



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería
Informática**

SmartBeds

**Aplicación de técnicas de
minería de datos para
detección de crisis epilépticas
y aplicación Android.
Documentación Técnica**



Presentado por Alicia Olivares Gil
en Universidad de Burgos — 23 de junio
de 2019

Tutores: Álvaro Arnaiz González y José
Franciso Díez Pastor

Índice general

Índice general	I
Índice de figuras	III
Índice de tablas	IV
Apéndice A Plan de Proyecto Software	1
A.1. Introducción	1
A.2. Planificación temporal	1
A.3. Estudio de viabilidad	12
Apéndice B Especificación de Requisitos	17
B.1. Introducción	17
B.2. Objetivos generales	17
B.3. Catalogo de requisitos	18
B.4. Especificación de requisitos	20
Apéndice C Especificación de diseño	33
C.1. Introducción	33
C.2. Diseño de datos	33
C.3. Diseño procedimental	34
C.4. Diseño arquitectónico	36
C.5. Diseño de interfaces	38
Apéndice D Documentación técnica de programación	39
D.1. Introducción	39
D.2. Estructura de directorios	39

D.3. Manual del programador	39
D.4. Compilación, instalación y ejecución del proyecto	39
D.5. Pruebas del sistema	39
Apéndice E Documentación de usuario	41
E.1. Introducción	41
E.2. Requisitos de usuarios	41
E.3. Instalación	41
E.4. Manual del usuario	41
Bibliografía	43

Índice de figuras

A.1. <i>GNU Affero General Public License v3.</i>	15
B.1. Diagrama de casos de uso, nivel 0.	21
B.2. Diagrama de casos de uso, visualización de datos.	21
B.3. Diagrama de casos de uso, administración de usuarios.	22
B.4. Diagrama de casos de uso, administración de camas.	22
C.1. Diagrama de secuencias, llamada a los servicios genéricos de la API.	35
C.2. Diagrama de secuencias, petición de datos en tiempo real.	36
C.3. Abstracción de la arquitectura de microservicios.	37
C.4. Prototipos iniciales de las pantallas de: login, administración, visualización de camas y visualización de datos.	38

Índice de tablas

A.1. Tareas del sprint 1	2
A.2. Tareas del sprint 2	2
A.3. Tareas del sprint 3	3
A.4. Tareas del sprint 4	3
A.5. Tareas del sprint 5	3
A.6. Tareas del sprint 6	4
A.7. Tareas del sprint 7	4
A.8. Tareas del sprint 8	5
A.9. Tareas del sprint 9	5
A.10.Tareas del sprint 10	6
A.11.Tareas del sprint 11	6
A.12.Tareas del sprint 12	6
A.13.Tareas del sprint 13	7
A.14.Tareas del sprint 14	7
A.15.Tareas del sprint 15	8
A.16.Tareas del sprint 16	8
A.17.Tareas del sprint 17	9
A.18.Tareas del sprint 18	9
A.19.Tareas del sprint 19	9
A.20.Tareas del sprint 20	10
A.21.Tareas del sprint 21	10
A.22.Tareas del sprint 22	11
A.23.Tareas del sprint 23	11
A.24.Tareas del sprint 24	11
A.25.Tareas del sprint 25	12
A.26.Tareas del sprint 26	12
A.27.Costes de personal.	13

A.28.Costes de <i>hardware</i>	13
A.29.Coste total.	14
B.1. Caso de uso 1: Loguear	23
B.2. Caso de uso 2: Visualizar de datos	24
B.3. Caso de uso 2.1: Elegir cama	24
B.4. Caso de uso 2.2: Ver datos en tiempo real	25
B.5. Caso de uso 3: Administrar de usuarios	26
B.6. Caso de uso 3.1: Añadir usuarios	27
B.7. Caso de uso 3.2: Modificar contraseña	28
B.8. Caso de uso 3.3: Borrar usuario	28
B.9. Caso de uso 4: Administrar de camas	29
B.10.Caso de uso 4.1: Añadir cama	30
B.11.Caso de uso 4.2: Modificar cama	31
B.12.Caso de uso 4.3: Borrar cama	31
B.13.Caso de uso 4.4: Asignar cama a usuario	32

Apéndice A

Plan de Proyecto Software

A.1. Introducción

En este apartado se va exponer la planificación temporal del proyecto, indicando qué tareas y cuándo se realizaron. Además, se presenta un análisis de la viabilidad legal y económica del proyecto.

A.2. Planificación temporal

La planificación temporal se ha realizado adaptando la metodología *Scrum* a un proyecto educativo, con los cambios que esto conlleva.

- El desarrollo se ha basado en iteraciones o *sprints* de una semana de duración aproximadamente.
- Cada uno de los *sprints* contiene las tareas o *issues* que se realizaron esa semana.
- Cada tarea tiene asociado un coste, que simboliza su dificultad en cuanto al esfuerzo que se estima invertir en ella.
- En caso de que la estimación del coste resultara inexacta al realizar el *issue*, este se modificó para reflejar el esfuerzo real empleado.
- Al finalizar cada *sprint* se realizaba una reunión de revisión con los tutores donde se exponían los progresos realizados y se planificaba el siguiente *sprint*.

Sprint 1

Fecha: 19/12/2018 - 23/12/2018

El primer *sprint* consistió en realizar una exploración bibliográfica inicial sobre el estado del arte.

<i>Issue</i>	Estimado	Final
Crear y configurar repositorio	2	2
Exploración bibliográfica inicial	13	13

Tabla A.1: Tareas del sprint 1

Sprint 2

Fecha: 23/12/2018 - 29/12/2018

Se continuó la exploración bibliográfica inicial, centrándose en artículos especialmente interesantes encontrados hasta el momento y se comenzó la exploración bibliográfica sobre otros métodos aplicables al problema.

<i>Issue</i>	Estimado	Final
Continuación de la exploración bibliográfica inicial	8	8
Exploración bibliográfica sobre otros métodos aplicables al problema	8	8
Lectura de «Automated Epileptic Seizure Detection Methods: A Review Study»	8	8
Instalar y configurar cliente VPN	2	2

Tabla A.2: Tareas del sprint 2

Sprint 3

Fecha: 29/12/2018 - 11/01/2019

Se inició la documentación y se empezó a trabajar en la visualización de los datos en bruto y de algunos datos estadísticos.

<i>Issue</i>	Estimado	Final
Iniciar documentación	-	-
Instalar entorno y librerías de Python	5	5
Aprender a usar librerías	8	8
Procesar y mostrar datos	8	8

Tabla A.3: Tareas del sprint 3

Sprint 4

Fecha: 11/01/2019 - 18/01/2019

Se configuró el acceso al equipo de cómputo del grupo de investigación para probar técnicas de reducción de la dimensionalidad de los datos y algunas opciones básicas de filtrado y suavizado de la señal.

<i>Issue</i>	Estimado	Final
Configurar acceso a gamma	5	5
Probar opciones filtrado y suavizado	8	8
Probar otras formas de proyección de datos	8	21

Tabla A.4: Tareas del sprint 4

Sprint 5

Fecha: 18/01/2019 - 25/01/2019

Se hicieron cambios en el preprocesado, se probaron otras formas de filtrado de la señal y se estudiaron los puntos clave de las proyecciones del *sprint* anterior.

