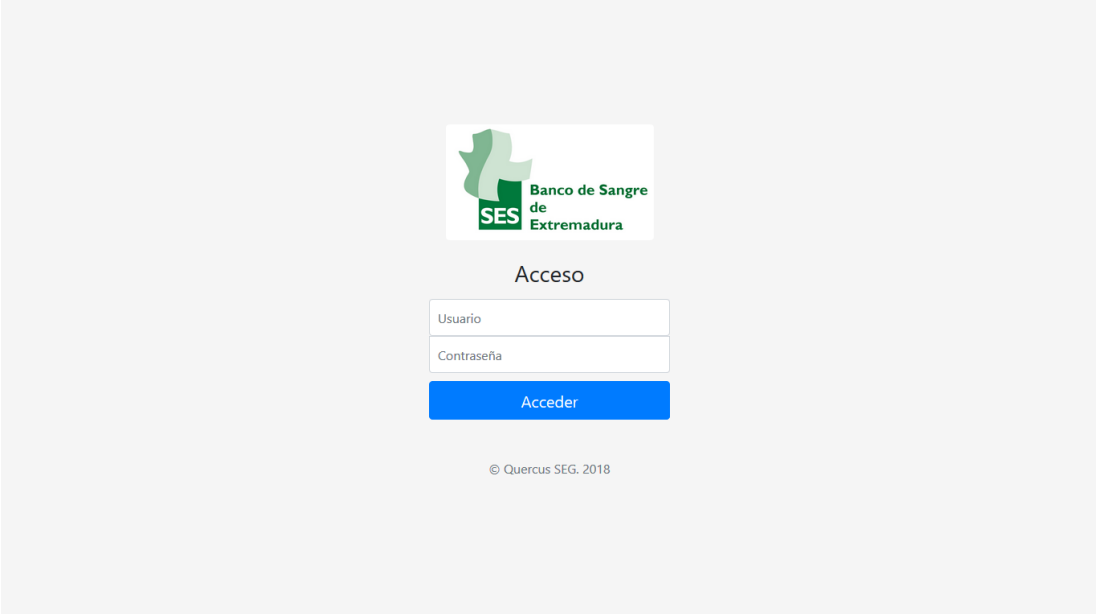


Figura A.1: Aplicación corporativa. Página de acceso

El administrador del sistema, proporcionará al usuario un nombre de usuario y

una contraseña (que deberá ser modificada). El usuario, con estas credenciales podrá acceder a todas las páginas a las que tiene permisos concedidos.
Una vez introducidos los datos de acceso y pulsado el botón **Acceder**, siendo esas credenciales correctas, el usuario será redirigido a la página de Home o **Inicio**. Esta página tiene el aspecto de la figura A.2.

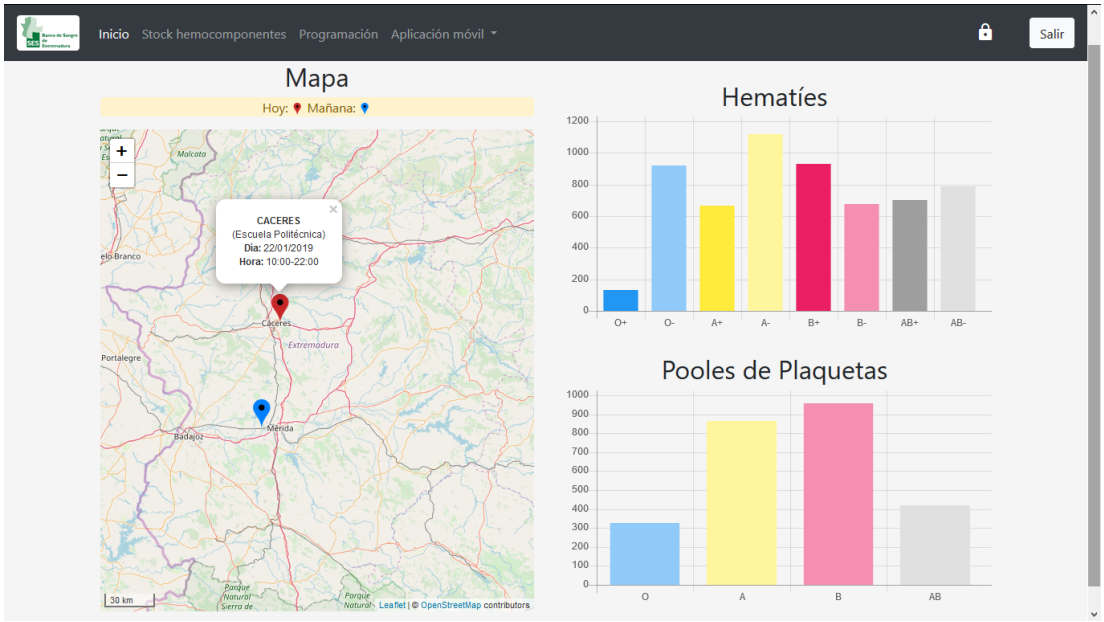


Figura A.2: Aplicación corporativa. Página de inicio

En esta página de inicio, podemos observar un mapa, con las colectas que tendrán lugar el día actual y el día siguiente. Incluye un gráfico con los hematíes divididos por grupo sanguíneo (incluyendo el Rh). También incluye un gráfico con los pooles de plaquetas divididos por grupo sanguíneo (sin incluir el Rh).
Los colores de las barras de ambos gráficos son los colores estándar para diferenciar los grupos sanguíneos.

Si se tienen los permisos necesarios, aparecerá en la barra de navegación superior un apartado denominado **Stock Hemocomponentes**. Desde aquí podremos añadir, modificar, revisar e imprimir los informes de existencias de hemocomponentes.
Al pulsar en este apartado, aparecerá un listado con todos los stocks creados hasta la fecha. En la figura A.3 podemos observar este listado.

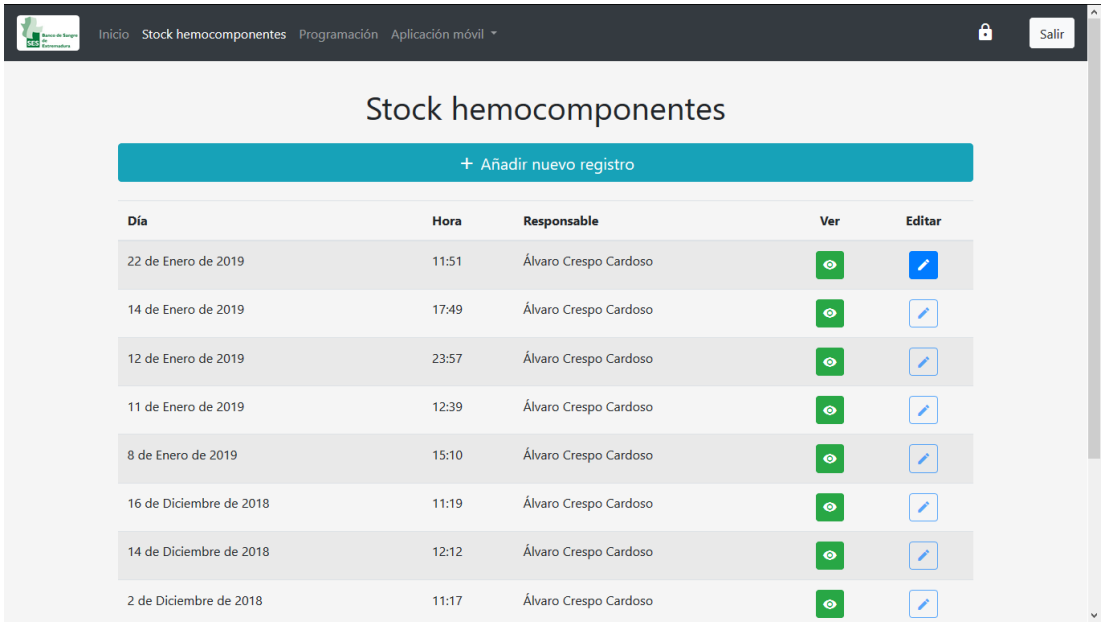


Figura A.3: Aplicación corporativa. Página de Stock Hemocomponentes

Si se tienen los permisos necesarios, podremos añadir un nuevo registro. Para ello deberemos rellenar cada uno de los campos tal y como se hacía anteriormente en el documento de la figura A.5. En la figura A.4 se pueden observar los campos a rellenar. Por último tan solo deberemos hacer clic en el botón **Guardar**.

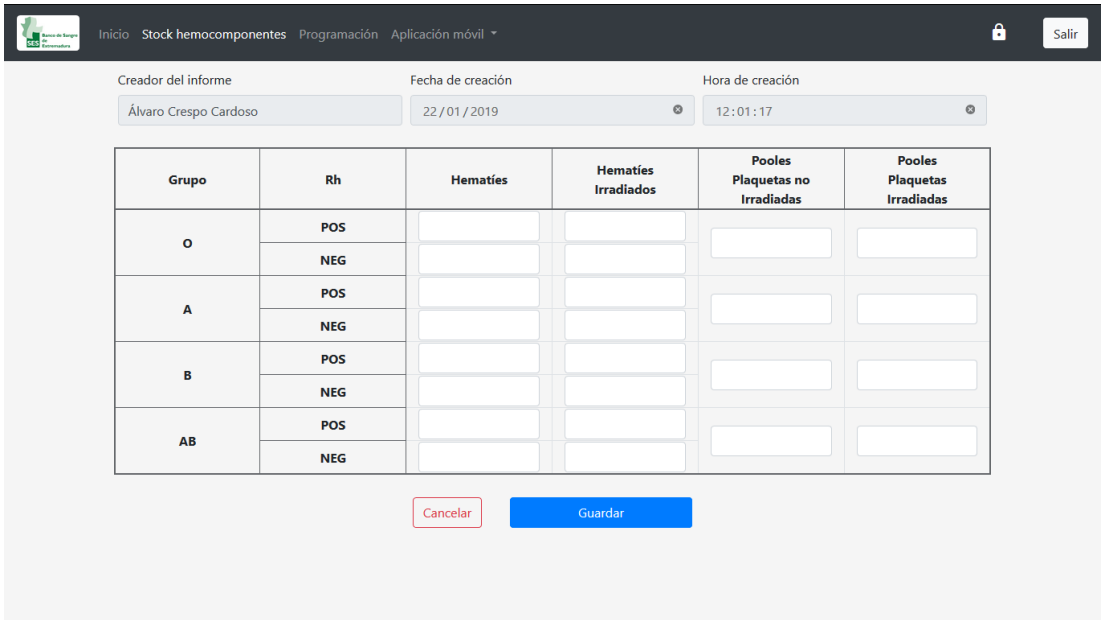


Figura A.4: Aplicación corporativa. Añadir Stock

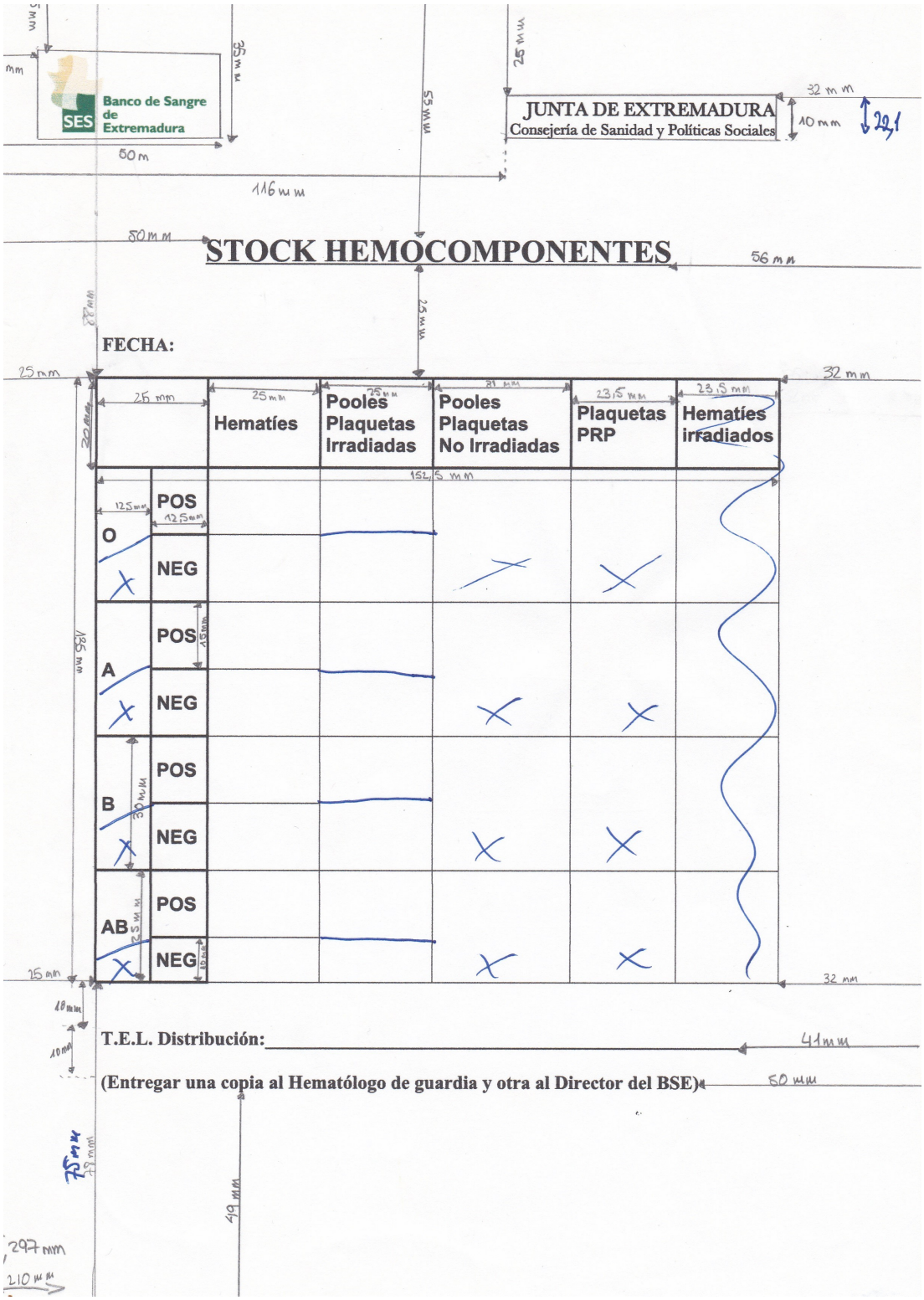
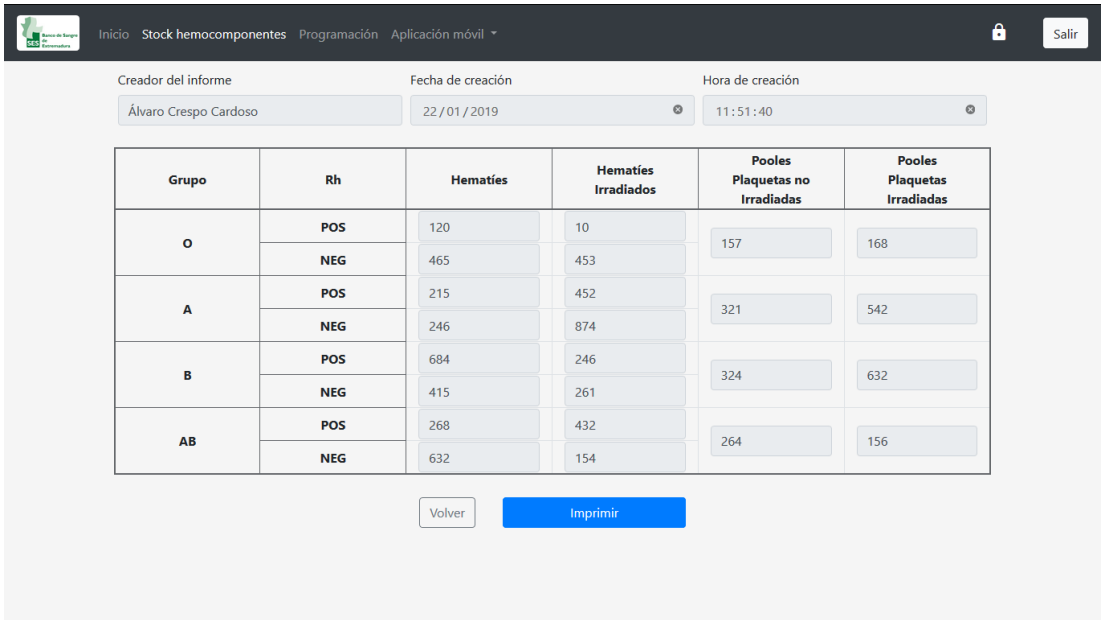


