Facultad
de
Ciencias

Desarrollo de una aplicación móvil para la realización de experimentos en el aula

(Development of a mobile application for conducting experiments in the classroom)

Trabajo de Fin de Grado
para acceder al

GRADO EN INGENIERÍA INFORMÁTICA

Autor: Balduino López Arce

Directora: Marta Elena Zorrilla Pantaleón

Septiembre – 2016

Índice de contenido

Agradecimientos 6
Resumen 7
Abstract 8
1. Introducción 9
1.1 Antecedentes y objetivos 9
1.2 Metodología 9
1.3 Plan de trabajo y presupuesto 10
2. Tecnologías y herramientas usadas 11
2.1 Tecnologías 11
2.1.1 Java 11
2.1.2 Android 11
2.1.3 C# 11
2.1.4 MySQL 11
2.1.5 Sockets 11
2.1.6 JDBC 12
2.2 Herramientas 12
2.2.1 Eclipse 12
2.2.2 Xamarin 12
2.2.3 MySQL Workbench 12
2.2.4 Junit 12
2.2.5 SonarQube 12
2.2.6 Project 2016 12
2.2.7 GitHub 12
2.2.8 MagicDraw UML 13
3. Especificación de requisitos 13
3.1 Captura de requisitos 13
3.2 Requisitos funcionales 13
3.3 Requisitos no funcionales 16
4. Diseño del sistema 16
4.1 Diseño arquitectónico 16
4.2 Diseño detallado 17
5. Implementación 18
5.1 Aplicación servidor 19
5.2 Cliente de administración 19
5.3 Aplicación móvil 21

5.4 Protocolo de comunicación 25
5.5 Gestión de la configuración 26
6. Pruebas 26
6.1 Pruebas Unitarias y Funcionales 26
6.2 Pruebas de integración 26
6.3 Pruebas de aceptación 27
6.4 Medición de calidad 27
7. Conclusiones y trabajos futuros 30
7.1 Conclusiones 30
7.2 Trabajos futuros 31
Bibliografía 32

Índice de figuras

Figura 1. Diagrama de trabajo de una metodología ágil. 10
Figura 2. Diagrama de Gantt que recoge el plan de trabajo 11
Figura 3. Diagrama de despliegue del sistema. 17
Figura 4. Modelo de datos del sistema 18
Figura 5. Organización de paquetes del servidor 19
Figura 6. Organización de paquetes del cliente de administración 19
Figura 7. Interfaz de bienvenida, menú principal 20
Figura 8. Formulario para la creación de un experimento 20
Figura 9. Lista de los experimentos (vacía) 21
Figura 10. Información del experimento 21
Figura 11. Organización de paquetes del cliente móvil. 22
Figura 12. Pantalla de bienvenida/login 23
Figura 13. Interfaz del Experimento Beauty Contest 23
Figura 14. Interfaz del experimento Fondo público y privado 24
Figura 15. Interfaz de agradecimiento 25
Figura 16. Resumen de calidad del servidor 27
Figura 17. Resumen de calidad del servidor tras las refactorizaciones 28
Figura 18. Resumen de calidad del cliente 28
Figura 19. Resumen de calidad del cliente tras las refactorizaciones 29
Figura 20. Resumen de calidad de la app 29
Figura 21. Resumen de calidad de la app tras refactorizaciones. 30

Índice de tablas

Tabla 1. Requisitos comunes a todo el sistema. 14
Tabla 2. Requisitos del tipo de experimento Oferta y Demanda. 15
Tabla 3. Requisitos del tipo de experimento Ultimatum 15
Tabla 4. Requisitos del tipo de experimento Fondo Público y Privado 15
Tabla 5. Requisitos del tipo de experimento Beauty Contest 15
Tabla 6. Requisitos no funcionales del sistema 16

Agradecimientos

Me gustaría agradecer a toda mi familia por animarme y apoyarme siempre en todas las decisiones que he tomado, y haber sido mi principal ayuda a lo largo de toda mi vida.