<i>Issue</i>	Estimado	Final
Leer apuntes de minería de datos	8	8
Modificar preprocesado	3	3
Representar señales en torno a la crisis epiléptica	5	5
Probar formas de filtrado de la señal	5	5
Estudiar los puntos clave de las proyecciones	5	8

Tabla A.5: Tareas del sprint 5

Sprint 6

Fecha: 25/01/2019 - 31/02/2019

Se centraron las pruebas en las proyecciones con mejor rendimiento, concretamente en MDS [7], y se iniciaron la documentación de la planificación temporal y el cuaderno de investigación.

<i>Issue</i>	Estimado	Final
Cambiar a proyecciones con mejor rendimiento	13	13
Pasar cálculos estadísticos a funciones	5	5
Documentar 5 primeros Sprints en el Plan de Proyecto	8	8
Documentar investigación en Overleaf [11]	5	5

Tabla A.6: Tareas del sprint 6

Sprint 7

Fecha: 31/02/2019 - 07/02/2019

Se codificaron las transformaciones generadas en los *sprints* anteriores (normalización, filtros y estadísticas) como transformadores de Sklearn [12].

<i>Issue</i>	Estimado	Final
Aprender sobre la clase sklearn.base.TransformerMixin [5]	3	3
Generar transformadores para las funciones usadas	21	13

Tabla A.7: Tareas del sprint 7

Sprint 8

Fecha: 07/02/2019 - 14/02/2019

Se exploraron otras formas de proyección y se realizó una primera aproximación de clasificación mediante Random Forest [6] y detección de anomalías One-class [8].

<i>Issue</i>	Estimado	Final
Probar Kernel PCA	5	5
Acotar ataque a partir del aspecto de la señal y la salida de las proyecciones (MDS)	3	5
Probar MDS con el ataque reetiquetado	8	8
Probar clasificador Random Forest	8	8
Aplicar detección de anomalías one-class	13	13

Tabla A.8: Tareas del sprint 8

Sprint 9

Fecha: 14/02/2019 - 21/02/2019

Se planteó la evaluación de los clasificadores mediante el área bajo la curva ROC y se terminaron de documentar las proyecciones en el cuaderno de investigación.

<i>Issue</i>	Estimado	Final
Valorar los resultados de Random Forest mediante el área bajo la curva	3	3
Incluir las proyecciones en la documentación	5	5
Preparar la visualización de las proyecciones para la documentación	5	8

Tabla A.9: Tareas del sprint 9

Sprint 10

Fecha: 21/02/2019 - 28/02/2019

Una parte se invirtió en aprender sobre clasificación de conjuntos de datos desequilibrados mediante ensembles y por otro lado se realizó una exploración de ventanas para la aplicación de los datos al clasificador Random Forest.

<i>Issue</i>	Estimado	Final
Lectura de aprendizaje sobre datos desequilibrados	-	-
Aplicar Random Forest a datos estadísticos con distintas ventanas	8	8

Tabla A.10: Tareas del sprint 10

Sprint 11

Fecha: 28/02/2019 - 07/03/2019

Se continuó con la lectura sobre desequilibrados y se inició el aprendizaje sobre la librería tsfresh [2] para extracción de características en series temporales.

<i>Issue</i>	Estimado	Final
Continuar con la lectura sobre uso de ensembles para conjuntos desequilibrados	5	5
Extracción de características en series temporales	8	8

Tabla A.11: Tareas del sprint 11

Sprint 12

Fecha: 07/03/2019 - 14/03/2019

Se trataron de aplicar los resultados de la extracción de características de series temporales al clasificador Random Forest.

<i>Issue</i>	Estimado	Final
Random Forest con características de series temporales	5	5
Continuar extracción de características en series temporales	13	13

Tabla A.12: Tareas del sprint 12

Sprint 13

Fecha: 14/03/2019 - 21/03/2019

Principalmente se exploraron formas de filtrar y combinar las mejores características de series temporales para ser aplicadas al clasificador.

<i>Issue</i>	Estimado	Final
Filtrado de características	13	5
Aplicar Random Forest a combinaciones de las mejores características	5	5
Documentación de sprints pasados y actualización del cuaderno de investigación	5	5

Tabla A.13: Tareas del sprint 13

Sprint 14

Fecha: 21/03/2019 - 28/03/2019

Se planteó un filtrado de características mediante un algoritmo genético usando el framework DEAP [3] de python y se realizó una investigación inicial de técnicas para la implementación de servidores de *streaming*.

<i>Issue</i>	Estimado	Final
Investigar técnicas para implementar servidores de streaming	8	8
Algoritmo genético para la selección de características	13	13
Avanzar con la documentación en el cuaderno de investigación	5	13

Tabla A.14: Tareas del sprint 14

Sprint 15

Fecha: 28/03/2019 - 04/04/2019

Se mejoró el algoritmo genético, se finalizó su ejecución con la ayuda de tmux [13] y se documentaron los resultados. Además, se inició el diseño de

los requisitos y los casos de uso de la aplicación y se plantearon los primeros prototipos.

<i>Issue</i>	Estimado	Final
Generar prototipo de la pantalla de visualización de datos	5	5
Aprender sobre tmux para la ejecución del genético	3	3
Mejorar algoritmo genético y documentar resultados	8	8
Plantear primeras cuestiones de diseño de la app	5	5

Tabla A.15: Tareas del sprint 15

Sprint 16

Fecha: 04/04/2019 - 11/04/2019

Se ultimaron los detalles del cuaderno de investigación con la documentación generada hasta el momento, se finalizaron los prototipos y se documentó la parte de diseño y de las técnicas. Además, se instaló Android Studio para su uso en sprints posteriores.

<i>Issue</i>	Estimado	Final
Ultimar detalles del cuaderno e trabajo	-	-
Finalizar y documentar prototipos	5	5
Avanzar en la documentación temporal, de diseño y de las técnicas	13	13
Instalar Android Studio	2	2

Tabla A.16: Tareas del sprint 16

Sprint 17

Fecha: 11/04/2019 - 18/04/2019

Se refactorizó el código de los experimentos para incluir el testeo mediante la métrica precision-recall, más adecuada para conjuntos de datos desequilibrados, y se volvieron a ejecutar los filtrados de características.

<i>Issue</i>	Estimado	Final
Refactorizar el código de Random Forest para incluir la métrica precision-recall	8	8
Volver a ejecutar los filtrados de características para la nueva métrica	8	8

Tabla A.17: Tareas del sprint 17

Sprint 18

Fecha: 18/04/2019 - 02/05/2019

Se documentaron los resultados de los filtrados de características del sprint anterior y se comenzó la lectura sobre la documentación de Android Studio y la visualización del curso *Android Development for Beginners* de Google.

<i>Issue</i>	Estimado	Final
Documentar los resultados de las nuevas ejecuciones en el cuaderno de investigación	8	8
Aprender a usar Android Studio	13	21

Tabla A.18: Tareas del sprint 18

Sprint 19

Fecha: 02/05/2019 - 09/05/2019

Se terminó el comportamiento de la pantalla de autenticación, se generó el clasificador obtenido con el mejor conjunto de características encontrado y se continuó con la documentación de la planificación temporal.