Figura A.5: Aplicación corporativa. Documento original de Stock

Una vez se ha guardado el stock podremos ver en detalle el stock, imprimirlo y editarlo. Para verlo, pulsaremos el botón verde con el icono de un ojo en blanco que se puede observar en la figura A.3. La página que aparecerá es la que está representada en la figura A.6.

Aparecerán los datos del stock sin posibilidad de ser modificados y en la parte inferior tenemos dos botones, **Volver** para ir a la página con el listado e **Imprimir** para obtener el documento generado con los datos del stock.



Grupo	Rh	Hematíes	Hematíes Irradiados	Pooles Plaquetas no Irradiadas	Pooles Plaquetas Irradiadas
O	POS	120	10	157	168
	NEG	465	453		
A	POS	215	452	321	542
	NEG	246	874		
B	POS	684	246	324	632
	NEG	415	261		
AB	POS	268	432	264	156
	NEG	632	154		

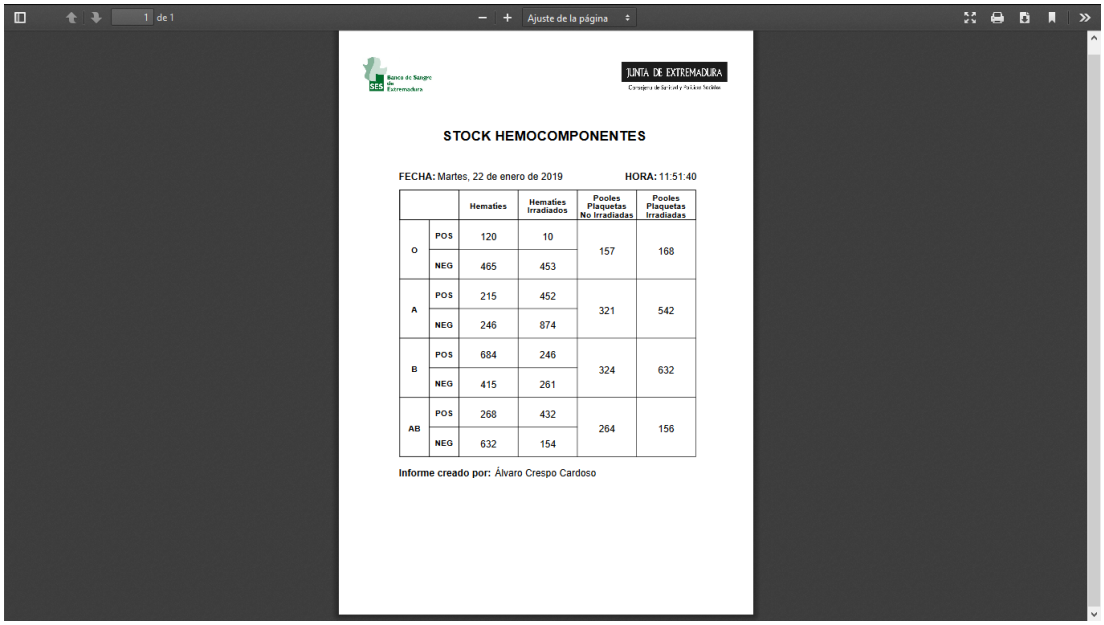
Figura A.6: Aplicación corporativa. Review Stock

Al pulsar el botón imprimir aparecerá el documento PDF de la figura A.7 con la posibilidad de ser imprimido.

La otra opción posible es la de modificar el stock. Para ello, no podrá pasar más de **tres días**, ya que si ha pasado este tiempo el stock no podrá ser modificado. Esto se podrá observar a simple vista en la figura A.3.

Si es posible, se permitirán modificar los valores y se actualizarán pulsando el botón **Actualizar**. En la figura A.8 podemos observar esta página.

Con estas páginas, ya habríamos terminado de ver todo lo relacionado con los stocks de hemocomponentes. Ahora pasaremos a las páginas dedicadas a la **programación de colectas**.



JUNTA DE EXTREMADURA

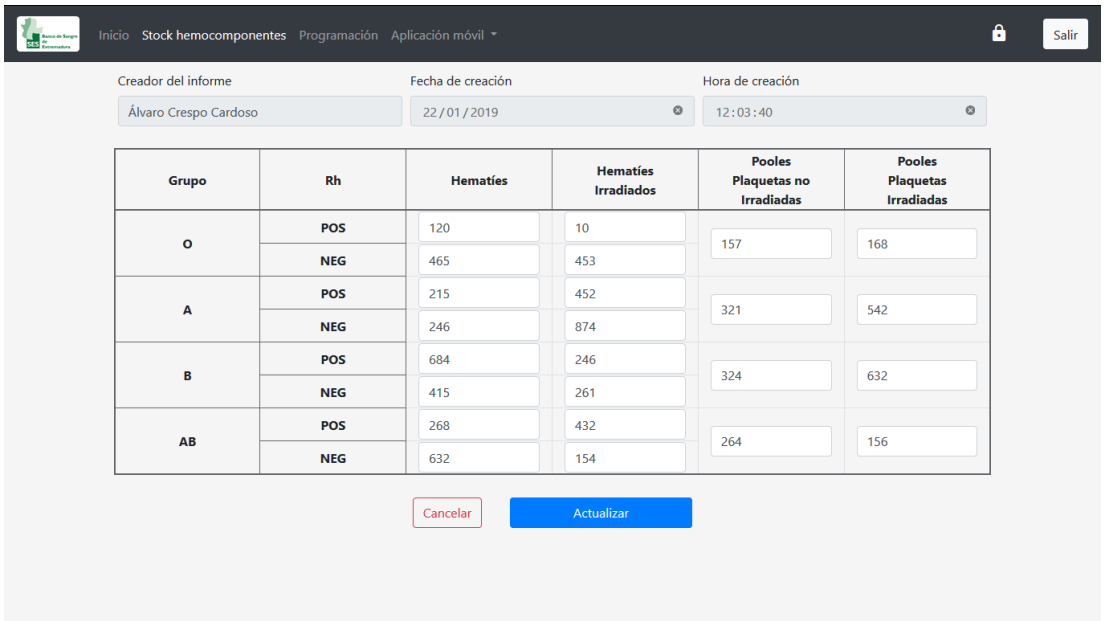
STOCK HEMOCOMPONENTES

FECHA: Martes, 22 de enero de 2019 HORA: 11:51:40

		Hematíes	Hematíes Irradiados	Pooles Plaquetas No Irradiadas	Pooles Plaquetas Irradiadas
O	POS	120	10	157	168
	NEG	465	453		
A	POS	215	452	321	542
	NEG	246	874		
B	POS	684	246	324	632
	NEG	415	261		
AB	POS	268	432	264	156
	NEG	632	154		

Informe creado por: Álvaro Crespo Cardoso

Figura A.7: Aplicación corporativa. Imprimir Stock



Inicio Stock hemocomponentes Programación Aplicación móvil Salir

Creador del informe: Álvaro Crespo Cardoso

Fecha de creación: 22/01/2019

Hora de creación: 12:03:40

Grupo	Rh	Hematíes	Hematíes Irradiados	Pooles Plaquetas no Irradiadas	Pooles Plaquetas Irradiadas
O	POS	120	10	157	168
	NEG	465	453		
A	POS	215	452	321	542
	NEG	246	874		
B	POS	684	246	324	632
	NEG	415	261		
AB	POS	268	432	264	156
	NEG	632	154		

Cancelar Actualizar

Figura A.8: Aplicación corporativa. Modificar Stock

La página principal que incluye el listado con la programación de las colectas se accede haciendo clic en el menú superior en la sección **Programación**. Nos cargará la página de la figura A.9 en la que aparecerá un listado con todas las colectas programadas activas y cuyo día es igual o superior al actual.

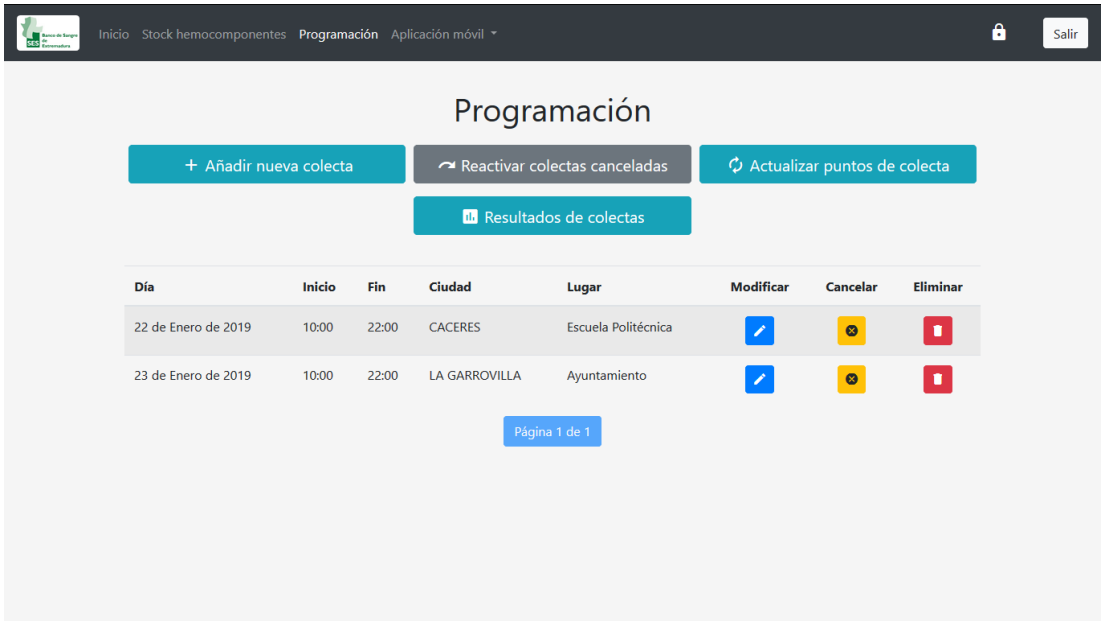


Figura A.9: Aplicación corporativa. Programación de colectas

Como podemos observar en la figura A.9, hay diversas opciones, como son añadir nuevas colectas, reactivar colectas canceladas, actualizar los puntos de colecta, ver, añadir o modificar los resultados de las colectas. Además se podrán modificar, cancelar y eliminar colectas individuales.

Primeramente, vamos a observar cómo añadiremos una colecta. Para ello pulsaremos el botón **Añadir nueva colecta**. En la página que se cargará (figura A.10) aparecerá una lista con las áreas de salud de Extremadura, para continuar el proceso, pulsaremos el botón **Siguiente**.

Si hemos seleccionado un área de salud, a continuación nos aparecerán todas las ciudades que pertenecen al área de salud seleccionada en la anterior página. En la figura A.11 podemos ver esta página.