También agradecer a mis amigos y compañeros de clase por su apoyo.

Por último, pero no por ello menos importante, agradecer a todos los profesores que he tenido a lo largo de la carrera por todos los conocimientos, aptitudes ganadas y buenas prácticas transmitidas a lo largo de estos 4 años, entre los que cabe destacar la profesora Marta Zorrilla, por haber sido mi guía a lo largo de este proyecto.

Resumen

Los recientes avances en el diseño de experimentos en el aula, en concreto en el área de economía, han ayudado a los profesores a transmitir los conceptos esenciales de la teoría económica de una manera más convincente y comprensible a sus estudiantes. Profesores del área de microeconomía de la UC, utilizan regularmente estos experimentos en el aula con una gran aceptación y aprovechamiento por parte de sus alumnos. Pero, actualmente, el método utilizado (generalmente, papel) supone un gran esfuerzo humano por el número de profesores que deben estar en el aula para realizar de forma ágil los experimentos y computar sus resultados. Por ello, el objetivo de este proyecto es el desarrollo de una aplicación informática que permita a los alumnos realizar estos experimentos con su móvil o tableta y tener los resultados de sus decisiones en tiempo real. Este sistema software constará de una aplicación web para la configuración de los experimentos y análisis de los resultados y una aplicación móvil para participar en los experimentos previamente definidos.

La aplicación móvil se programará para Android por ser el sistema operativo más extendido y la aplicación servidora se construirá con software libre. Se utilizará Xamarin para la creación de la aplicación móvil, al ser una tecnología actualmente en auge y muy potente. El servidor será codificado en Java, al ser un lenguaje orientado a objetos utilizado particularmente para la creación de aplicaciones cliente-servidor. La base de datos residirá en MySQL, ya que es un gestor relacional con licencia GPL.

Palabras clave: experimentos de economía, aplicación móvil, innovación educativa

Abstract

Recent advances in the design of experiments in the classroom, particularly in the area of economy, have helped teachers to transmit the basic concepts of economic theory to learners in a more convincing and understandable way. Lecturers of microeconomics at University of Cantabria, conduct these experiments as learning activities carried out in the classroom. These are widely accepted and used by their students but regrettably, nowadays, the method used (usually paper) requires a huge human effort to perform experiments and compute their results nimbly. Therefore, these lecturers requested the development of a mobile application that allows students to perform these experiments with their mobile or tablet and get the results of their decisions in real time. In order to meet this need, a server application to design experiments will be programmed for teachers and an APP to participate in experiments will be developed for students.

APP will be programmed for Android operating system with Xamarin and Web application will be coded in Java. MySQL database will be used to store data.

Keywords: experiments in Economics, mobile application, innovations in education

1. Introducción

A continuación, se expondrá brevemente el contexto en el cual se ha realizado el presente trabajo fin de grado, sus objetivos y la metodología seguida. Así mismo se presentará el plan de trabajo seguido y el presupuesto.

1.1 Antecedentes y objetivos

Este proyecto se desarrolla al amparo de un proyecto de innovación docente financiado por la Universidad de Cantabria que lleva por título "*Consolidación del uso de experimentos como recurso didáctico en la enseñanza de la economía. Implantación de un sistema inalámbrico de respuesta interactiva*". El objeto de este proyecto es diseñar e implementar un sistema informático para la creación y realización de experimentos en el ámbito de la economía para los alumnos de los grados de Economía y de Administración y Dirección de Empresas.

Actualmente dichos experimentos se realizan o bien manualmente en el aula, en soporte papel, lo cual requiere de varios profesores para llevarlo a cabo, o bien en la sala de ordenadores usando el software ZTree (1), lo que conlleva el problema de la disponibilidad de la sala de ordenadores y la limitación de los experimentos al número de puestos. Cabe mencionar que el número de alumnos en estas asignaturas puede llegar a alcanzar hasta los 200 alumnos.