<i>Issue</i>	Estimado	Final
Terminar el comportamiento de la pantalla de login	3	3
Extraer características deseadas con tsfresh y generar clasificador	5	5
Documentación de la memoria	3	3

Tabla A.19: Tareas del sprint 19

Sprint 20

Fecha: 02/05/2019 - 16/05/2019

Se crearon las pantallas principales de la aplicación de Android.

<i>Issue</i>	Estimado	Final
Crear la pantalla de visualización de camas	8	13
Crear pantalla de visualización de datos	13	13
Terminar pantallas de gestión de Usuarios	5	5

Tabla A.20: Tareas del sprint 20

Sprint 21

Se terminó el comportamiento general de la aplicación de Android, se solucionaron algunos bugs y se adecuaron las interfaces de usuario a la guía de estilos empleada.

Fecha: 16/05/2019 - 23/05/2019

<i>Issue</i>	Estimado	Final
Terminar comportamiento de la pantalla de visualización de datos	5	5
Pantallas de gestión de camas	8	8
Modificar estructura de las pantallas de gestión de usuarios	5	5
Bug en la visualización de los datos	3	3
Modificaciones menores de las interfaces para adecuarse a la guía de estilos	5	5
Crear menú para el rol de usuario	8	13

Tabla A.21: Tareas del sprint 21

Sprint 22

Fecha: 23/05/2019 - 30/05/2019

Se codificó el comportamiento de la aplicación ante pérdidas de conexión y ante pérdidas de la sesión por autenticación con el mismo usuario en otro dispositivo.

<i>Issue</i>	Estimado	Final
Comprobar si la sesión ha caducado y redirigir	3	3
Controlar la pérdida de conexión a internet	5	5

Tabla A.22: Tareas del sprint 22

Sprint 23

Fecha: 30/05/2019 - 07/06/2019

Se solucionaron varios bugs de la aplicación.

<i>Issue</i>	Estimado	Final
Bug en la pantalla de gráficas	5	5
Bug en la pantalla de administración cuando la lista supera la longitud de la pantalla	5	5
Cambiar cambio de contraseña de usuarios desde el administrador	2	2

Tabla A.23: Tareas del sprint 23

Sprint 24

Fecha: 07/06/2019 - 13/06/2019

Se comenzó a redactar la memoria y se eliminaron algunos bugs de la aplicación.

<i>Issue</i>	Estimado	Final
Comenzar documentación de la memoria	13	13
Bug de las gráficas al refrescar	8	8

Tabla A.24: Tareas del sprint 24

Sprint 25

Fecha: 13/06/2019 - 19/06/2019

Se terminó la primera versión de la memoria, se generó el archivo apk definitivo y se probó en distintos dispositivos compatibles.

<i>Issue</i>	Estimado	Final
Eliminar filas huérfanas y viudas	-	-
Terminar la primera versión de la memoria	13	21
Generar apk y probarla en varios dispositivos	3	3
Eliminar errores al instalar en otros dispositivos	5	5

Tabla A.25: Tareas del sprint 25

Sprint 26

Fecha: 19/06/2019 - 27/06/2019

Se añadió una pantalla «about» a la aplicación, se comentó y documentó el código usando javadoc y se terminó la documentación de los anexos.

<i>Issue</i>	Estimado	Final
Añadir pantalla de «about»	5	5
Comentar código	5	5
Generar javadoc	3	1
Terminar los anexos	21	21

Tabla A.26: Tareas del sprint 26

A.3. Estudio de viabilidad

Viabilidad económica

Para considerar la viabilidad económica del proyecto se deben calcular los costes derivados de su realización. Se van a tener en cuenta tanto el coste del personal como el del softwares y el hardware empleados. Dado que este proyecto se ha realizado de forma conjunta con José Luis Garrido Labrador, ambos calcularemos el coste asumiendo que el proyecto cuenta con dos empleados.

Costes de personal

Siguiendo estas consideraciones calculamos el coste total de personal de la siguiente forma:

Concepto	Coste
Salario mensual neto de una persona	1.225,7€ [4]
Retención IRPF (15 %)	216,3€
Seguridad Social (29,9 %)	615,06€
Salario mensual bruto de una persona	2.057,06€
Total 7 meses y dos empleados	28.798,84€

Tabla A.27: Costes de personal.

La retribución de la seguridad social se ha calculado siguiendo la guía de GoBees [9].

Costes del *software*

El *software* empleado no supone ningún coste en este proyecto ya que todas las herramientas y bibliotecas utilizadas son de código abierto o gratuitas.

Costes del *hardware*

Para el desarrollo de este proyecto no se ha adquirido ningún *hardware* nuevo, por lo que únicamente se incluirán los costes del material con el que ya se contaba asumiendo una amortización en 5 años, y calculando solo el coste de amortización correspondiente a la duración del proyecto (7 meses):

Concepto	Coste	Coste amortizado
Dispositivo móvil	150€	17,5€
Ordenador portátil (x2)	800€	93,33€
<i>MainFrame</i>	3.000€	350€
GPU (x3)	4.500€	525€
Total	8.450€	985,83€

Tabla A.28: Costes de *hardware*.

Coste total

Teniendo en cuenta los costes de personal y de *hardware*, el coste económico total del proyecto asciende a:

Concepto	Coste
Coste de personal	28.798,84€
Coste del <i>hardware</i>	985,83€
Total	29.784,67€

Tabla A.29: Coste total.

Viabilidad legal

En esta sección se hablará de la viabilidad legal del proyecto en términos de la licencia del software generado. A la hora de escoger la licencia más adecuada para nuestro software estamos limitados por las condiciones de las licencias de las herramientas o bibliotecas que hemos empleado para generarlos.

En la aplicación se han usado herramientas y bibliotecas con las siguientes licencias ordenadas de más a menos permisivas :

- MIT: Socket.IO-client Library.
- Apache 2.0: Android Studio, Gradle, Android Support Library, MPAndroidChart.

Para la fase de investigación y documentación se usan herramientas y bibliotecas con las siguientes licencias ordenadas de más a menos permisivas:

- MIT: tsfresh.
- Zero clause BSD: tmux.
- Modified BSD: Anaconda, Scikit-Learn, Jupyter Notebook.
- GPLv2: Pencil.
- GPLv3: Weka, DEAP.

Todas las licencias son compatibles con GPL, siendo GPLv3 la más restrictiva. Para este proyecto se ha decidido usar la licencia AGPLv3^[14] (*GNU Affero General Public License v3*), derivada de GPLv3 pero diseñada específicamente para asegurar la cooperación con la comunidad en el caso de software que corra en servidores de red.



Figura A.1: *GNU Affero General Public License v3*.

Apéndice *B*

Especificación de Requisitos

B.1. Introducción

Aunque este trabajo se centra sobre todo en investigación, cuando hablamos de requisitos nos referiremos a los de la aplicación Android generada. En esta sección se enumerarán los requisitos funcionales y no funcionales de la aplicación, y se definirán los casos de uso derivados.