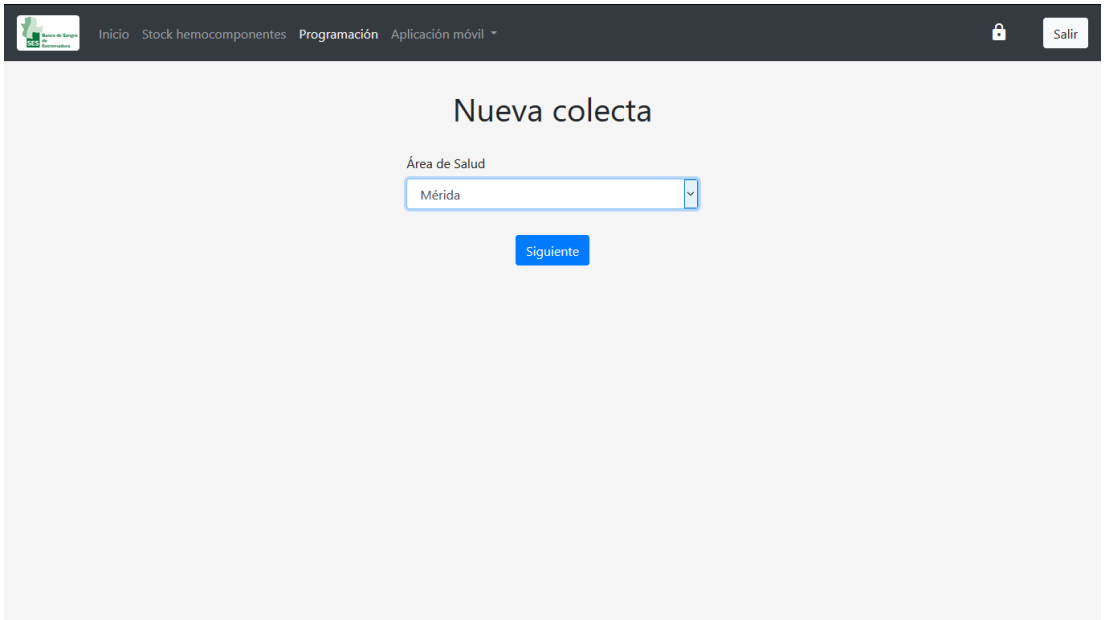


Figura A.10: Aplicación corporativa. Añadir colecta, paso 1

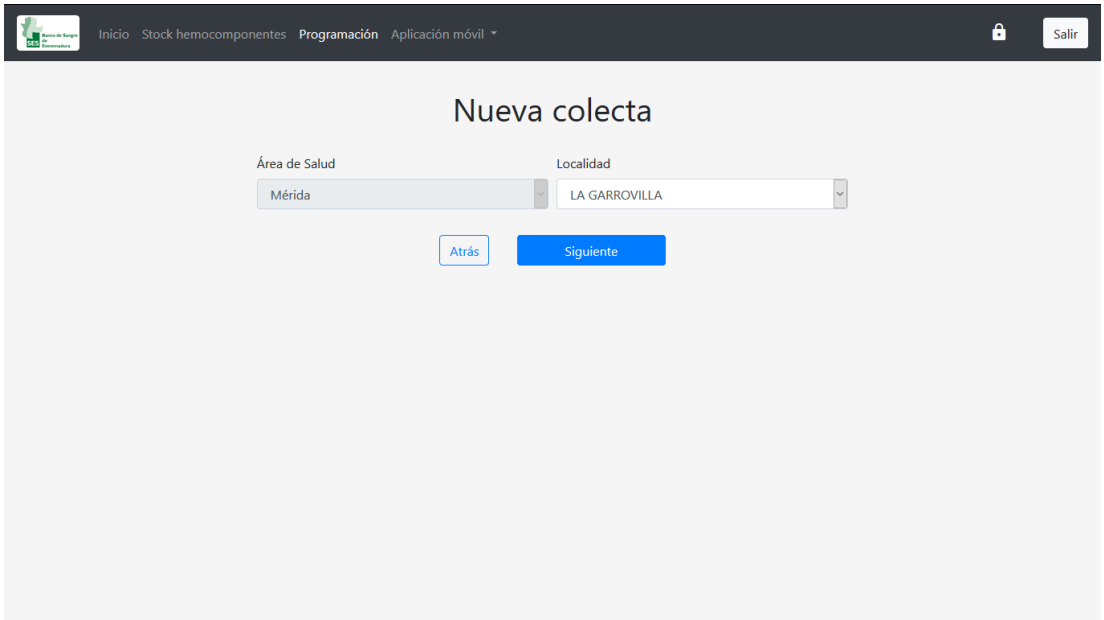


Figura A.11: Aplicación corporativa. Añadir colecta, paso 2

Tras seleccionar una ciudad y haber pulsado el botón **Siguiente**, aparecerá una lista con los puntos de extracción pertenecientes a esa ciudad, la fecha de la donación y las horas de comienzo y finalización de la colecta. Una vez se han completado todos los campos correctamente, se hace clic en el botón **Guardar** y se creará el evento de

donación. Esta página se puede observar en la figura A.12.

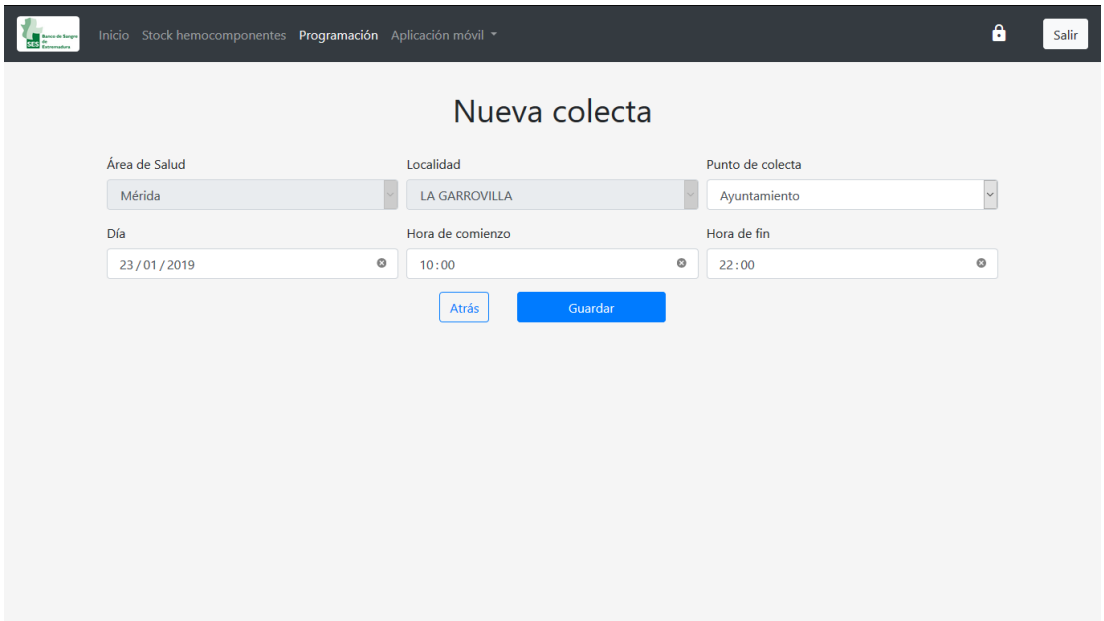


Figura A.12: Aplicación corporativa. Añadir colecta, paso 3

Una vez realizado este proceso, el evento ya aparecerá en la lista de las colectas de la página de programación. Si durante el proceso de añadir una colecta detectamos que el punto de colecta no aparece en la lista, deberemos actualizar los puntos de colecta. Para ello, en la página de programación existe un botón denominado **Actualizar puntos de colecta**.

Al pulsar sobre él deberemos utilizar el archivo Excel para actualizar estos puntos. En este archivo Excel se deberá añadir el punto o puntos de colecta que falten con el formato especificado y una vez actualizado, guardarlo. Se seleccionará el archivo Excel y se pulsará el botón **Actualizar**.

Este proceso puede tomar un tiempo máximo de 15 minutos para la primera instalación y un mínimo de 20 segundos en el caso de cambiar o añadir tan solo una fila. La mayor parte del tiempo gastado en este proceso, es debido a las restricciones del servicio Open Street Maps Nominatim.

En la figura A.13 podemos observar la página de actualización de los puntos de colecta.

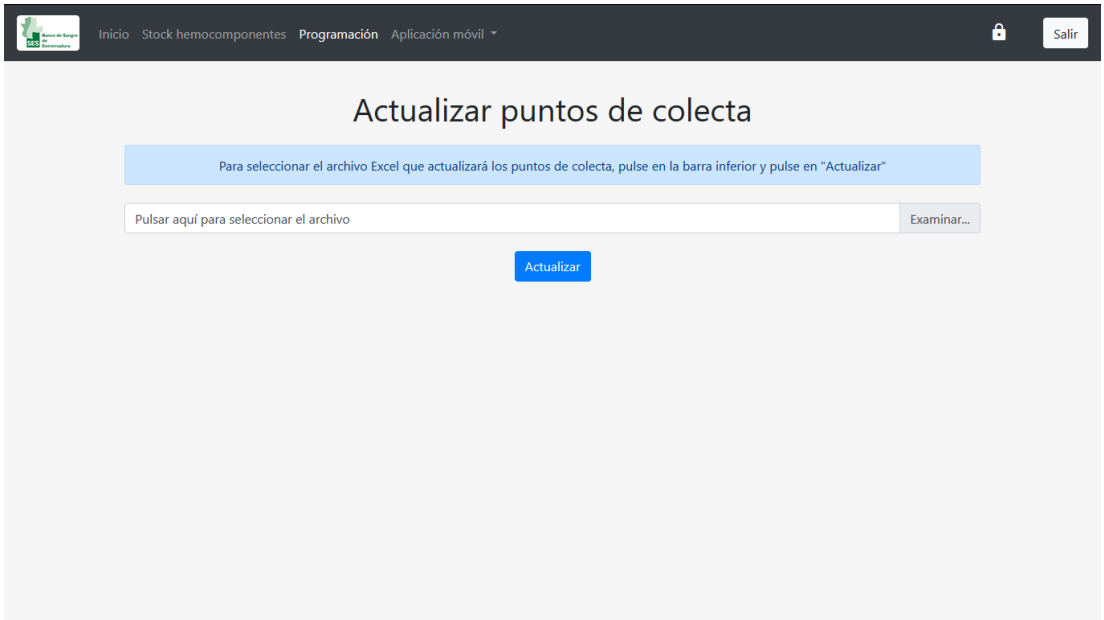


Figura A.13: Aplicación corporativa. Actualizar puntos de colecta

Una vez actualizados los puntos de colecta, ya podremos añadir la colecta sin problema alguno. Una vez añadidas las colectas, y como comentamos anteriormente, se podrán modificar, cancelar y eliminar.

Para modificar la colecta deberemos pulsar el botón de color azul con un lápiz blanco, como puede observarse en la figura A.9. Hay que tener en cuenta que si nos hemos equivocado a la hora de seleccionar el punto de colecta, deberemos eliminar esta colecta, ya que en la opción de modificar tan solo se podrán modificar los datos referentes a la fecha y horas del evento, como puede observarse en la figura A.14

Otra opción es cancelar la colecta, que no la elimina, sino que informará a los usuarios de la aplicación móvil que no es necesario acudir a esa colecta, ya que no va a ser celebrada. Para ello, deberemos pulsar el botón amarillo con una **x** dentro de un círculo y nos cargará la página encargada de cancelar el evento. Para confirmar la cancelación del evento deberemos pulsar el botón **Cancelar colecta**. Esta página puede verse en la figura A.15

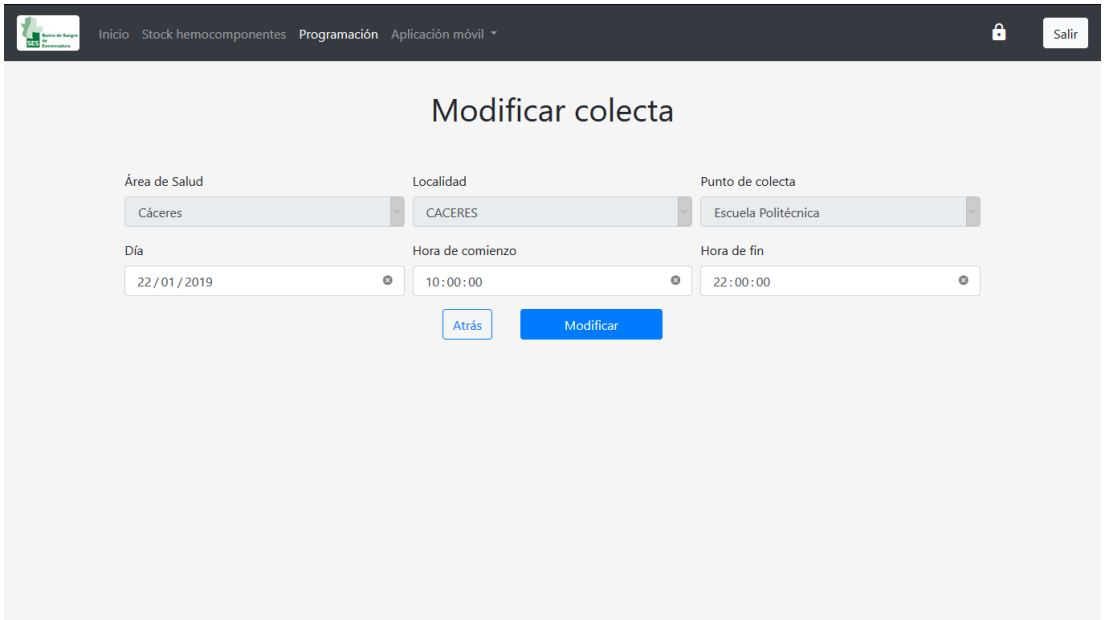


Figura A.14: Aplicación corporativa. Modificar colectas programadas

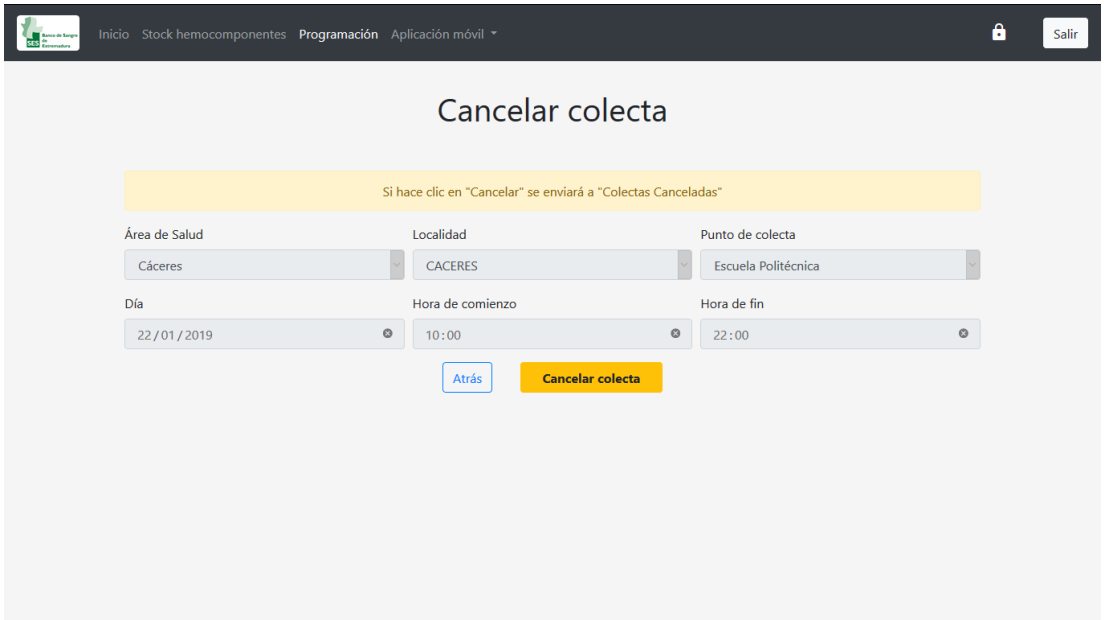


Figura A.15: Aplicación corporativa. Cancelar colectas programadas

Todas aquellas colectas que hayan sido canceladas, desaparecerán de la página principal de las colectas y pasarán a estar en el apartado de **Recuperar colectas canceladas**, que veremos posteriormente.