Como alternativa a las dos opciones anteriores, se pensó en simplificar y automatizar el sistema utilizando los *smartphones* de los participantes. De esta forma, los experimentos pueden realizarse sin necesidad de la sala de ordenadores, y sin la necesidad de personal docente para la recogida de datos y el control de los mismos.

Para su consecución, además de la *app* para los alumnos, será necesario la creación de una base de datos para almacenar la información de los experimentos así como de un programa para que los responsables de los mismos puedan acceder a la información de manera fácil y rápida.

1.2 Metodología

En este proyecto se ha optado por seguir una metodología de desarrollo ágil debido a que el cliente no ha especificado ninguna y se desarrollaba el producto en solitario.

Al contar el equipo de desarrollo con una única persona, no puede decirse que la metodología utilizada sea Scrum (2), pero podría considerarse una variante de la misma.

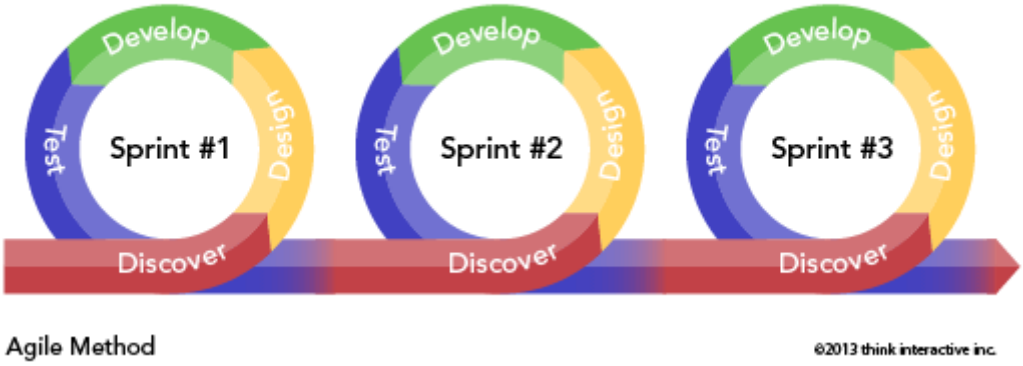

Figura 1. Diagrama de trabajo de una metodología ágil.

El proceso de la metodología a seguir, el cual puede observarse gráficamente en la figura 1, se explica brevemente a continuación.

En primer lugar, se capturaron los requisitos iniciales del sistema a través de reuniones con los clientes, en los cuales me explicaron la funcionalidad deseada por el sistema y otros aspectos relevantes del mismo. A continuación se ordenaron los requisitos y se seleccionaron los dos experimentos más prioritarios para la primera versión del producto. Se comenzó el desarrollo del producto, primero especificando los modelos software e implementándolos a continuación. Antes de su despliegue, se presentó el producto a los clientes y se recogieron detalles para mejorar la interfaz y alguna matización sobre la funcionalidad. Se realizó la fase de pruebas unitarias y funcionales por parte del desarrollador y, una vez superadas, se construyó un desplegable del producto para la prueba y validación por parte del cliente.

Este proceso de implementación/despliegue/realimentación se repitió dos veces antes de la entrega de esta memoria y se continuará realizando hasta la finalización del proyecto.

1.3 Plan de trabajo y presupuesto

Al utilizar una metodología ágil basada en iteraciones, el proyecto completo no debería tener una duración fija. Sin embargo, para mi proyecto se estimaron dos iteraciones, lo cual supondrá un total de 72 días de trabajo. Por cada iteración extra que posteriormente se desee añadir, se estimarán los días de trabajo ya que esto depende de la cantidad y complejidad de la nueva funcionalidad a añadir. Y así, hasta completar el sistema. En la figura 2, se puede apreciar el desglose de tareas para las dos iteraciones especificadas para este proyecto.