B.2. Objetivos generales

En la memoria se exponen los objetivos generales del trabajo, los cuales, dada la naturaleza del trabajo, se centran principalmente en la fase de investigación. En este apartado nos centraremos en los requisitos relativos al último de los objetivos generales expuestos:

- Investigar sobre técnicas del estado del arte aplicadas a problemas similares.
- Aplicar técnicas de minería de datos siguiendo los pasos del Descubrimiento de Conocimiento en Bases de Datos (*KDD*) [10].
- Explorar, aplicar y comparar distintas formas de preprocesado de los datos (filtrado, normalización, transformación, etc.).
- Usar técnicas de proyección de datos a dos dimensiones para comprobar si los instancias de «crisis» son fácilmente separables de las instancias de «no crisis».

- Probar modelos de clasificación para conjuntos de datos con preprocesados basados en estadísticas simples.
- Probar modelos de clasificación para conjuntos de datos con preprocesados basados en características de series temporales.
- Probar modelos de clasificación mediante *ensembles* para conjuntos de datos desequilibrados.
- Probar detección de anomalías mediante un modelo *One-Class*.
- Comparar el rendimiento de los modelos obtenidos.
- Comparar distintas métricas usadas para evaluar el rendimiento los modelos obtenidos.
- Generar un modelo de clasificación capaz de detectar crisis epilépticas a partir de los datos disponibles.
- **Desarrollar una app de Android para mostrar la aplicabilidad del modelo de clasificación generado.**

B.3. Catalogo de requisitos

Aquí se enumeran los requisitos funcionales y no funcionales de la aplicación desarrollada para dispositivos Android. Dado que mi compañero de proyecto José Luis Garrido Labrador y yo hemos realizado dos aplicaciones (una web y una para Android) con el mismo objetivo y las mismas funcionalidades, la especificación de los requisitos se ha realizado de forma conjunta, y por lo tanto, muchos de los puntos de estos apartados coincidirán en ambos trabajos.

Requisitos funcionales

- **RF-1 Confidencialidad del sistema:** Solamente los usuarios autorizados podrán acceder al sistema.
 - **RF-1.1 Identificación de usuario:** los usuarios se identificarán con un *nickname* y una contraseña
 - **RF-1.2 Rol de administración:** existirá un usuario especial que podrá administrar el sistema completamente sin restricciones.

- **RF-1.3 Visualización de una cama:** los usuarios validados deben poder observar los datos en tiempo real de las camas disponibles.
- **RF-1.4 Restricción de acceso:** los usuarios solamente podrán tener acceso a los datos de las camas permitidas.
- **RF-1.5 Acceso completo al administrador:** el administrador debe poder acceder a los datos de todas las camas existentes.
- **RF-2 Gestión de las camas:** El administrador debe poder gestionar las camas pudiendo añadir, modificar, borrar y dar acceso a un usuario a los datos de una cama determinada.
 - **RF-2.1 Añadir cama:** el administrador debe poder añadir una nueva cama al sistema.
 - **RF-2.2 Modificar cama:** el administrador debe poder modificar los datos una cama existente.
 - **RF-2.3 Borrar cama:** el administrador debe poder borrar una cama del sistema.
 - **RF-2.4 Asignar camas a usuarios:** el administrador se encarga de decidir qué usuario puede acceder a los datos de qué cama.
- **RF-3 Gestión de los usuarios:** el administrador debe poder gestionar los usuarios pudiendo añadir, modificar y borrar. El usuario debe poder gestionar su propia contraseña.
 - **RF-3.1 Añadir usuario:** el administrador debe poder añadir un nuevo usuario al sistema.
 - **RF-3.2 Modificar usuario:** el administrador debe poder modificar los datos un usuario existente. Igualmente el usuario debe poder modificar su propia contraseña.
 - **RF-3.3 Borrar usuario:** el administrador debe poder borrar un usuario del sistema.
- **RF-4 Visualización de los datos:** los usuarios deben poder ver, de las camas disponibles, el estado actual del paciente, la probabilidad de crisis epiléptica, sus constantes vitales y las presiones.

Requisitos no funcionales

- **RNF-1 Usabilidad:** la aplicación debe cumplir estándares de usabilidad teniendo una curva de aprendizaje baja y un uso de metáforas adecuado.
- **RNF-2 Confidencialidad:** los datos de las camas, al ser en parte constantes vitales de pacientes, solamente han de ser accesibles por los usuarios permitidos.
- **RNF-3 Escalabilidad:** el sistema debe ser escalable para adaptarse de manera correcta a un incremento de carga del sistema.
- **RNF-4 Seguridad:** los usuarios deben poder identificarse sólidamente con el sistema sin que sus datos o sus credenciales (*tokens*) sean accesibles por terceros, incluso el administrador.

B.4. Especificación de requisitos

De la misma forma, en lo relativo a las funcionalidades del cliente, la especificación de los casos de uso se ha hecho de forma conjunta con mi compañero José Luis Garrido Labrador, por lo que los contenidos de este apartado coincidirán en gran medida con los suyos.

Actores

En los casos de uso se distinguen dos actores:

- **Administrador:** Tiene acceso a la gestión de usuarios, la gestión de camas y la visualización de los datos de todas las camas existentes.
- **Usuario:** Tiene acceso a la visualización de los datos de las camas que tiene asignadas y a la gestión de su propio usuario.

Diagramas de casos de uso

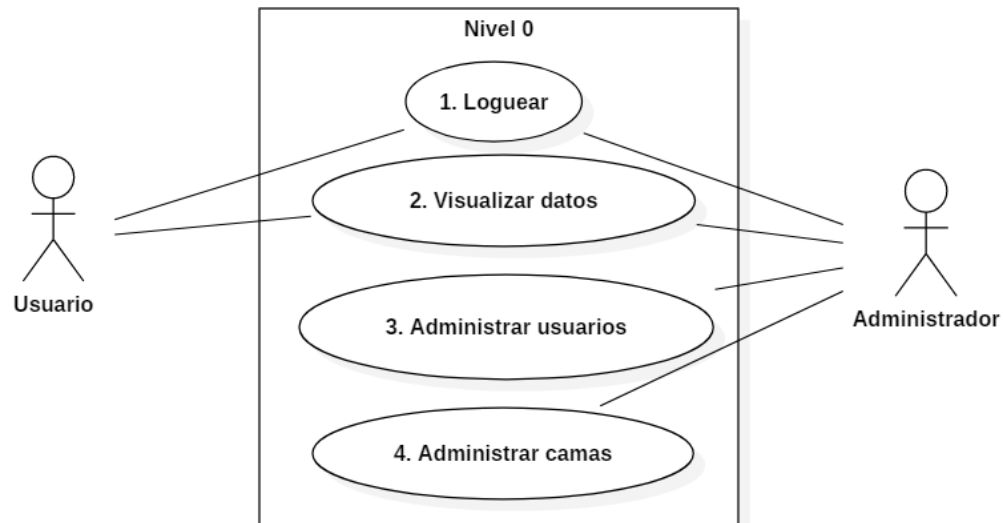


Figura B.1: Diagrama de casos de uso, nivel 0.