Por último, podremos eliminar las colectas. Para ello pulsaremos el botón rojo

con un cubo de basura blanco, como podemos observar en la figura A.9. Al pulsar este botón nos mostrará los datos de la colecta que se quiere eliminar. Si queremos eliminarla de forma definitiva, deberemos pulsar el botón **Eliminar**, como podemos ver en la figura A.16. Es preciso recordar la diferencia entre **cancelar** y **eliminar**, ya que son dos conceptos diferentes. **Si una colecta se elimina, no se notificará al usuario de esta eliminación, sino que simplemente desaparecerá, mientras que si se cancela, al usuario le aparecerá la colecta cancelada.**

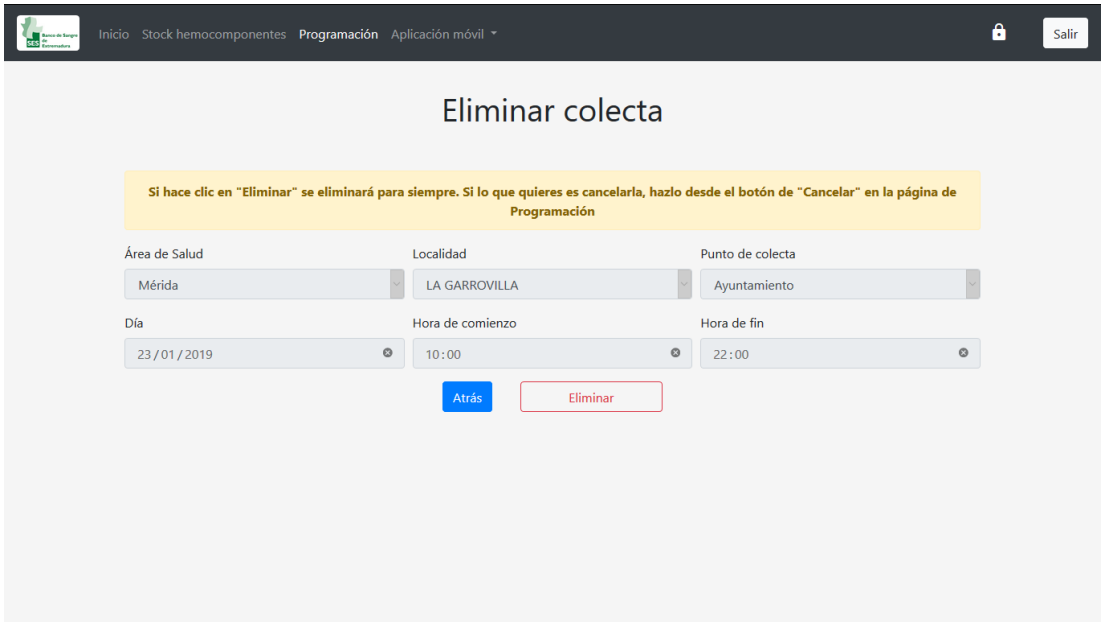


Figura A.16: Aplicación corporativa. Eliminar colectas programadas

Ahora vamos a pasar al apartado de las colectas canceladas. Para acceder, lo haremos desde la pantalla de **Programación** pulsando el botón **Reactivar colectas canceladas**. En esta página, que podemos observar en la figura A.17, veremos un listado de eventos que todavía no han concluido (fecha igual o superior que el día actual) que han sido cancelados.

Si queremos reactivar alguna de las colectas canceladas, tan solo deberemos pulsar en el botón amarillo con una flecha curva hacia la derecha.

Nos aparecerá una nueva página (figura A.18) para confirmar la reactivación de este evento, eliminándose de los eventos cancelados y trasladándose al listado de la pantalla de **Programación**.

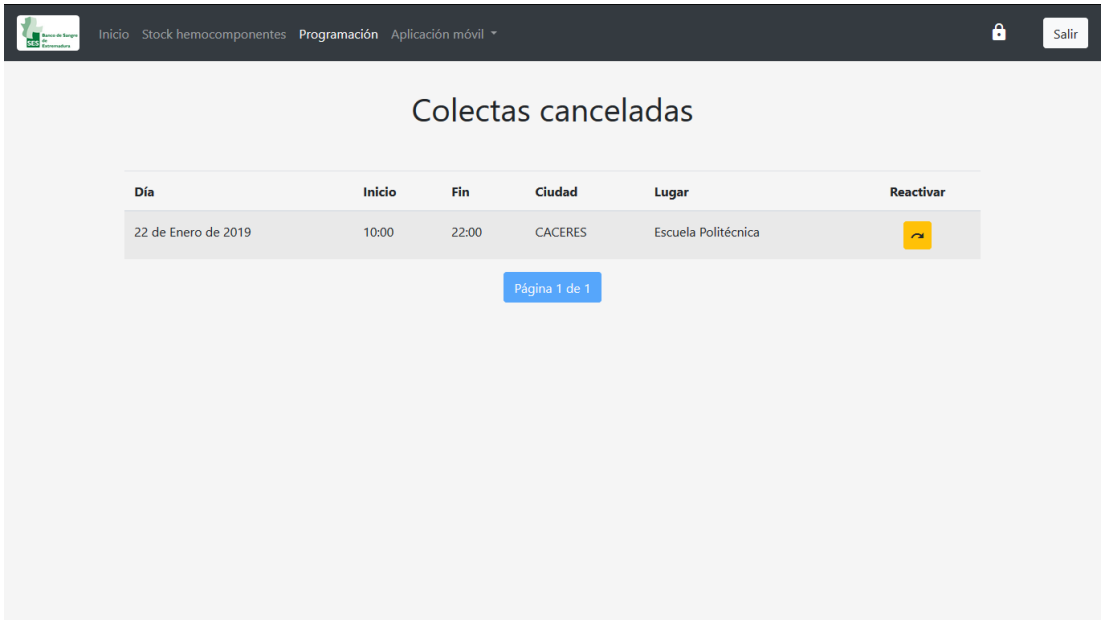


Figura A.17: Aplicación corporativa. Colectas canceladas

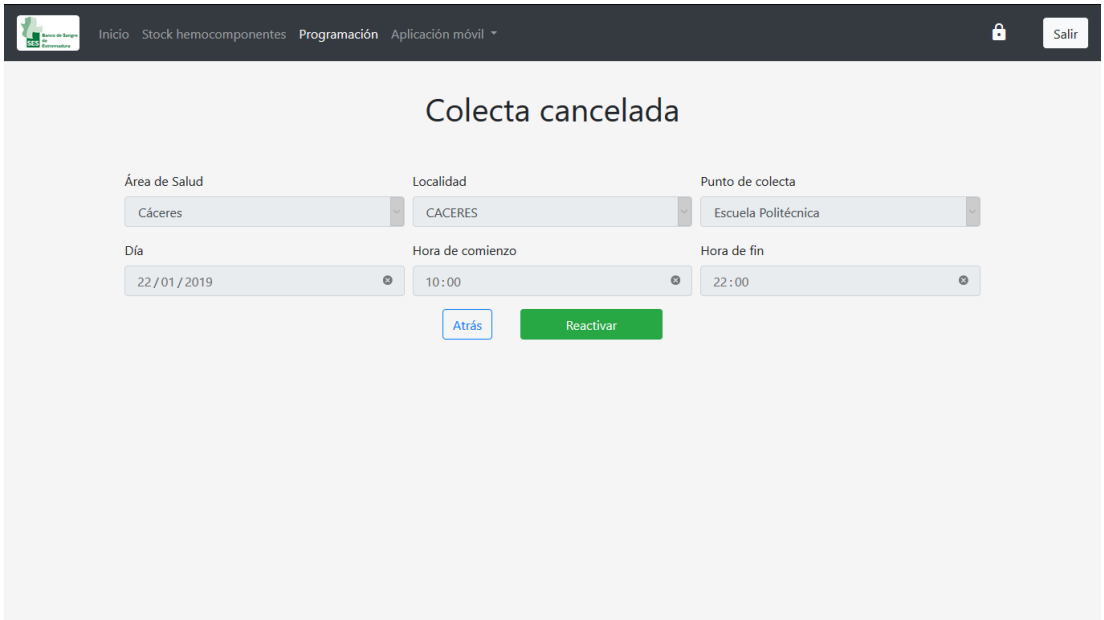


Figura A.18: Aplicación corporativa. Reactivar colectas canceladas

Por último, veremos la sección para añadir los resultados de las colectas. Para ello, tendremos que pulsar el botón **Resultados de colecta** y nos aparecerá un listado con todas las colectas que han terminado o que van a terminar en el día actual. La página cargada es la representada en la figura A.19.

Tenemos tres opciones, añadir el resultado, revisarlo en detalle y modificarlo. Hay que tener en cuenta que si no se ha añadido un resultado, no se podrá modificar ni revisar en detalle. Además, si no se añade el resultado en un plazo de tres días, no se podrá ni añadir, ni modificar ni ver en detalle.

Figura A.19: Aplicación corporativa. Resultados de las colectas

Para añadir el resultado de una colecta, tendremos que pulsar el botón de color turquesa con el símbolo de suma de color blanco, se cargará la página de la figura A.20 donde se muestran los datos de las colectas y se deberán rellenar dos campos, el correspondiente a las bolsas conseguidas y a los nuevos donantes.
Para guardar los resultados, deberemos pulsar el botón **Guardar**. Una vez guardado, podrá modificarse y revisar esos datos, como se explicaba anteriormente.

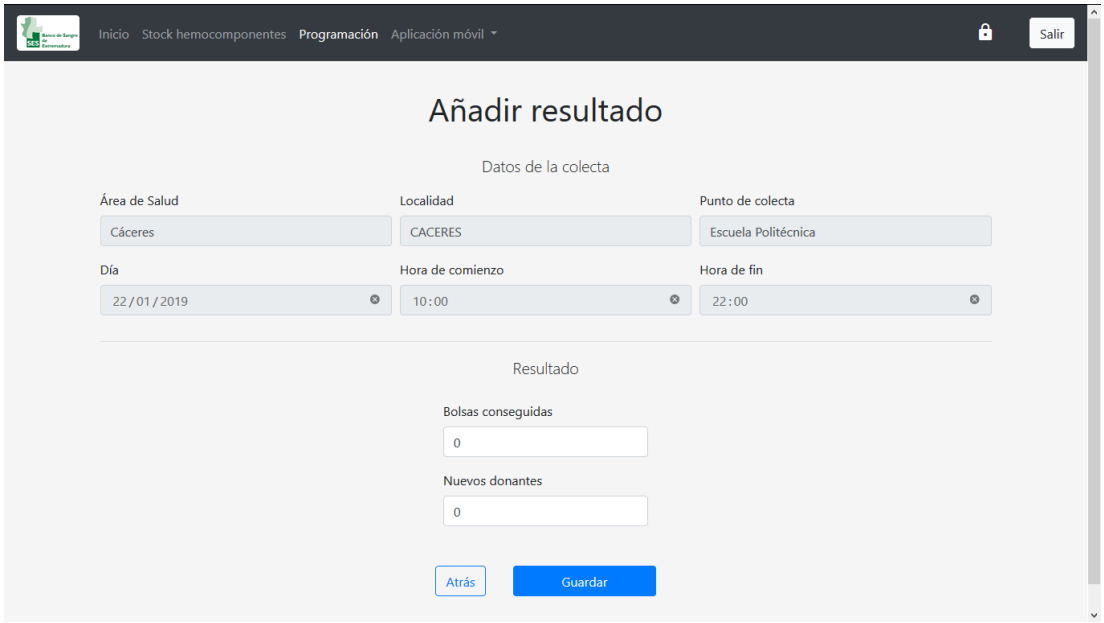


Figura A.20: Aplicación corporativa. Añadir resultados de las colectas

De la misma forma, podremos modificar los resultados pulsando en el botón azul con un lápiz blanco. Podemos ver esta página en la figura A.21. Para actualizar los valores modificados, pulsaremos el botón **Actualizar**.

Figura A.21: Aplicación corporativa. Modificar resultados de las colectas

Por último, podremos revisar los resultados pulsando el botón verde con un ojo

blanco. Podemos ver esta página en la figura A.22.