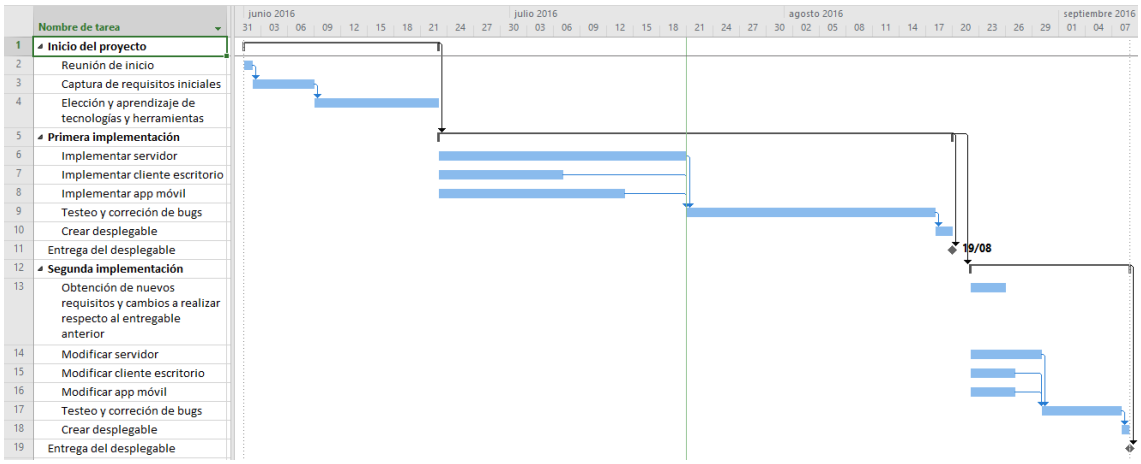

Figura 2. Diagrama de Gantt que recoge el plan de trabajo

Respecto al presupuesto, este trabajo se ha realizado disfrutando de una beca de prácticas de colaboración financiada por el mencionado proyecto.

2. Tecnologías y herramientas usadas

En este capítulo se describirán las tecnologías y herramientas usadas a lo largo de todo el proyecto.

2.1 Tecnologías

2.1.1 Java

Java (3) es un lenguaje de programación de propósito general, concurrente y orientado a objetos. Originalmente desarrollado por Sun Microsystems y actualmente propiedad de Oracle. Su uso está muy extendido en el desarrollo de aplicaciones cliente/servidor. Para ejecutar aplicaciones Java se requiere de una máquina virtual de Java (JVM).

2.1.2 Android

Android (4) es un sistema operativo generalmente utilizado en *smartphones* táctiles. Su núcleo está basado en Linux.

2.1.3 C#

C# o C *Sharp* (5) es un lenguaje de programación orientado a objetos, desarrollado por Microsoft. Su sintaxis deriva de C/C++, y posee mucha similitud con Java.

2.1.4 MySQL

MySQL (6) es un sistema gestor de bases de datos relacional, multihilo y multiusuario. Es muy conocido y utilizado debido a las ventajas que aporta, entre las que destacan ser *open source*, instalarse en multitud de sistemas operativos y permitir una fácil integración con algunos de los lenguajes de programación más utilizados como puede ser Java.

2.1.5 Sockets

Un socket (7) representa un canal de comunicación entre dos programas, los cuales pueden estar situados en distinto entorno de ejecución, siempre y cuando se tenga comunicación entre ellos. Un socket se define con la dirección IP y el puerto de los

programas de ambos extremos. La mayoría, por no decir todos los lenguajes de programación importantes, disponen de APIs (Application Programming Interface) o librerías para la utilización de los mismos.

2.1.6 JDBC

JDBC (Java Database Conectivity) (8) es una API que permite la comunicación de un programa Java con una base de datos, utilizando el lenguaje estándar SQL. En el caso del presente TFG, se utilizará la API de JDBC específica para MySQL.

2.2 Herramientas

2.2.1 Eclipse

Eclipse (9) es un IDE (*Integrated Development Environment*) de código abierto perteneciente a Eclipse Foundation. Se utilizará para el desarrollo del servidor y del cliente de escritorio del sistema.