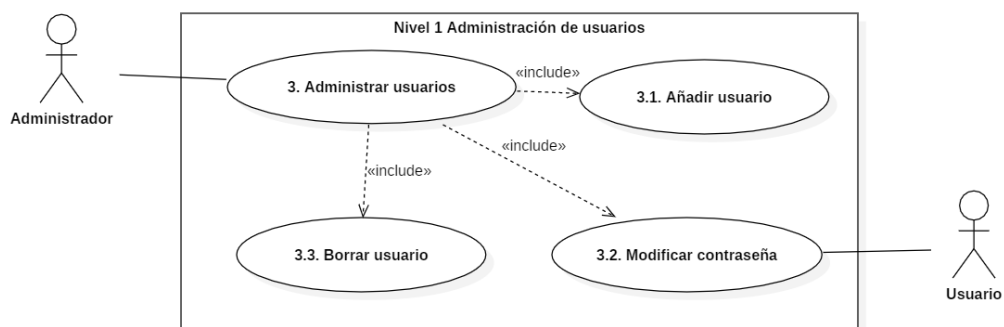


Figura B.2: Diagrama de casos de uso, visualización de datos.

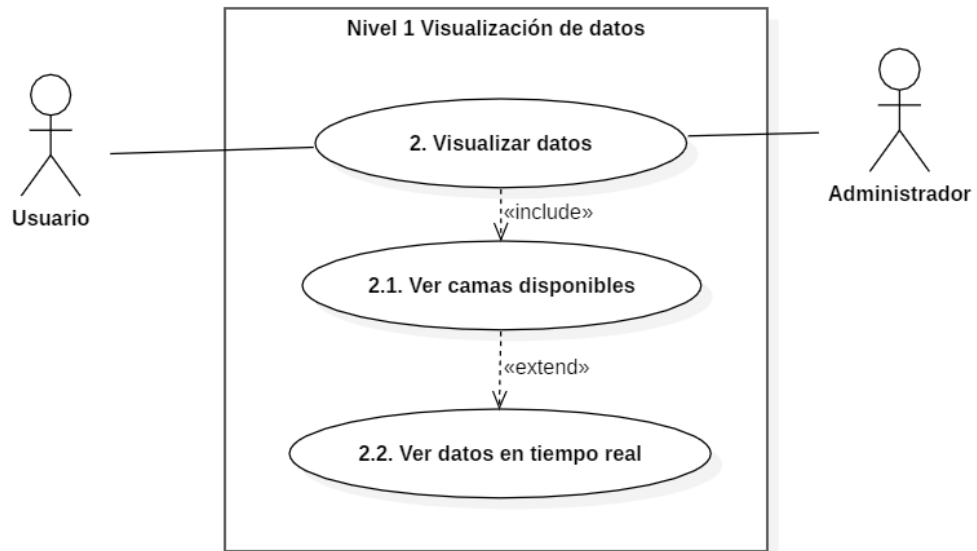


Figura B.3: Diagrama de casos de uso, administración de usuarios.

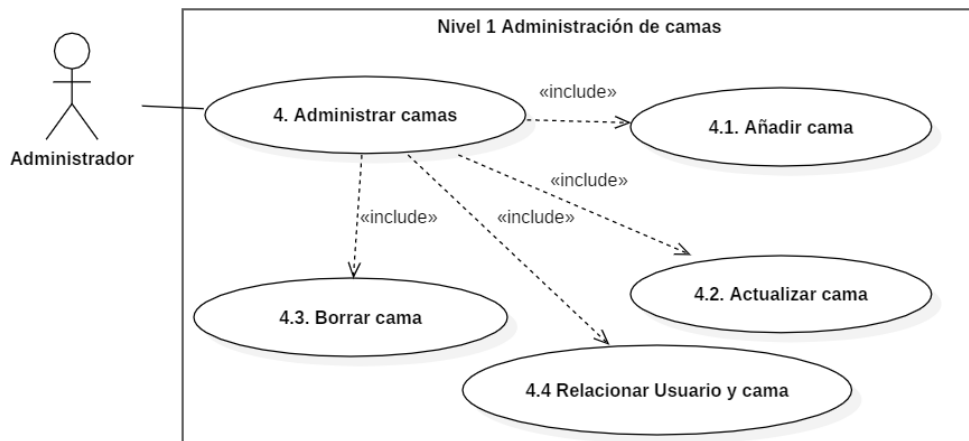


Figura B.4: Diagrama de casos de uso, administración de camas.

Especificación de casos de uso

CU-1: Loguear		
Descripción	El usuario se identifica en el sistema	
Precondiciones	No existe una sesión activa válida	
Requisitos	RF-1, RF-1.1	
Usuario	Anónimo	
Secuencia normal	Paso	Acción
	1	El cliente envía sus credenciales al servidor
	2	El servidor acepta las credenciales devolviendo el token de sesión
Postcondiciones	El usuario tiene una sesión activa válida	
Excepciones	Paso	Acción
	2	Si las credenciales son incorrectas el servidor responde con error
Frecuencia	Alta	
Importancia	Crítico	
Comentarios	Es siempre lo primero que aparecerá	

Tabla B.1: Caso de uso 1: Loguear

CU-2: Visualizar de datos		
Descripción	Ver lista de las camas disponibles	
Precondiciones	Sesión activa válida	
Requisitos	RF-1.3, RF-1.4	
Usuario	Administrador y Usuario	
	Paso	Acción
Secuencia normal	1	El cliente solicita ver las camas disponibles
Postcondiciones	El cliente está en la pantalla de camas disponibles	
Frecuencia	Alta	
Importancia	Alta	

Tabla B.2: Caso de uso 2: Visualizar de datos

CU-2.1: Elegir cama		
Descripción	Elegir cama	
Precondiciones	Sesión activa válida	
Requisitos	RF-1.3, RF-1.4, RF-4	
Usuario	Logueado	
	Paso	Acción
Secuencia normal	1	El cliente solicita ver las camas disponibles
	2	El servidor abre conexiones paralelas para actualizar en tiempo real el estado de las camas
	3	El cliente decide que cama ver
Postcondiciones	El cliente entra en la ventana de los datos en tiempo real	
Frecuencia	Alta	
Importancia	Alta	

Tabla B.3: Caso de uso 2.1: Elegir cama

CU-2.2: Ver datos en tiempo real		
Descripción	Ver datos en tiempo real	
Precondiciones	Sesión activa válida y cama existente y accesible	
Requisitos	RF-1.3, RF-1.4, RF-4	
Usuario	Administrador y usuario	
Secuencia normal	Paso	Acción
	1	El cliente solicita una nueva conexión
	2	El servidor provee una conexión en tiempo real con los datos
Postcondiciones	El usuario tiene una conexión paralela abierta con los datos en tiempo real	
Excepciones	Paso	Acción
	2	Si un paquete faltase o la señal fuera, débil se alertaría al usuario
Frecuencia	Alta	
Importancia	Máxima	

Tabla B.4: Caso de uso 2.2: Ver datos en tiempo real

CU-3: Administrar de usuarios		
Descripción	Administración de usuario: alta, baja y modificación	
Precondiciones	Sesión de administrador válida	
Requisitos	RF-3	
Usuario	Administrador	
	Paso	Acción
Secuencia normal	1	El administrador entra en el menú de administración de usuarios
Postcondiciones	El administrador está en el menú de administración de usuarios	
Frecuencia	Baja	
Importancia	Alta	