Figura A.22: Aplicación corporativa. Revisar resultados de las colectas

Ahora pasamos a otras de las secciones de la aplicación corporativa. Como explicamos anteriormente, las **alertas de emergencia** son alertas con carácter urgente que tienen la finalidad de avisar al usuario de un evento extraordinario, por el cual el Banco de Sangre de Extremadura va a perder sustancialmente parte de todas sus reservas.

Para estas ocasiones y tan solo éstas, se crearán las alertas de emergencia. Podemos ver las alertas que se han mandado con anterioridad desde el menú superior de navegación, pulsando sobre **Aplicación móvil** y seleccionando la opción **Alertas de emergencia**.

En la figura podremos observar un listado de alertas enviadas y dos estados posibles: **enviada** o **caducada**. Al crearla estará durante dos horas en estado enviada y posteriormente pasará a caducada, ya que es el tiempo máximo esperado que tarda un usuario en recibir esta alerta.

Figura A.23: Aplicación corporativa. Alertas de emergencia

Para crear una alerta tan solo deberemos hacer clic en el botón **Añadir nueva alerta de emergencia**. Se cargará la página **Nueva alerta de emergencia** donde tendremos que rellenar el título y el cuerpo de la alarma y pulsar el botón **Enviar alerta**. Podemos observar esta página en la figura A.24. **Todas las alertas enviadas son guardadas y el usuario que la envía queda registrado**.

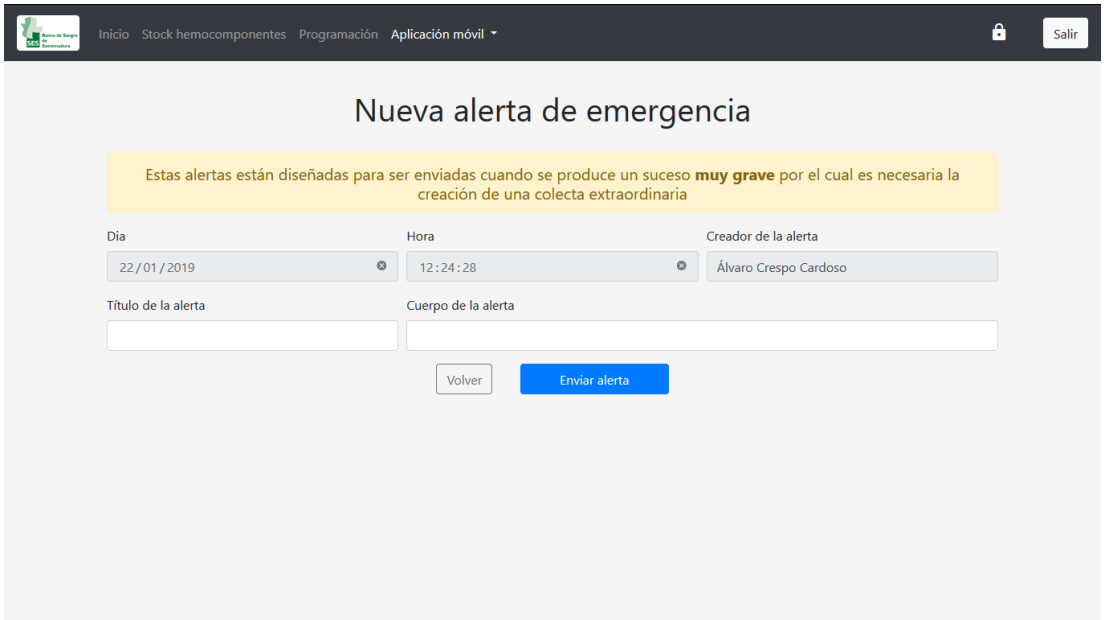


Figura A.24: Aplicación corporativa. Añadir alerta de emergencia

Otras de las funcionalidades de la aplicación corporativa es la creación de noticias. Se puede encontrar el listado de noticias creadas haciendo clic en **Aplicación móvil** en el menú superior y seleccionando **Noticias**. Nos aparecerá la página tal y como puede verse en la figura A.25 (con las noticias creadas por los distintos usuarios).

Figura A.25: Aplicación corporativa. Noticias

Las noticias tienen dos estados diferentes, **publicada** o **borrador**. Se podrán añadir, eliminar, guardar como borrador, publicar borrador, modificar borrador y eliminar borrador.

Para añadir una noticia, haremos clic en el botón **Añadir nueva noticia**. Nos aparecerá la página de la figura A.26.

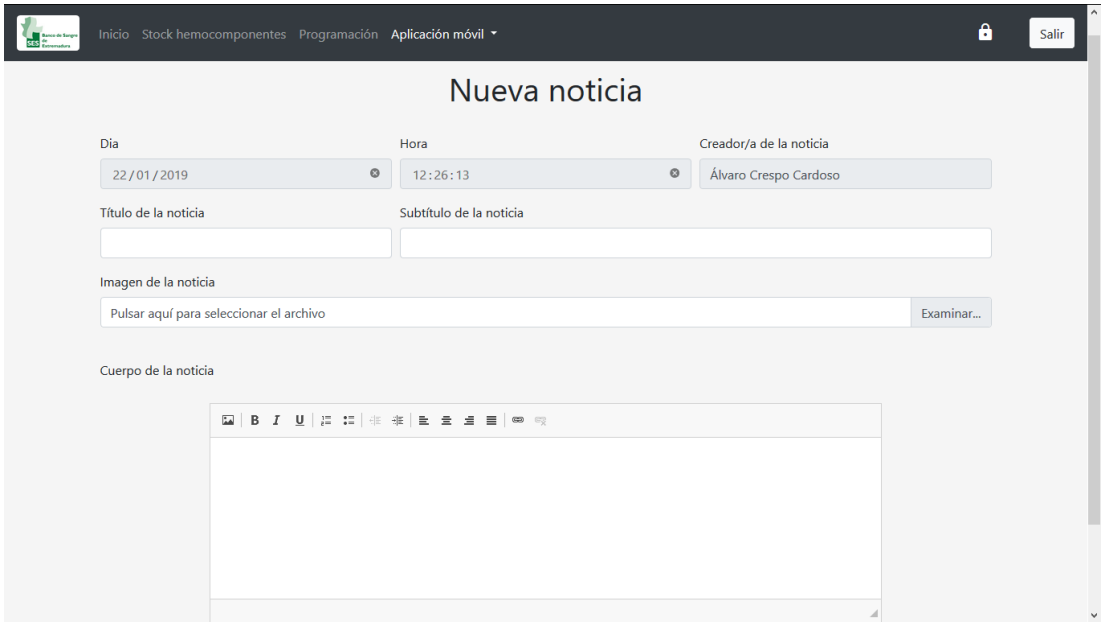


Figura A.26: Aplicación corporativa. Añadir noticias

En esta página tendremos que rellenar el título de la noticia, el subtítulo de la noticia, la imagen destacada de la noticia y el cuerpo de la noticia.
Se ha creado de tal forma que, en el cuerpo de la noticia se podrán añadir imágenes, se podrán alinear los textos, crear listas ordenadas o desordenadas, añadir enlaces y utilizar negritas, subrayados y cursivas.
Una vez creada la noticia se puede **guardar como borrador** para seguir modificándola posteriormente o **publicarla**. **Los usuarios de la aplicación móvil no podrán verla hasta que su estado sea publicado**.

Para ver la noticia en detalle deberemos pulsar el botón verde con un ojo blanco. Nos aparecerá la página de la figura A.27 con los datos de la noticia creada. No se podrá modificar ningún campo desde esta página.

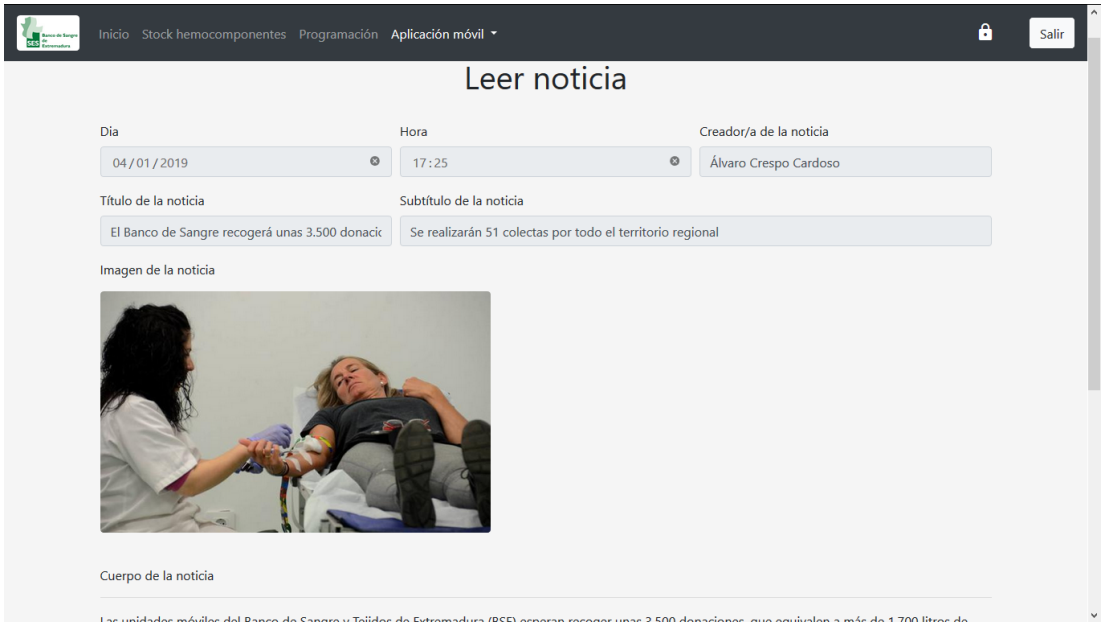


Figura A.27: Aplicación corporativa. Revisar noticias

También tenemos la opción de eliminar una noticia creada, haciendo clic en el botón rojo con una cubo de basura blanco. Se cargará una página (figura A.28) donde nos mostrará la noticia en detalle que quiere ser eliminada. Para eliminarla será necesario hacer clic en el botón **Eliminar**.

Figura A.28: Aplicación corporativa. Eliminar noticias

Ahora pasaremos a los borradores de las noticias. Si hacemos clic en **Ver borradores** nos aparecerán los borradores (figura A.29). Además, podremos hacer tres operaciones: **publicar**, **modificar** y **eliminar**.

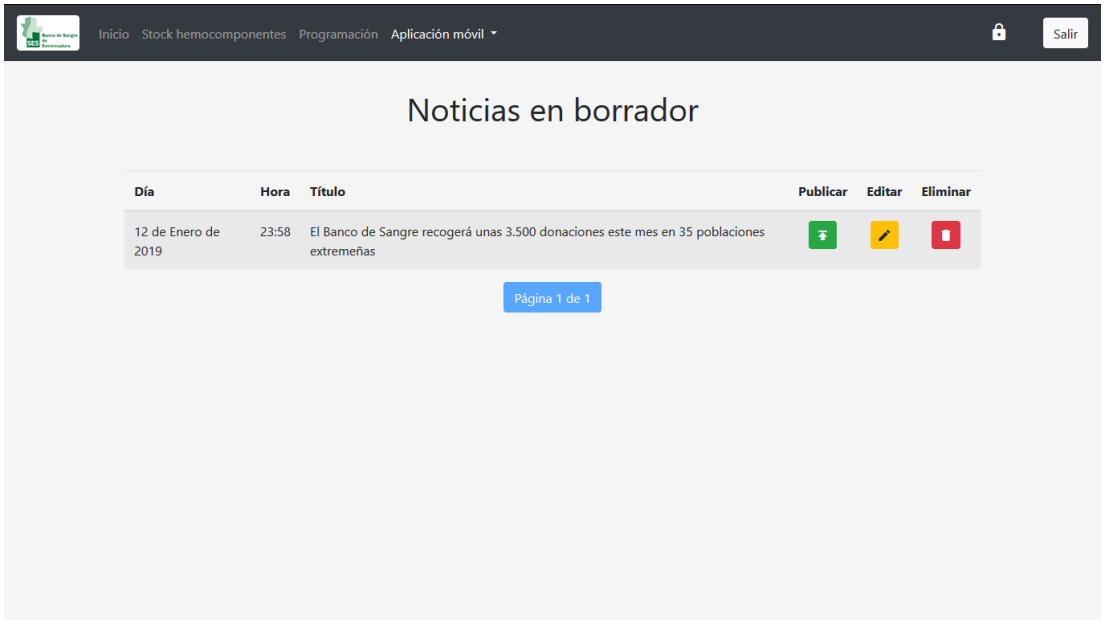


Figura A.29: Aplicación corporativa. Borradores

Si pulsamos el botón verde con una flecha blanca hacia arriba, se cargará la página de la figura A.30 dónde se nos mostrarán los datos de esa noticia y haciendo clic en el botón **Publicar**, será publicada de inmediato.

Si pulsamos el botón amarillo con un lápiz blanco, se cargará la página de la figura A.31. Aquí podremos modificar todos los campos del borrador. En este caso tenemos dos opciones, pulsar el botón **Publicar**, con lo que el borrador pasaría a ser una noticia publicada o pulsar el botón **Actualizar borrador**, con lo que se actualizaría el borrador, pero sin publicarlo.