2.2.2 Xamarin

Xamarin (10) es una API para crear aplicaciones multiplataforma en lenguaje C#. Para su utilización se dispone tanto de su propio IDE como un *plug-in* para Visual Studio. Para este proyecto, se utilizará el *plug-in* de Visual Studio.

2.2.3 MySQL Workbench

MySQL Workbench (11) es una herramienta que facilita el trabajo con el gestor de bases de datos MySQL. Permite el diseño de bases de datos y la creación y mantenimiento del sistema, entre otras funciones.

2.2.4 Junit

JUnit (12) es un conjunto de librerías usadas para la implementación de pruebas unitarias en aplicaciones Java, las cuales permitir verificar el correcto funcionamiento de la misma.

2.2.5 SonarQube

SonarQube (13) es una plataforma que permite analizar la calidad del código fuente. Entre otras opciones, analiza los duplicados (bloques de código repetidos), potenciales bugs, cantidad de comentarios, reglas de código, etc.

2.2.6 Project 2016

Microsoft Project (14) es un software desarrollado por Microsoft, que facilita la gestión y administración de proyectos. Se pueden definir tareas, hitos, recursos y asignar tareas a los recursos entre otras funciones.

2.2.7 GitHub

GitHub (15) es un repositorio de código fuente, utilizado muy comúnmente para el control de versiones de un sistema software. Es de gran utilidad a la hora de desarrollar un software ya que nos permite tanto poseer una copia de seguridad del sistema, en caso de perder la copia local, como la posibilidad de volver a versiones previas, (*rollbacks)* en caso de ser necesario.

2.2.8 MagicDraw UML

MagicDraw (16) es una herramienta CASE (Computer-Aided Software Engineering), cuya utilidad es facilitar el diseño de los modelos software en notación UML (Unified Modeling Language). Permite la creación de distintos tipos de diagramas, de entre los cuales se utilizarán el diagrama de clases, de componentes y de despliegue.

3. Especificación de requisitos

Esta sección recoge los requisitos tanto funcionales como no funcionales del sistema a desarrollar.

3.1 Captura de requisitos

Para la captura de requisitos del sistema, se ha optado por realizar una serie de entrevistas y reuniones, al ser el cliente un grupo pequeño de personas. En concreto, se llevaron a cabo dos, una con cada experto en el uso y objetivo de los diferentes tipos de experimentos. Después se procedió a redactar y validar estos requisitos con el cliente previo a su implementación. En reuniones de seguimiento se matizaban y recogía nuevos cambios y requisitos del sistema que son apuntados e incorporados en la siguiente iteración.

La lista definitiva y consensuada de los requisitos funcionales y no funcionales del sistema a desarrollar se especifica a continuación.

3.2 Requisitos funcionales

Los requisitos funcionales del sistema indican como debe comportarse y responder el mismo. Para contextualizar y que se pueda entender de forma más sencilla el funcionamiento de los experimentos, a continuación se procederá a explicar en lenguaje natural el funcionamiento de los mismos.

- *Beauty Contest* (17) (18): Experimento en el cual los participantes deben elegir un número (entre el 0 y el 100, por ejemplo) siendo el ganador aquel participante que haya elegido el número más cercano a las tres cuartas partes de la media de todos los números elegidos por los participantes. Con este juego se pretende analizar el comportamiento de los usuarios, ya que si se actuara de forma lógica, si el máximo valor fuese 100, los ¾ de la media nunca sería superior a 75, por lo que nadie debería elegir un valor mayor a 75.
- *Fondo público y privado* (17) (18): Con este experimento se pretende analizar el comportamiento de los usuarios respecto a su opinión sobre los demás. En el experimento se le provee a cada participante de una cierta cantidad de dinero (pongamos 200€ para el ejemplo). Ese dinero deben invertirlo entre dos fondos, uno en el cual no da ningún tipo de interés, pero que es privado y otro que es común y da un interés de, por ejemplo, un 50%. Al final de la ronda, la cantidad final de cada participante es lo invertido en su fondo privado más su parte equivalente del fondo público. De esta forma, lo más seguro sería invertir todo al fondo privado, ya que no perdería nada, y además se llevaría una parte del fondo público (si alguien invierte en él), pero lo óptimo sería que todos invirtieran todo al fondo público, ya que así las ganancias de todos se multiplicarían por un 50%.
- *Ultimatum* (17) (18): Este experimento se juega por parejas. Un miembro de la pareja tiene una cierta cantidad de dinero, y le hace una oferta a la otra persona. La otra persona puede aceptar o rechazar dicha oferta. Si acepta la oferta, los dos se llevan la parte del dinero que les corresponda, pero si la rechaza nadie se lleva nada. De