Tabla B.5: Caso de uso 3: Administrar de usuarios

CU-3.1: Añadir usuarios		
Descripción	Añadir usuarios	
Precondiciones	Sesión de administración activa	
Requisitos	RF-3.1	
Usuario	Administrador	
Secuencia normal	Paso	Acción
	1	El administrador elige añadir un nuevo usuario
	2	Se introduce un nombre de usuario para identificarlo
	3	Se introduce una contraseña dos veces
	4	Se almacenan los datos
Postcondiciones	Existe un nuevo usuario en el sistema	
Excepciones	Paso	Acción
	2	Si el nickname existiese
	3	La contraseña añadida no coincide en las dos ocasiones
Frecuencia	Baja	
Importancia	Alta	

Tabla B.6: Caso de uso 3.1: Añadir usuarios

CU-3.2: Modificar contraseña		
Descripción	Cambiar la contraseña de un usuario	
Precondiciones	Sesión activa válida, usuario existente	
Requisitos	RF-3.2	
Usuario	Administrador y Usuario	
Secuencia normal	Paso	Acción
	1	Si es usuario normal ir a 3
	2	Si es administrador elegir a qué usuario cambiar la contraseña
	3	Se introduce una contraseña nueva dos veces
	4	Se actualizan los datos
Postcondiciones	La contraseña ha cambiado	
Excepciones	Paso	Acción
	3	La contraseña añadida no coincide en las dos ocasiones
Frecuencia	Baja	
Importancia	Alta	

Tabla B.7: Caso de uso 3.2: Modificar contraseña

CU-3.3: Borrar usuario		
Descripción	Elimina un usuario de la base de datos	
Precondiciones	Sesión de administración válida, usuario existente	
Requisitos	RF-3.3	
Usuario	Administrador	
Secuencia normal	Paso	Acción
	1	Elegir a que usuario (no administrador) eliminar
	2	Eliminar usuario y todos los datos vinculados
Postcondiciones	El usuario ha sido eliminado	
Frecuencia	Baja	
Importancia	Media	

Tabla B.8: Caso de uso 3.3: Borrar usuario

CU-4: Administrar de camas		
Descripción	Administración de camas: alta, baja, modificación y asignación a usuarios	
Precondiciones	Sesión de administración válida	
Requisitos	RF-2	
Usuario	Administrador	
Secuencia normal	Paso	Acción
	1	El administrador entra en el menú de administración de camas
Postcondiciones	El administrador está en el menú de administración de camas	
Frecuencia	Baja	
Importancia	Media	

Tabla B.9: Caso de uso 4: Administrar de camas

CU-4.1: Añadir cama		
Descripción	Añadir cama	
Precondiciones	Sesión de administración válida	
Requisitos	RF-2.1	
Usuario	Administrador	
Secuencia normal	Paso	Acción
	1	El administrador elige añadir una nueva cama
	2	Se introduce el grupo multicast de la cama (IP y Puerto)
	3	Se introduce el nombre identificador
	4	Se almacenan los datos
Postcondiciones	Existe una nueva cama en el sistema	
Excepciones	Paso	Acción
	2	El grupo multicast pertenece a otra cama
	3	El nombre identificativo existe para otra cama
Frecuencia	Media	
Importancia	Crítica	
Comentarios	El grupo multicast se configura en la cama y el administrador solamente debe conocerlo, no configurar la cama física	

Tabla B.10: Caso de uso 4.1: Añadir cama

CU-2.2: Modificar cama		
Descripción	Modificar los datos de la cama	
Precondiciones	Sesión de administración válida, cama existente	
Requisitos	RF-2.2	
Usuario	Administrador	
Secuencia normal	Paso	Acción
	1	Se elige que cama modificar
	2	Se actualizan los datos a conveniencia del administrador según CU-4.1
	4	Se actualizan los datos
Postcondiciones	Los datos de la cama se modifican	
Excepciones	Paso	Acción
	2	Mismas excepciones que en CU-4.1
Frecuencia	Baja	
Importancia	Alta	

Tabla B.11: Caso de uso 4.2: Modificar cama

CU-4.3: Borrar cama		
Descripción	Elimina una cama de la base de datos	
Precondiciones	Sesión de administrador válida, cama existente	
Requisitos	RF-2.3	
Usuario	Administrador	
Secuencia normal	Paso	Acción
	1	Elegir a que cama eliminar
	2	Eliminar cama y todos los datos vinculados
Postcondiciones	La cama ya no está en la base de datos	
Frecuencia	Baja	
Importancia	Media	

Tabla B.12: Caso de uso 4.3: Borrar cama

CU-4.4: Asignar cama a usuario		
Descripción	Permite a un usuario ver los datos de una cama o quitar ese permiso	
Precondiciones	Sesión de administración válida, cama y usuario existentes	
Requisitos	RF-2.4	
Usuario	Administrador	
Secuencia normal	Paso	Acción
	1	Elegir cama
	2	Elegir usuario
	3	Si la relación existe se puede eliminar el permiso
	3	Si la relación no existe se puede crear el permiso
Postcondiciones	El usuario tiene acceso a la cama, o pierde el mismo	
Frecuencia	Media	
Importancia	Crítica	

Tabla B.13: Caso de uso 4.4: Asignar cama a usuario

Apéndice C

Especificación de diseño

C.1. Introducción

En esta sección se va a hablar de los aspectos más relevantes del diseño de la aplicación de Android.

- En primer lugar, se hará una breve mención al diseño de los datos.
- A continuación, en el apartado de diseño procedimental se expondrán detalladamente los dos procesos más relevantes de la aplicación, y en los que se basa fundamentalmente su comportamiento, obviando otros procesos más sencillos o basados en objetos propios del *framework* de Android cuyo funcionamiento puede encontrarse en la documentación.
- Después se expondrán las características de la arquitectura general de la aplicación, considerando dos puntos de vista que pueden servir para definir de forma complementaria la arquitectura general del sistema.
- Por último se expondrán los prototipos de la interfaz de usuario generados antes del desarrollo de la aplicación.

C.2. Diseño de datos

Para el desarrollo de la aplicación Android no podemos hablar estrictamente de un diseño de los datos, ya que se trata de un cliente que no interactúa directamente con la base de datos, sino que lo hace a través de la API del servidor, la cual devuelve siempre todos los datos en un objeto con formato JSON.

En este sentido, podemos abstraernos del diseño e implementación concretas de la base de datos, ya que nuestra gestión de los datos solo depende de las características del JSON que nos va a devolver cada petición. La estructura de la respuesta de cada petición a la API se especifica en el Manual del programador de mi compañero José Luis Garrido Labrador.

C.3. Diseño procedimental

Como se explica más adelante, el funcionamiento general de la aplicación se basa en la comunicación con la API del servidor remoto, el cual proporciona el modelo de datos y la lógica de negocio. Por esta razón, los procesos más relevantes corresponden con los que permiten realizar la comunicación de la aplicación con la API del servidor remoto.