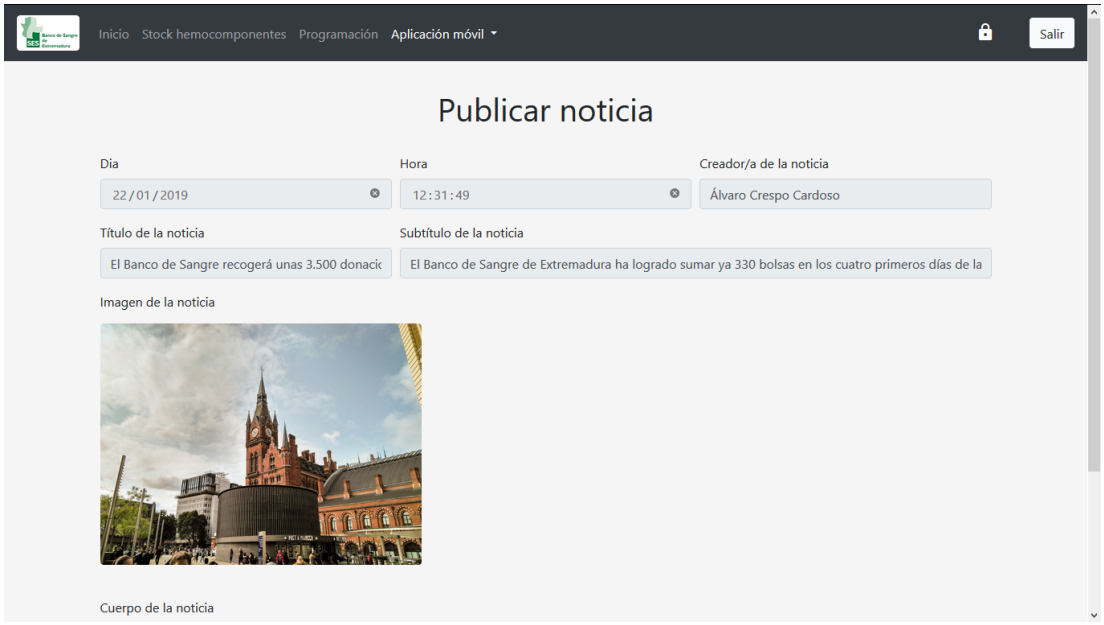


Figura A.30: Aplicación corporativa. Publicar borradores

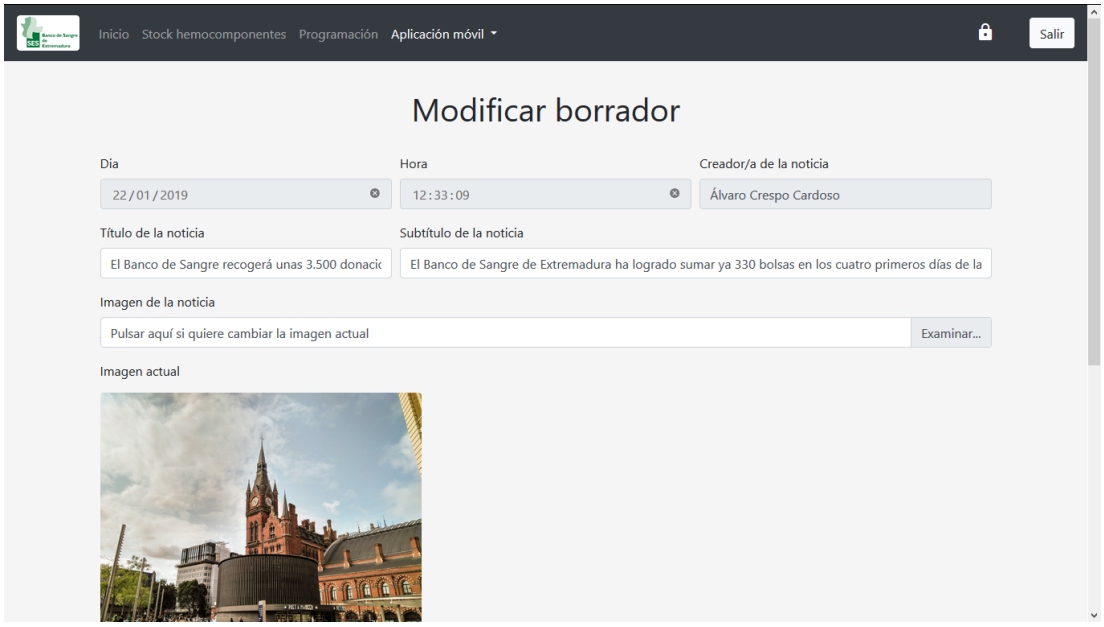


Figura A.31: Aplicación corporativa. Modificar borradores

Por último, podremos eliminar los borradores. Para ello deberemos hacer clic en el botón rojo con un cubo de basura blanco. Al hacer clic se cargará la página de la figura . En ella se mostrarán los datos del borrador a eliminar. Pulsando en el botón **Eliminar**, se eliminará de forma definitiva.

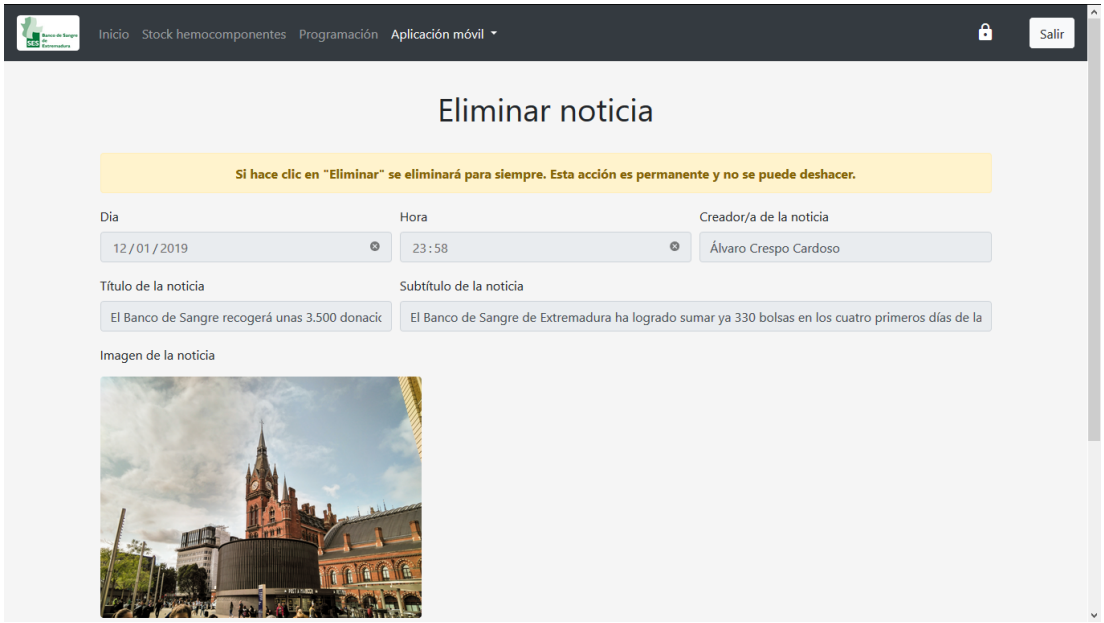


Figura A.32: Aplicación corporativa. Eliminar borradores

Una de las páginas más importantes para el usuario es la de **Perfil**. Para acceder a la página de perfil, el usuario deberá hacer clic en el icono del candado blanco en la barra de navegación, cercana al botón de salir. En esta página, ilustrada en la figura A.33, podemos observar nuestro nombre, apellidos y grupo de usuarios al que pertenecemos. Dependiendo del grupo al que pertenezcamos, podremos acceder a unas cosas u otras. También podremos cambiar la contraseña, **cosa que es necesaria la primera vez que se inicia sesión con las credenciales proporcionadas por el administrador del sistema**.

En último lugar recordar que por nuestra seguridad y para que ningún usuario pueda utilizar nuestra cuenta, **deberemos cerrar sesión**. La sesión se cerrará también de forma automática cuando pasen cuatro horas desde el último acceso por seguridad.
Para cerrar sesión, tan solo tendremos que hacer clic en el botón de la barra de navegación **Salir**.

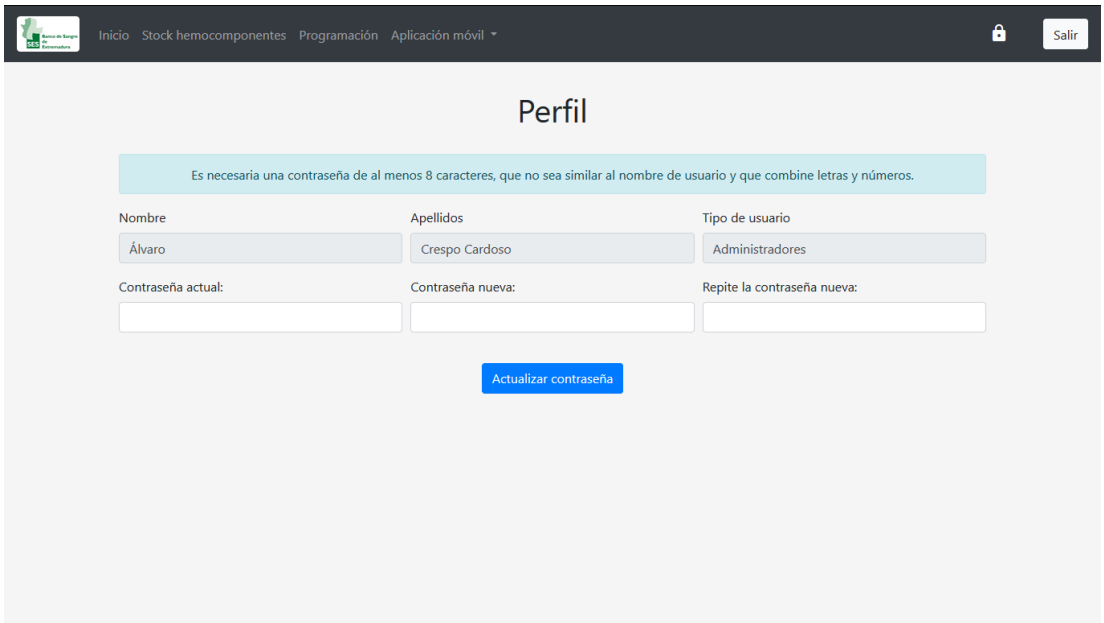


Figura A.33: Aplicación corporativa. Perfil

Apéndice B

Manual de instalación. Aplicación corporativa

Para la instalación de la aplicación web, se han determinado unos requisitos recomendables, concretamente:

- **Sistema operativo**: Ubuntu 18.04.1 LTS 64 bits
- **Memoria RAM**: 4 GB
- **Disco duro**: 32 GB
- **CPU**: Procesador con dos núcleos a 2,20 GHz cada uno

Una vez hemos cumplido estos requisitos, comenzaremos con la instalación. Para ello, necesitaremos copiar el directorio que contiene la aplicación Django, en nuestro caso desde Windows, al servidor remoto, para ello crearemos el directorio **www** en la ruta **/var/** y crearemos el directorio **bse** en la ruta **/var/www/** mediante los siguientes comandos:

```
cd /var
mkdir www
cd www
mkdir bse
```

En nuestro caso, tenemos el proyecto en una carpeta del disco C (local) y la copiaremos al directorio **/var/www/bse**.

En primer lugar será necesario descargar PuTTY, que es un cliente de telnet y SSH para Windows desde esta página web: https://www.putty.org/. Tras la instalación abriremos la consola de comandos de Windows o ”símbolo del sistemaz ejecutaremos el siguiente comando:

```
pspc -r C:\BSE_WebApp usuario.linux@ip.maquina:/var/www/bse
```

Tras la copia de todos los ficheros comenzaremos con la instalación de las herramientas necesarias.

Instalación de pip para Python 3:

```
sudo apt-get install python3-pip
```

Instalación de virtualenv:

```
pip3 install virtualenv
```

Creación de un entorno virtual:

```
cd \var\www\bse\BSE_WebApp
virtualenv venv
```

Activación del entorno virtual:

```
source venv/bin/activate
```

Instalación de las herramientas necesarias para Django:

```
pip install --upgrade setuptools
pip install django
pip install django-ckeditor
pip install django-iprestrict
pip install django-jchart
pip install django-leaflet
pip install djangorestframework
```

```
pip install geopy
pip install gunicorn
sudo apt-get install python3.6-dev libmysqlclient-dev
pip install mysqlclient
pip install numpy
pip install pandas
pip install reportlab
pip install xlrd
```

Instalación de la base de datos MariaDB:

```
sudo apt-get remove mariadb-server
sudo apt-get install software-properties-common
sudo apt-key adv --recv-keys --keyserver
  hkp://keyserver.ubuntu.com:80 0xF1656F24C74CD1D8
sudo add-apt-repository 'deb [arch=amd64]
  http://mirror.zol.co.zw/mariadb/repo/10.3/ubuntu bionic
  main'
sudo apt update
sudo apt -y install mariadb-server mariadb-client
```

Después de introducir este comando se solicitará una contraseña para la base de datos. Tras esto comprobaremos que funciona la base de datos utilizando el siguiente comando:

```
mysql -u root -p
```

Si al pedirnos la contraseña e introducirla correctamente aparece el mensaje "mysql ERROR 1524 (HY000): Plugin 'unix_socket' is not loaded" deberemos realizar los siguientes pasos, en caso contrario, deberemos omitirlos.

```
sudo su
/etc/init.d/mysql stop
```

```
mysqld_safe --skip-grant-tables & mysql -uroot
use mysql;
update user set password=PASSWORD("CONTRASEÑA") where User='root';
update user set plugin="mysql_native_password";
quit;
/etc/init.d/mysql stop
kill -9 $(pgrep mysql)
/etc/init.d/mysql start
```

Ahora al comprobar que funciona la base de datos, no debería aparecer el anterior mensaje de error. Ahora crearemos la base de datos que será usada por la aplicación web Django utilizando los siguientes comandos:

```
mysql -u root -p
CREATE DATABASE bsedb;
exit
```

Deberemos modificar la contraseña de la base de datos en los archivos de configuración del driver de MariaDB en Django con los siguientes comandos:

```
cd /var/www/bse/BSE_WebApp/BSE_WebApp
nano settings.py
```

Pulsamos la tecla Ctrl + -, escribimos 91 y pulsamos Intro. Modificamos la contraseña root por la definida anteriormente. Pulsamos Ctrl + x y pulsamos Y. Por último, para confirmar los cambios pulsamos la tecla Intro.