esta forma, se puede analizar la picardía de los participantes, observando el porcentaje que se ofrece a la otra persona, y si la otra persona prefiere no ganar nada, a dar un mayor beneficio a la otra persona.

- *Oferta y Demanda* (17) (18): Con este experimento se pretende analizar la tendencia del mercado al equilibrio. Los ofertantes tienen un costo de producción por un artículo, mientras que los compradores tienen un precio máximo a pagar por dicho artículo. De esta forma, se les deja negociar, ya que al tener costos de producción y precios máximos diferentes, no todos los compradores podrán permitirse todos los artículos, y no todos los vendedores podrán ofrecer sus productos a todos los compradores, sin conllevar pérdidas.

El conjunto de requisitos funcionales que la aplicación debe cumplir se recogen en las tablas 1 a la 5, clasificadas según el tipo de experimento.

Identificador	Descripción
RC-001	El sistema debe soportar 2 tipos de usuario: administrador y usuarios normales.
RC-002	El administrador podrá crear "instancias" de los juegos que estén dados de alta en el sistema.
RC-003	El administrador podrá dar de alta usuarios, con usuario y contraseña únicos, que serán utilizados exclusivamente para evitar que se conecten a los experimentos usuarios no deseados.
RC-004	El administrador debe poder asignar los usuarios a las instancias de los experimentos creados.
RC-005	El administrador debe poder generar el informe de cada experimento cuando desee.
RC-006	Los usuarios deben disponer de la aplicación móvil ExperimentsUC para conectarse a los experimentos.

Tabla 1. Requisitos comunes a todo el sistema.

Identificador	Descripción
ROD-001	El sistema debe poder dividir un conjunto de participantes N en 2 grupos (ofertantes y demandantes)
ROD-002	Cada ofertante o demandante se identifica con un número dentro de su grupo (ej.: Ofertante1, Demandante5, …)
ROD-003	Tras cada ronda, los ofertantes y demandantes cambian
ROD-004	El administrador puede elegir si la reorganización es completamente aleatoria, o si los ofertantes se mantienen como ofertantes y los demandantes como demandantes (entre rondas).
ROD-005	Los ofertantes tienen un coste del artículo ofrecido (cada uno puede tener uno distinto)
ROD-006	Los demandantes tienen un precio máximo a pagar por dicho artículo (cada uno puede tener uno distinto)
ROD-007	El sistema debe almacenar, por cada transacción, el identificador de los implicados, el coste del ofertante, el máximo del comprador y el precio acordado.
ROD-008	Cada ofertante sólo puede vender una vez su artículo a un comprador.

ROD-009	Para los requisitos RF-005 y RF-006, los valores pueden ser dados todos por el administrador, o bien este dar un mínimo y un máximo y que el sistema genere valores dentro de ese rango.
ROD-010	El administrador podrá solicitar un informe en el cual se le muestren todas las transacciones realizadas, así como comparativas frente a la desviación de la media de dicha ronda o al precio teórico de equilibrio

Tabla 2. Requisitos del tipo de experimento Oferta y Demanda.