Tal y como se ha planteado la aplicación de Android, existen dos procesos relevantes que es importante entender, ya que supondrán la base del funcionamiento de la aplicación:

- En primer lugar, la **llamada a los servicios genéricos de la API**. Todas las interacciones con la lógica de negocio de la aplicación (iniciar sesión, visualizar usuarios, añadir usuarios...) se realizan de esta manera. Por esta razón, para garantizar el buen funcionamiento de la aplicación, este proceso debe incluir una comprobación sobre si existe conexión a internet, si la sesión está activa y si la respuesta de la API es la esperada.

Esta interacción se representa en el diagrama de secuencias de la figura C.1. El elemento *Activity* corresponde con cualquiera de las clases que extienden de *AppCompatActivity* desde las que se realiza una petición a la API del servidor.

- En segundo lugar, la **petición y recepción de los datos de una cama en tiempo real**. Este proceso se realiza a través de conexión a nivel de *sockets* con el servidor y la gestión de eventos empleando la librería Socket.IO.

Esta interacción se representa en el diagrama de secuencias de la figura C.2. Los mensajes `give_me_data` y `package` son eventos de Socket.IO definidos por el programador de la API.

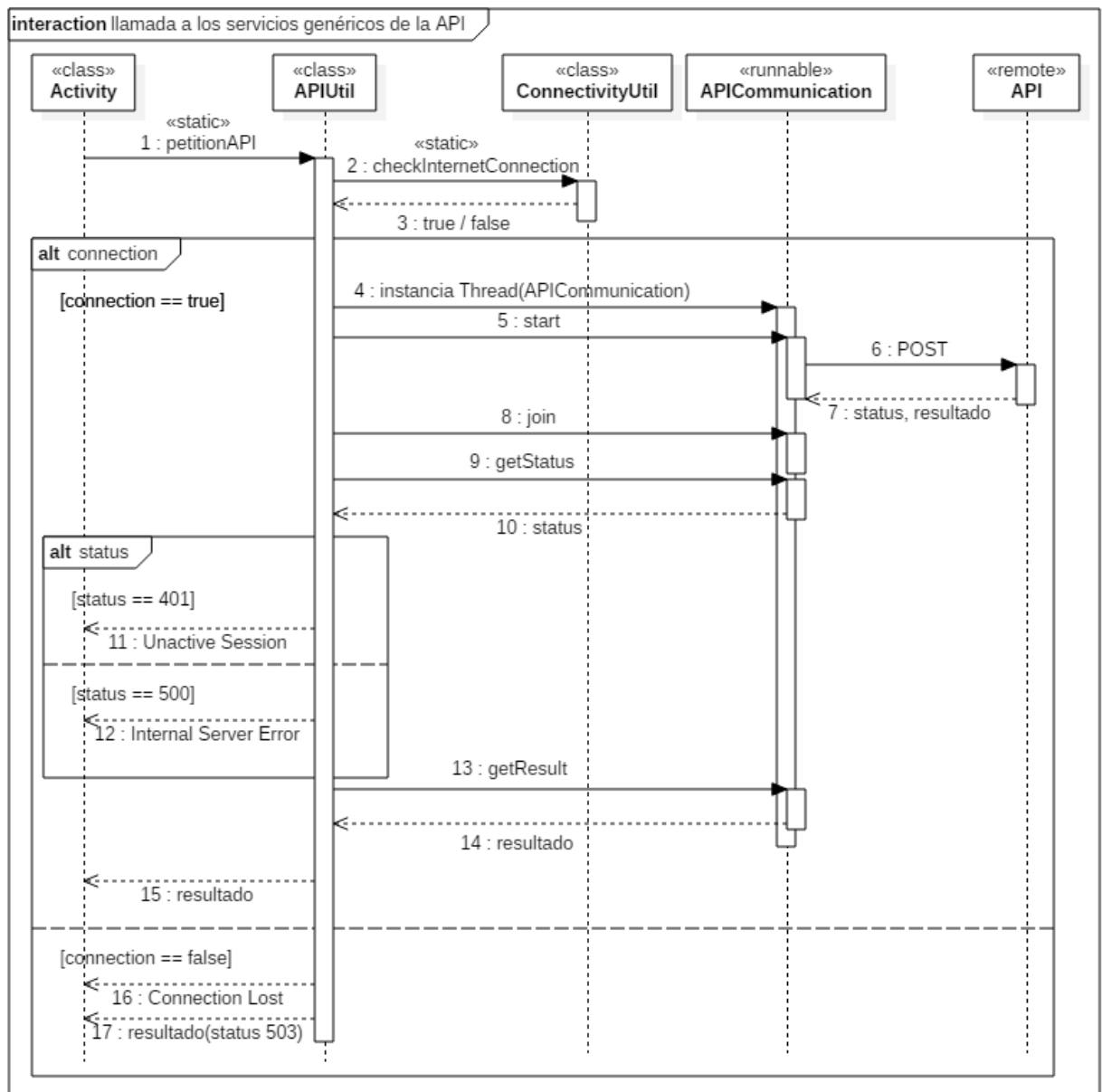


Figura C.1: Diagrama de secuencias, llamada a los servicios genéricos de la API.

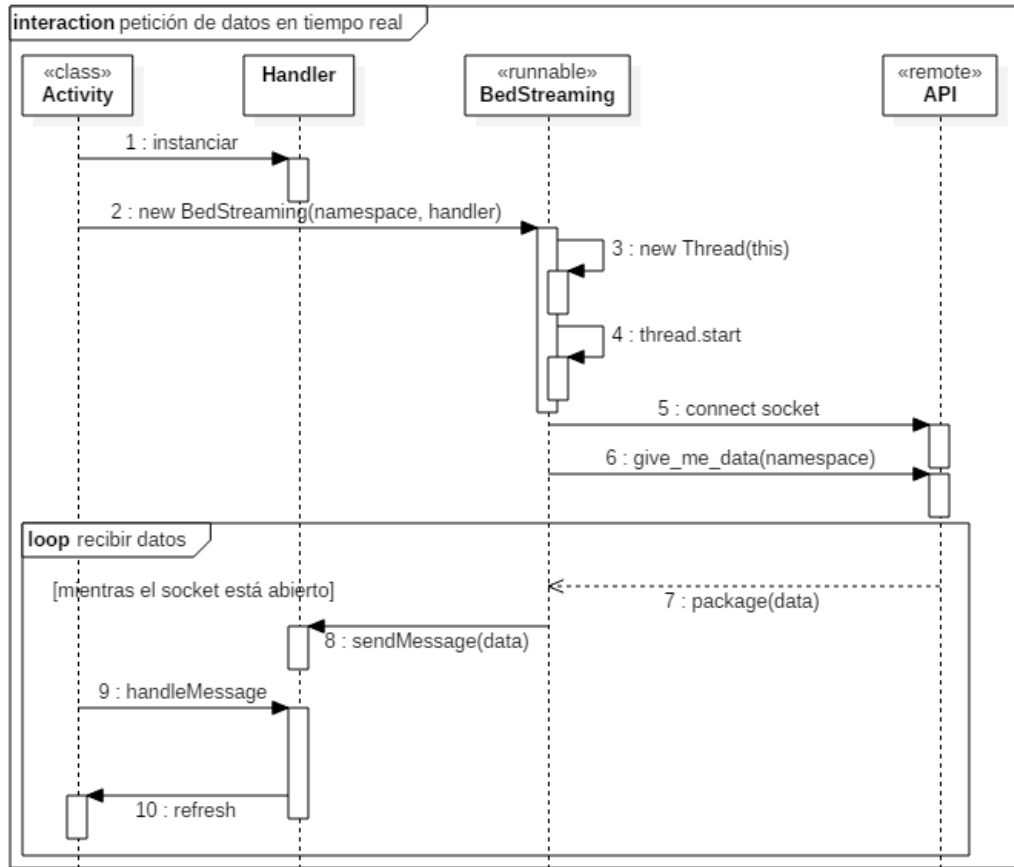


Figura C.2: Diagrama de secuencias, petición de datos en tiempo real.