Ahora ya podemos hacer las migraciones a la base de datos y la creación de los ficheros estáticos, para ello deberemos ejecutar los siguientes comandos:

```
cd /var/www/bse/BSE_WebApp/
python manage.py makemigrations
python manage.py migrate
python manage.py collectstatic
```

Una vez migrada, vamos a comprobar que se han creado todas las tablas, con los siguientes comandos:

```
mysql -u root -p
use bsedb;
show tables;
```

Deberían aparecer las tablas de la figura B.1.

```
+-----------------------------------+
| auth_group                        |
| auth_group_permissions            |
| auth_permission                   |
| auth_user                         |
| auth_user_groups                  |
| auth_user_user_permissions        |
| authtoken_token                   |
| django_admin_log                  |
| django_content_type               |
| django_migrations                 |
| django_session                    |
| panel_control_article             |
| panel_control_city                |
| panel_control_donationevent       |
| panel_control_donationpoint       |
| panel_control_fixeddonationevent  |
| panel_control_healtharea          |
| panel_control_notification        |
| panel_control_stockreport         |
+-----------------------------------+
19 rows in set (0.000 sec)

MariaDB [bsedb]>
```

Figura B.1: Instalación aplicación corporativa. Tablas generadas

Ahora, deberemos añadir unos datos a la base de datos, concretamente los siguientes:

```
INSERT INTO `panel_control_healtharea` (`id`, `name`) VALUES
        (1, 'Badajoz'),
```

```
(2, 'Mérida'),
(3, 'Don Benito - Vva. de la Serena'),
(4, 'Llerena - Zafra'),
(5, 'Cáceres'),
(6, 'Plasencia'),
(7, 'Coria'),
(8, 'Navalmoral de la Mata');
```

Por último y antes de comenzar con la instalación del servidor web y su posterior configuración vamos a crear el superusuario de Django que nos permitirá crear el resto de usuarios, para ello utilizaremos el siguiente comando:

```
python manage.py createsuperuser
```

Deberemos introducir el nombre del usuario, el correo electrónico y la contraseña. Una vez terminado deberá aparecer un mensaje: "Superuser created successfully".

Ahora pasaremos a la instalación del servidor Nginx. Para ello ejecutaremos las siguientes sentencias:

```
sudo apt update
sudo apt-get install nginx
```

Una vez instalado, vamos a crear el script para ejecutar Gunicorn[1]. Para ello ejecutaremos los siguientes comandos:

```
cd /var/www/bse
sudo nano gunicorn_start.sh
```

Ahora copiaremos el siguiente script:

```
#!/bin/bash
NAME="BSE_WebApp"
DJANGODIR=/var/www/bse/BSE_WebApp
SOCKFILE=/var/www/bse/BSE_WebApp/run/gunicorn.sock
```

[1]Gunicorn es un servidor HTTP WSGI para entornos UNIX.

```
USER=root
GROUP=root
NUM_WORKERS=2
DJANGO_SETTINGS_MODULE=BSE_WebApp.settings
DJANGO_WSGI_MODULE=BSE_WebApp.wsgi

echo "Starting $NAME as `whoami`"

cd $DJANGODIR
source /var/www/bse/BSE_WebApp/venv/bin/activate
export DJANGO_SETTINGS_MODULE=$DJANGO_SETTINGS_MODULE
export PYTHONPATH=$DJANGODIR:$PYTHONPATH

RUNDIR=$(dirname $SOCKFILE)
test -d $RUNDIR || mkdir -p $RUNDIR

exec /var/www/bse/BSE_WebApp/venv/bin/gunicorn
  ${DJANGO_WSGI_MODULE}:application \
  --name $NAME \
  --workers $NUM_WORKERS \
  --group $GROUP \
  --user $USER \
  --timeout 2400 \
  --bind=unix:$SOCKFILE \
```

Posteriormente le daremos permisos de ejecución mediante el siguiente comando:

```
chmod 744 gunicorn_start.sh
```

A continuación crearemos el servicio que iniciará la aplicación automáticamente en caso de reinicio del servidor, ejecutaremos los siguientes comandos:

```
cd /etc/systemd/system
sudo nano gunicorn_BSE_WebApp.service
```

Ahora copiaremos el siguiente script:

```
[Unit]
Description=BSE_WebApp gunicorn daemon

[Service]
Type=simple
User=root
ExecStart=/var/www/bse/gunicorn_start.sh

[Install]
WantedBy=multi-user.target
```

Ahora, le daremos los permisos necesarios para la ejecución:

```
chmod 755 gunicorn_BSE_WebApp.service
```

Por último deberemos modificar la configuración de Nginx para que pueda servir nuestra aplicación web, para ello ejecutaremos el siguiente comando:

```
sudo nano /etc/nginx/nginx.conf
```

Y copiaremos el siguiente texto de configuración sustituyendo al anterior:

```
user www-data;
worker_processes auto;
pid /run/nginx.pid;
include /etc/nginx/modules-enabled/*.conf;

events {
        worker_connections 16384;
        # multi_accept on;
```

```
}

http {

        ##
        # Basic Settings
        ##

        sendfile on;
        tcp_nopush on;
        tcp_nodelay on;
        keepalive_timeout 65;
        types_hash_max_size 2048;
        # server_tokens off;

        # server_names_hash_bucket_size 64;
        # server_name_in_redirect off;

        include /etc/nginx/mime.types;
        default_type application/octet-stream;

        upstream test_server {
                server unix:/var/www/bse/BSE_WebApp/run/gunicorn.sock
                fail_timeout=10s;
        }
        server {
                listen 80 reuseport;
                server_name 158.49.112.144;
```

```
        client_max_body_size 4G;

        location /static/ {
                autoindex on;
                alias /var/www/bse/BSE_WebApp/static/;
        }

        location /upload/ {
                autoindex on;
                alias /var/www/bse/BSE_WebApp/upload/;
        }

        location / {
                proxy_set_header X-Forwarded-For
                  $proxy_add_x_forwarded_for;
                proxy_set_header Host $http_host;
                proxy_redirect off;
                proxy_connect_timeout 70;
                proxy_send_timeout 90;
                proxy_read_timeout 2400;
                if (!-f $request_filename) {
                        proxy_pass http://test_server;
                        break;
                }
        }
}
##
# SSL Settings
##
```

```
ssl_protocols TLSv1 TLSv1.1 TLSv1.2; # Dropping SSLv3, ref: POODLE
ssl_prefer_server_ciphers on;

##
# Logging Settings
##

access_log /var/log/nginx/access.log;
error_log /var/log/nginx/error.log;

##
# Gzip Settings
##

gzip on;

# gzip_vary on;
# gzip_proxied any;
# gzip_comp_level 6;
# gzip_buffers 16 8k;
# gzip_http_version 1.1;
# gzip_types text/plain text/css application/json
  # application/javascript text/xml application/xml
  # application/xml+rss text/javascript;


##
# Virtual Host Configs
```

```
        ##

        include /etc/nginx/conf.d/*.conf;
        include /etc/nginx/sites-enabled/*;
}
```

Con esto estaría todo configurado, tan solo tendremos que reiniciar los servicios para que se apliquen los cambios, para ello ejecutaremos los siguientes comandos:

```
1  systemctl restart gunicorn_BSE_WebApp.service
2  systemctl restart nginx.service
```

Si accedemos al servidor por su IP (158.49.112.144 en nuestro caso), debería aparecer la página de acceso.

Ahora, deberemos de añadir los puntos de colecta por primera vez. En primer lugar deberemos de descargar del siguiente enlace la hoja de cálculo con todos los datos: `https://drive.google.com/file/d/19frrfukrMVjqH4R-f-0kgzeD5gR6_5Hs` Una vez descargado, accederemos a la página de **Programación** >**Actualizar puntos de colecta**, seleccionaremos el fichero descargado. Pulsaremos **Actualizar** y esperaremos pacientemente, ya que al ser la primera carga, será un proceso largo que tomará entre 10 y 15 minutos aproximadamente, dependiendo de la máquina y del estado de saturación del servicio Open Street Maps Nominatim.

Una vez terminado el proceso, el único paso que nos quedaría por hacer para terminar la instalación y despliegue de forma completa y efectiva, sería la creación de los grupos y la asignación de los permisos a los diferentes grupos.
Para ello tendremos que acceder a la página de administración, en nuestro ejemplo http://158.49.112.144/admin/ y en **Autenticación y autorización** seleccionamos el apartado de grupos. A continuación se crearán los siguientes:

- **Administradores**: tendrán todos los permisos.
- **Administrativos**: tendrán acceso a todos los permisos que comiencen por “panel control | informe stock” y “panel control | evento de donación”.

- **Gestores de promoción**: tendrán acceso a todos los permisos que comiencen por “panel control | artículo”.

- **API**: no tendrán ningún permiso.

Estos son los grupos que han sido definidos por el cliente pero pueden ser modificados con gran facilidad y flexibilidad.

Por último deberemos crear el usuario que será utilizado por la aplicación móvil para conectarse con la API REST, concretamente será:

- **Nombre de usuario**: app

- **Contraseña**: 22A405A22E6CE6FF059A8D1BDEC434DE69E3ABB225D1963DD 987521CE8362FE6

Ahora, una vez lo hemos creado pasaremos a modificar más atributos de este usuario, concretamente:

- **Permisos**: tan solo marcaremos la casilla **Activo**, si hay alguna marcada, la quitaremos.

- **Grupos**: seleccionamos API.

Apéndice C

Algoritmo de creación y actualización de los puntos de colectas y eventos fijos de donación

Mediante el siguiente algoritmo y teniendo como entrada una hoja de cálculo con formato *xlsx* se crearán o actualizarán todos los puntos de donación, ya sean fijos o móviles y también todos los eventos de donación que sean fijos.

```
Procedimiento setup_places (excel)
    places ← DonationPoint.objects.all()
    cities ← City.objects.all()
    health_areas ← HealthArea.objects.all()
    # pandas.read_excel (hoja excel a leer, nombre de la hoja a leer)
    # devuelve un DataFrame
    data ← pandas.read_excel (excel, 'Extremadura completo')
    Si places.count() = 0 Entonces
        Para fila En data.values() Hacer
            # Fila[Columna]. Esa columna tiene los valores VÁLIDO
            # o NO VÁLIDO
            Si fila[5] = 'VÁLIDO' Entonces
```

```
                setup_cities_bd(fila, health_areas)
                dp ← DonationPoint()
                # fila[0] es el nombre de la ciudad
                city ← City.objects.filter(fila[0])[0]
                dp.city_id ← city.id
                setup_point_db(fila, dp, Falso)
            Fin Si
        Fin Para
    Si no Entonces
        Para lugar En places Hacer
            p ← DonationPoint.objects.get(lugar.id)
            p.active ← Verdadero
            p.save()
        Fin Para
        setup_successive(data, places, health_areas, Falso)
    Fin Si
    fixed_events ← FixedDonationEvent.objects.all()
    Para evento En fixed_events Hacer
        evento.active ← Falso
        evento.save()
    Fin Para
    data ← pandas.read_excel (excel, 'Puntos fijos')
    setup_successive (data, places, health_areas, Verdadero)
Fin Procedimiento

Procedimiento setup_cities_db (fila, health_areas)
    # fila[0] es el nombre de la localidad
    city ← City.objects.filter(fila[0])
    Si city.count() = 0 Entonces
```

```
        city ← City()
        city.name ← fila[0]
        # para utilizar Nominatim es necesario proporcionar un
        # user_agent, y en este caso es un correo electrónico
        geolocator ← Nominatim('alcrespoc@alumnos.unex.es')
        # fila[6] es el nombre de la provincia
        location ← geolocator.geocode(city.name + ', ' + fila[6])
        Si is_inside_extremadura(location) Entonces
            city.latitude ← location.latitude
            city.longitude ← location.longitude
        Fin Si
        city.province ← fila[6]
        Para area en health_areas Hacer
            Si area.name = fila[3] Entonces
                city.health_area ← area
                Break
            Fin Si
        Fin Para
        city.save(City)
    Fin Si
Fin Procedimiento

Función is_inside_extremadura(location)
    Si location ≠Nulo Entonces
        Si 40.48621 ≥location.latitude ≥37.94103 Y
            -4.648903 ≥location.longitude ≥-7.541611 Entonces
            Escribe Verdadero
        Fin Si
    Fin Si
```

```
    Escribe Falso
Fin Función

# fixed_point es un booleano que indica si es o no un punto fijo
Procedimiento setup_point_db(fila, dp, fixed_point)
    dp.name ← fila[1]
    dp.address ← fila[2]
    geolocator ← Nominatim('alcrespoc@alumnos.unex.es')
    Si fixed_point = Falso Entonces
        location ← geolocator.geocode(dp.address)
    Fin Si

    Si fixed_point = Verdadero
        o is_inside_extremadura(location) = Falso Entonces
        location ← geolocator.geocode(dp.name+' '+dp.city.name)
        Si is_inside_extremadura(location) = Falso Entonces
            location ← geolocator.geocode(dp.city.name+' '+dp.city.province)
            Si is_inside_extremadura(location) = Falso Entonces
                dp.latitude ← 0.0
                dp.longitude ← 0.0
            Si no Entonces
                dp.latitude ← location.latitude
                dp.longitude ← location.longitude
            Fin Si
        Si no Entonces
            dp.latitude ← location.latitude
            dp.longitude ← location.longitude
        Fin Si
    Si no Entonces
```

```
        dp.latitude ← location.latitude
        dp.longitude ← location.longitude
    Fin Si
    dp.fixed_point ← fixed_point
    dp.active ← Verdadero
    Si dp.id ≠Nulo Entonces
        dp.save()
    Si no Entonces
        dp.save(DonationPoint)
    Fin Si
Fin Procedimiento

Procedimiento setup_successive(data, places, health_areas, fixed_point)
    Para fila En data.values Hacer
        no_existe_nada ← Verdadero
        existen_ambos ← Falso
        Si fila[5] = 'VÁLIDO' Entonces
            Para lugar En places Hacer
                Si fila[0] = lugar.city.name Entonces
                    no_existe_nada ← Falso
                    Si fila[1] = lugar.name Entonces
                        existen_ambos ← Verdadero
                        Si fila[2] ≠lugar.address Entonces
                            lugar.address ← fila[2]
                            setup_point_db(fila, lugar, fixed_point)
                        Si no Entonces
                            lugar.active ← Verdadero
                            lugar.save()
                        Fin Si
```

```
                    Si fixed_point = Verdadero Entonces
                        setup_fixed_events(fila, lugar)
                    Fin Si
                    Break
                Fin Si
            Fin Si
        Fin Para
        Si existen_ambos = Falso Entonces
            Si no_existe_nada = Verdadero Entonces
                setup_cities_db(fila, health_areas)
            Fin si
            dp ← DonationPoint()
            city ← City.objects.filter(fila[0])[0]
            dp.city_id ← city.id
            setup_point_db(fila, dp, fixed_point)
        Fin Si
    Fin Si
  Fin Para
Fin Procedimiento

Procedimiento setup_fixed_events(fila, donation_point)
  fixed_events ← FixedDonationEvent.objects.all();
  exist ← Falso;
  Si fixed_events ≠Nulo Entonces
    Para evento En fixed_events Hacer
        Si evento.donation_point_id = donation_point.id Y
            evento.donation_point.name = fila[1] Entonces
            # fila[8] contiene el horario del evento fijo
            # El método partition permite dividir un String
```

```
                # El [0] contiene la hora de comienzo y el [2] la de fin
                hour_start ← fila[8].partition('-')[0];
                hour_end ← fila[8].partition('-')[2];
                exist ← Verdadero;
                Si evento.day_start ≠get_day_by_abbreviation(fila[7][0]) O
                    evento.day_end ≠get_day_by_abbreviation(fila[7][2]) O
                    evento.hour_start ≠hour_start O
                    evento.hour_end ≠hour_end O
                    evento.active = Falso Entonces

                    evento.day_start ← get_day_by_abbreviation(fila[7][0]);
                    evento.day_end ← get_day_by_abbreviation(fila[7][2]);
                    evento.hour_start ← hour_start;
                    evento.hour_end ← hour_end;
                    evento.active ← Verdadero;
                    evento.save();
                Si no Entonces
                    evento.active ← Verdadero
                    evento.save();
                Fin Si
            Fin Si
        Fin Para
    Fin Si
    Si fixed_events = Nulo O exist = Falso Entonces
        fde ← FixedDonationEvent()
        fde.day_start ← get_day_by_abbreviation(fila[7][0]);
        fde.day_end ← get_day_by_abbreviation(fila[7][2]);
        fde.hour_start ← fila[8].partition('-')[0];
        fde.hour_end ← fila[8].partition('-')[2];
```

```
        fde.active ← Verdadero;
        fde.save();
    Fin Si
Fin Procedimiento

Función get_day_by_abbreviation(dia)
    Según dia Hacer
        caso 'L';
            Escribir 'Lunes';
        caso 'M';
            Escribir 'Martes';
        caso 'X';
            Escribir 'Miércoles';
        caso 'J';
            Escribir 'Jueves';
        caso 'V';
            Escribir 'Viernes';
        caso 'S';
            Escribir 'Sábado';
        caso 'D';
            Escribir 'Domingo';
    Fin Según
Fin Función
```