Identificador	Descripción
RU-001	El sistema debe dividir un conjunto de participantes N en grupos de 2 personas.
RU-002	Una persona de la pareja dispondrá de una cantidad fija de dinero.
RU-003	La persona que dispone del dinero, realizará una oferta a la otra persona, ofreciéndola un porcentaje de ese dinero
RU-004	La otra persona podrá aceptar o declinar la oferta. En caso de aceptar, ambos se llevan la cantidad acordada, y en caso de declinar ninguno se lleva nada.
RU-005	El sistema debe almacenar las ofertas realizadas así como su resultado, y crear un informe a partir de esa información.

Tabla 3. Requisitos del tipo de experimento Ultimatum

Identificador	Descripción
RFP-001	El sistema asigna un dinero X a cada uno de los participantes
RFP-002	Los participantes pueden formar parte de un único grupo o de varios grupos más pequeños.
RFP-003	Por cada grupo, los integrantes pueden depositar su dinero en dos lugares, un fondo privado o en un fondo público.
RFP-004	Al final de cada iteración, el dinero de cada uno es el dinero de su fondo privado más su parte correspondiente del fondo público multiplicada por un factor Y. (Todos los integrantes del grupo reciben el mismo dinero del fondo público. Ejemplo: 4 integrantes, 25% del total del fondo público cada uno). Para la siguiente iteración, se reinician los fondos, es decir, se elimina todo el dinero que tenían.
RFP-005	El sistema debe almacenar, para cada grupo, lo que ha depositado cada integrante en cada fondo en cada iteración y el beneficio/pérdida de cada uno.

Tabla 4. Requisitos del tipo de experimento Fondo Público y Privado

Identificador	Descripción
RBC-001	Los participantes deben elegir un número entre el 0 y el 100
RBC-002	El sistema debe calcular la media de todos los números de los participantes
RBC-003	El sistema debe realizar un ranking de los participantes que más cerca se han quedado de los ¾ de la media.
RBC-004	El factor de ¾ utilizado en el requisito RS-003 puede variar según desee el administrador.

Tabla 5. Requisitos del tipo de experimento Beauty Contest

3.3 Requisitos no funcionales

Los requisitos no funciones imponen restricciones de diseño e implementación relativas al rendimiento, disponibilidad y seguridad, entre otros aspectos.

Los requisitos no funcionales del sistema a desarrollar se recogen en la tabla 6.

Identificador	Descripción
RNF-001	El sistema debe soportar una carga de usuarios concurrentes de hasta 300 usuarios.
RNF-002	El sistema debe ser fácilmente mantenible
RNF-003	Las interfaces de usuario a desarrollar deben ser intuitivas y de fácil aprendizaje para los usuarios finales de las mismas.

Tabla 6. Requisitos no funcionales del sistema

4. Diseño del sistema

Basados en los requisitos capturados anteriormente, se debe proceder al diseño de los modelos software que ayuden a su construcción.

En este apartado se mostrará el diseño arquitectónico y el diseño detallado especificados para este sistema.

4.1 Diseño arquitectónico

El diseño arquitectónico consiste en la aplicación de patrones y decisiones sobre cómo estructurar el código para cumplir los requisitos del sistema, así como facilitar el entendimiento y simplicidad del mismo, para futuros mantenimientos.

Al desarrollar tres productos softwares distintos (servidor, cliente de escritorio y cliente móvil), cada uno tendrá su propia arquitectura.

El servidor se diseñará siguiendo una arquitectura en tres capas, en las que se distinguen:

- Presentación: Proporciona las interfaces para trabajar con la base de datos a la aplicación de administración como a la aplicación móvil.
- Negocio: Procesa las llamadas a las interfaces y ejecuta la lógica de negocio necesaria en cada caso para dar soporte a las peticiones.
- Acceso a datos: Conecta la aplicación con la base de datos y realiza operaciones sobre ella.

Las aplicaciones tanto de escritorio como móvil se configuran en 2 capas, que son:

- Presentación: Contiene las interfaces de usuario y los mensajes que se muestran al usuario al realizar acciones sobre elementos de las mismas (pulsar un botón, modificar un campo…).
- Conexión: Contiene la representación de las interfaces del servidor y la implementación de las comunicaciones de red con el mismo.