El funcionamiento del resto de procesos es fácilmente deducible a partir del código o corresponde con el uso de elementos propios del *framework* de Android. La especificación y funcionamiento de estos elementos se puede consultar en la página web oficial de *Android Developers* [1].

C.4. Diseño arquitectónico

La arquitectura de la aplicación puede verse desde varios puntos de vista. En el contexto del patrón arquitectónico Modelo-Vista-Controlador (MVC) [15] definimos tres componentes:

- **Modelo:** Contiene la estructura de los datos que maneja el programa e interactúa con la persistencia para realizar consultas y actualizaciones.

- **Vista:** Presenta los datos del modelo y la interacción con la lógica de negocio de forma adecuada para el usuario.
- **Controlador:** Responde a eventos generados por la interacción del usuario con la vista y realiza peticiones de consulta/actualización al modelo. Se puede ver como un intermediario entre ambos componentes.

Según la definición de los componentes de este patrón, la aplicación Android podría ser considerada únicamente como un componente «Vista» de un sistema más grande formado también por los servicios proporcionados por la API del servidor remoto, la cual proporciona el acceso a la persistencia y la comunicación entre el modelo de datos y la aplicación.

Por otro lado, cuando se habla de este tipo de aplicaciones, es común encontrarnos con el término «**arquitectura de microservicios**». Según esta arquitectura, cada funcionalidad se encuentra contenida en un proceso del servidor al que llamaremos microservicio. Cada microservicio encapsula un solo aspecto de la lógica de negocio, y los microservicios son procesos independientes entre sí. Tal y como ocurre en esta aplicación Android, el cliente se limita a hacer peticiones a los microservicios de la API del servidor remoto, los cuales, basándose en la lógica de negocio, le proporcionan los datos necesarios para actualizar la «vista» o interfaz de usuario.

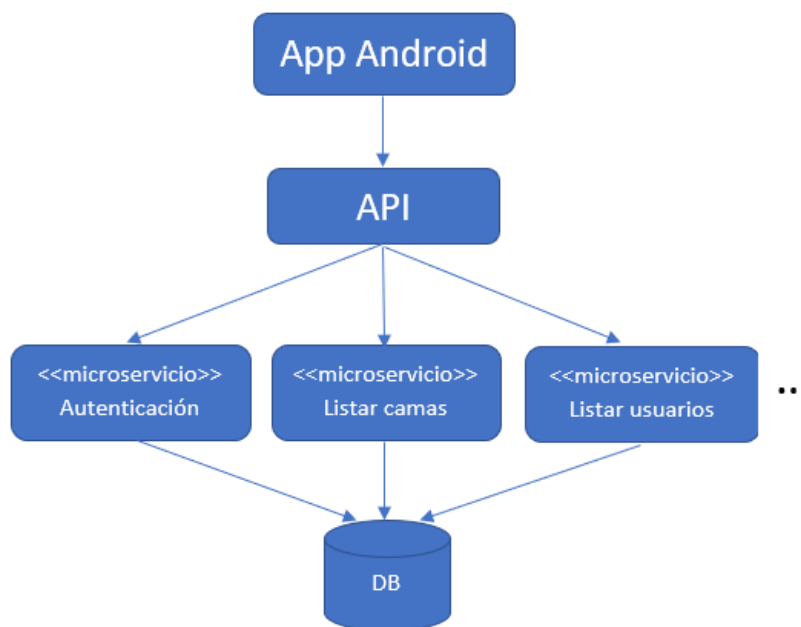


Figura C.3: Abstracción de la arquitectura de microservicios.

C.5. Diseño de interfaces

Inicialmente se realizaron una serie de prototipos básicos en los que se plasmaron las principales funcionalidades de la aplicación, sin prestar especial atención a los aspectos estéticos de la misma. Para ello se usó la herramienta de prototipado Pencil, ya que permite incorporar elementos propios de la guía de estilos que se ha seguido para el diseño de las interfaces de usuario: *Material Design*.



Figura C.4: Prototipos iniciales de las pantallas de: login, administración, visualización de camas y visualización de datos.

Apéndice D

Documentación técnica de programación

- D.1. Introducción
- D.2. Estructura de directorios
- D.3. Manual del programador
- D.4. Compilación, instalación y ejecución del proyecto
- D.5. Pruebas del sistema

Apéndice E

Documentación de usuario

- E.1. Introducción
- E.2. Requisitos de usuarios
- E.3. Instalación
- E.4. Manual del usuario

Bibliografía

- [1] Android Developers. Android developers. <https://developer.android.com/>. [Internet; descargado 17-junio-2019].
- [2] Maximilian Christ et al. tsfresh, 2019. [Internet; accedido 09-mayo-2019].
- [3] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175, jul 2012.
- [4] Indeed. Salarios para empleos de a.c.s. informáticos en España. <https://www.indeed.es/cmp/A.c.s.-Inform%C3%A1ticos/salaries>, jun 2019.
- [5] Scikit learn developers. sklearn.base.transformermixin, 2018. [Internet; accedido 09-mayo-2019].
- [6] Scikit learn developers. sklearn.ensemble.randomforestclassifier, 2018. [Internet; accedido 09-mayo-2019].
- [7] Scikit learn developers. sklearn.manifold.mds, 2018. [Internet; accedido 09-mayo-2019].
- [8] Scikit learn developers. sklearn.svm.oneclasssvm, 2018. [Internet; accedido 09-mayo-2019].
- [9] David Miguel Lozano. Gobees - monitorización del estado de una colmena mediante la cámara de un smartphone. anexos. <https://github.com/davidmigloz/go-bees/blob/master/docs/latex/anexos.pdf>, feb 2017.

- [10] Usama M. Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From data mining to knowledge discovery in databases. *AI Magazine*, 17:37–54, 03 1996.
- [11] Overleaf. Overleaf documentation, 2019. [Internet; accedido 09-mayo-2019].
- [12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [13] tmux. tmux - home, 2019. [Internet; accedido 09-mayo-2019].
- [14] Wikipedia. Gnu affero general public license — wikipedia, la enciclopedia libre, 2017.
- [15] Wikipedia. Modelo–vista–controlador — wikipedia, la enciclopedia libre. <https://es.wikipedia.org/w/index.php?title=Modelo%E2%80%93vista%E2%80%93controlador&oldid=116617617>, 2019.