Bibliografía

1. SANIDAD Y CONSUMO, Consejería de. *DECRETO 161/2004, de 26 de octubre, por el que se regula el Sistema de Hemoterapia de Extremadura* [online]. 2004 [visitado 2019-01-30]. Disponible desde: `http://doe.gobex.es/pdfs/doe/2004/1270o/04040174.pdf`.
2. SHAABAN, Sam. *Web-Based vs "Desktop" Software* [online] [visitado 2018-12-23]. Disponible desde: `https://nurelm.com/the-lab/web-based-vs-desktop-software`.
3. CÍA, Juan F. *MEAN, el cuarteto de desarrollo 'full-stack' en JavaScript para hacer aplicaciones web: El JavaScript aplicado a nivel de cliente y servidor permite a los desarrolladores adentrarse en proyectos 'full-stack' de desarrollo de aplicaciones web. MongoDB, Express, AngularJS y Node.js (MEAN) forman una pila completa de desarrollo.* [online]. 2015 [visitado 2018-12-26]. Disponible desde: `https://bbvaopen4u.com/es/actualidad/mean-el-cuarteto-de-desarrollo-full-stack-en-javascript-para-hacer-aplicaciones-web-0`.
4. *React - A JavaScript library for building user interfaces* [online] [visitado 2018-12-26]. Disponible desde: `https://reactjs.org/docs/getting-started.html`.
5. *Introduction: What is Vue.js?* [online] [visitado 2018-12-26]. Disponible desde: `https://vuejs.org/v2/guide/`.

6. *Introduction to GraphQL* [online] [visitado 2018-12-26]. Disponible desde: https://graphql.org/learn/.

7. *npm* [online] [visitado 2018-12-26]. Disponible desde: https://www.npmjs.com/.

8. *Resumen - Apache Cordova* [online] [visitado 2018-12-26]. Disponible desde: https://cordova.apache.org/docs/es/8.x/guide/overview/index.html.

9. LOZINSKY, Yuriy. *6 Web Development Stacks to Try in 2017* [online]. 2017 [visitado 2018-12-26]. Disponible desde: https://webinerds.com/6-web-development-stacks-try-2017/.

10. *Introducción a Django* [online]. 2018 [visitado 2018-12-26]. Disponible desde: https://developer.mozilla.org/es/docs/Learn/Server-side/Django/Introducci%C3%B3n.

11. *Getting Started with Rails* [online] [visitado 2018-12-26]. Disponible desde: https://guides.rubyonrails.org/getting_started.html.

12. GABRY, Omar El. *Spring: A Head Start — Introduction (Part 1): What, Why, Concepts, and Architecture* [online]. 2017 [visitado 2018-12-26]. Disponible desde: https://medium.com/omarelgabrys-blog/spring-a-head-start-introduction-part-1-130aa1b41e47.

13. *Converting from other Databases to PostgreSQL* [online]. 2018 [visitado 2018-12-26]. Disponible desde: https://wiki.postgresql.org/wiki/Converting_from_other_Databases_to_PostgreSQL#MySQL.

14. *Guía de arquitectura de apps* [online]. 2018 [visitado 2018-12-26]. Disponible desde: https://developer.android.com/jetpack/docs/guide?hl=es-419.

15. ABLESON, Frank. *Introduction to Android development: Discover the depth and breadth of the Android platform* [online]. 2018 [visitado 2018-12-22]. Disponible desde: https://developer.ibm.com/articles/os-android-devel/.

16. EGHAM. *Gartner Says Worldwide Mobile Phone Sales to End Users Grew 8 Per Cent in Fourth Quarter 2009; Market Remained Flat in 2009* [online]. 2010 [visitado 2018-12-22]. Disponible desde: https://www.gartner.com/newsroom/id/1306513.

17. EGHAM. *Gartner Says Huawei Secured No. 2 Worldwide Smartphone Vendor Spot, Surpassing Apple in Second Quarter 2018: Worldwide Sales of Smartphones Grew 2 Percent in Second Quarter 2018* [online]. 2018 [visitado 2018-12-22]. Disponible desde: https://www.gartner.com/en/newsroom/press-releases/2018-08-28-gartner-says-huawei-secured-no-2-worldwide-smartphone-vendor-spot-surpassing-apple-in-second-quarter.

18. *Content License* [online] [visitado 2018-12-22]. Disponible desde: https://source.android.com/license.

19. *Download IntelliJ IDEA* [online]. 2018 [visitado 2018-12-22]. Disponible desde: https://www.jetbrains.com/idea/download/.

20. *Términos y condiciones* [online]. 2016 [visitado 2018-12-22]. Disponible desde: https://developer.android.com/studio/terms?hl=es-419.

21. CHAN, Edwin. *The Great Firewall of China* [online]. Ed. por CLARK, Grant. 2018 [visitado 2018-12-22]. Disponible desde: https://www.bloomberg.com/quicktake/great-firewall-of-china.

22. *Cómo utilizar Play Console: Registrarse para obtener una cuenta de desarrollador de Google Play* [online] [visitado 2018-12-22]. Disponible desde: https://support.google.com/googleplay/android-developer/answer/6112435?hl=es.

23. SINICKI, Adam. *I want to develop Android Apps — What languages should I learn?* [online]. 2017 [visitado 2018-12-22]. Disponible desde: https://www.androidauthority.com/develop-android-apps-languages-learn-391008/.

24. ÁVILA, Alberto. *Así es como Google está luchando contra la fragmentación de Android: Project Treble, Android One y la apertura a más dispositivos de la beta de Android P son algunas de las armas de Google para luchar contra la fragmentación.* [online]. 2018 [visitado 2018-12-23]. Disponible desde: https://www.unocero.com/smartphones/asi-es-como-google-esta-luchando-contra-la-fragmentacion-de-android/.

25. SAMAT, Sameer. *Android 9 Pie: Powered by AI for a smarter, simpler experience that adapts to you* [online]. 2018 [visitado 2018-12-23]. Disponible desde: https://www.blog.google/products/android/introducing-android-9-pie/.

26. *Paneles de control* [online]. 2018 [visitado 2018-12-23]. Disponible desde: https://developer.android.com/about/dashboards/.

27. DIETER BOHN Aaron Souppouris, Dan Seifert. *iOS: A visual history* [online]. 2013 [visitado 2018-12-23]. Disponible desde: https://www.theverge.com/2011/12/13/2612736/ios-history-iphone-ipad.

28. *Xcode 10 Release Notes* [online] [visitado 2018-12-23]. Disponible desde: https://developer.apple.com/documentation/xcode_release_notes/xcode_10_release_notes.

29. HUGUET, Kristin. *Apple Reports Record First Quarter Results: iPhone, Apple Watch, Services & Apple TV Drive All-time Record Revenue* [online]. 2016 [visitado 2018-12-23]. Disponible desde: https://www.apple.com/newsroom/2016/01/26Apple-Reports-Record-First-Quarter-Results/.

30. *Choosing a Membership* [online] [visitado 2018-12-23]. Disponible desde: https://developer.apple.com/support/compare-memberships/.

31. *Swift 4: The powerful programming language that is also easy to learn.* [online] [visitado 2018-12-23]. Disponible desde: https://developer.apple.com/swift/.

32. *App Store* [online]. 2018 [visitado 2012-12-23]. Disponible desde: https://developer.apple.com/support/app-store/.

33. *Acerca del contenido de seguridad de iOS 12* [online]. 2018 [visitado 2012-12-23]. Disponible desde: https://support.apple.com/es-es/HT209106.

34. *Save data in a local database using Room* [online]. 2018 [visitado 2019-01-02]. Disponible desde: https://developer.android.com/training/data-storage/room/.

35. *Retrofit* [online] [visitado 2019-01-02]. Disponible desde: https://square.github.io/retrofit/.

36. GILIBETS, Laia. *Qué es la metodología Kanban y cómo utilizarla* [online]. 2013 [visitado 2018-12-23]. Disponible desde: https://www.iebschool.com/blog/metodologia-kanban-agile-scrum/.

37. LAPTICK, Sergey. *How & Why to Use the Kanban Methodology for Software Development* [online]. 2016 [visitado 2019-01-08]. Disponible desde: https://www.sitepoint.com/how-why-to-use-the-kanban-methodology-for-software-development/.

38. *Qué es el método Kanban y por qué funciona en la programación de software: El método Kanban fue creado para optimizar la producción de automóviles, pero ¿cómo se aplica al desarrollo de software? Desgranamos en qué consiste y sus partes fundamentales.* [online]. 2018 [visitado 2018-12-24]. Disponible desde: https://bbvaopen4u.com/es/actualidad/que-es-el-metodo-kanban-y-por-que-funciona-en-la-programacion-de-software.

39. ALICANTE, Universidad de. *Modelo vista controlador (MVC)* [online] [visitado 2019-01-01]. Disponible desde: https://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html.

40. HAKEEM, Husayn. *Android by example : MVVM +Data Binding -¿Introduction* [online]. 2017 [visitado 2019-01-01]. Disponible desde: https://medium.com/@husayn.hakeem/android-by-example-mvvm-data-binding-introduction-part-1-6a7a5f388bf7.

41. *Leaflet - a JavaScript library for interactive maps* [online] [visitado 2018-12-30]. Disponible desde: https://leafletjs.com/.

42. *Chart.js — Open source HTML5 Charts for your website* [online] [visitado 2018-12-30]. Disponible desde: https://www.chartjs.org/.

43. *Bootstrap · The most popular HTML, CSS, and JS library in the world* [online] [visitado 2018-12-30]. Disponible desde: https://getbootstrap.com/.

44. *jQuery* [online] [visitado 2018-12-30]. Disponible desde: https://jquery.com/.

45. *Introduction: Get started with Bootstrap, the world's most popular framework for building responsive, mobile-first sites, with BootstrapCDN and a template starter page* [online] [visitado 2018-12-30]. Disponible desde: https://getbootstrap.com/docs/4.2/getting-started/introduction/.

46. *Popper.js* [online] [visitado 2018-12-30]. Disponible desde: https://popper.js.org/.

47. *Pandas: Python Data Analysis Library* [online] [visitado 2018-12-30]. Disponible desde: https://pandas.pydata.org/.

48. *xlrd* [online]. 2018 [visitado 2018-12-30]. Disponible desde: https://pypi.org/project/xlrd/.

49. *geopy* [online]. 2018 [visitado 2018-12-30]. Disponible desde: https://pypi.org/project/geopy/.

50. *django-ckeditor* [online]. 2018 [visitado 2018-12-30]. Disponible desde: https://pypi.org/project/django-ckeditor/.

51. *CKEditor: Smart WYSIWYG HTML editor* [online] [visitado 2018-12-30]. Disponible desde: `https://ckeditor.com/ckeditor-4/`.

52. *django-iprestrict* [online]. 2018 [visitado 2018-12-30]. Disponible desde: `https://pypi.org/project/django-iprestrict/`.

53. *Art. 37 GDPR: Designation of the data protection officer* [online] [visitado 2019-01-10]. Disponible desde: `https://gdpr-info.eu/art-37-gdpr/`.