4.2 Diseño detallado

Una vez que la arquitectura está fijada, se procede a especificar el diagrama de despliegue y el modelo de datos del sistema.

A continuación, se muestra en la figura 3 el diagrama de despliegue del sistema.

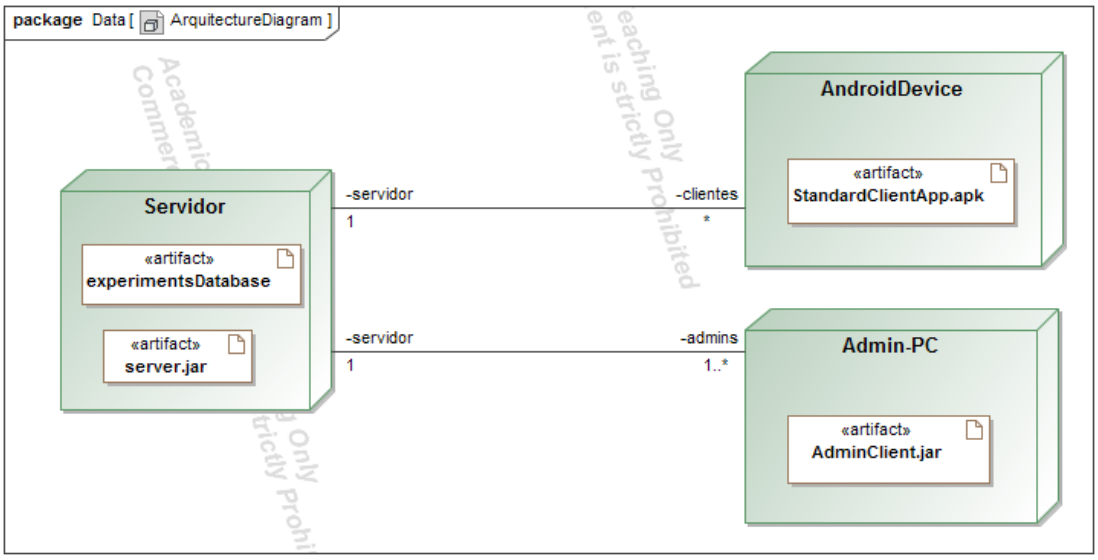

Figura 3. Diagrama de despliegue del sistema.

En este se pueden observar tres módulos:

- El primero, el servidor, alojará tanto la base de datos del sistema, como la aplicación servidora, que da soporte a las otras dos aplicaciones. Se dispondrá de únicamente un servidor.

- El segundo módulo corresponde al equipo del administrador. Se trata de un PC en el cual se ejecutará el *jar* correspondiente a la aplicación de gestión tanto para el diseño de los experimentos como para el análisis de los resultados. Pueden existir tantas máquinas administradoras como se crea necesario.

- El tercer y último módulo corresponde a los dispositivos Android, tanto teléfonos móviles como tabletas o incluso máquinas virtuales. El software que correrá en estos dispositivos será la *apk* a través de la cual se podrá participar en los experimentos.

En cuanto al diagrama de la base de datos para el almacenamiento de los datos está especificado en la figura 4.

Se ha diseñado de forma que sea lo más flexible posible a la hora de incluir nuevos experimentos y dar soporte a todos los tipos de resultado necesarios. Para incluir un nuevo tipo de experimento, simplemente sería necesario crear una nueva entrada en la tabla TipoExperimento, así como asignarle los TipoResultado a recoger por el mismo, o incluso dar de alta nuevos TipoResultado, en caso de así necesitarse.

También el modelo es flexible respecto a la asociación de usuarios a experimentos. Aunque actualmente un usuario está vinculado a un experimento concreto, igual en un futuro se quiere cambiar y que se disponga de usuarios fijos que pueden participar en varios experimentos.

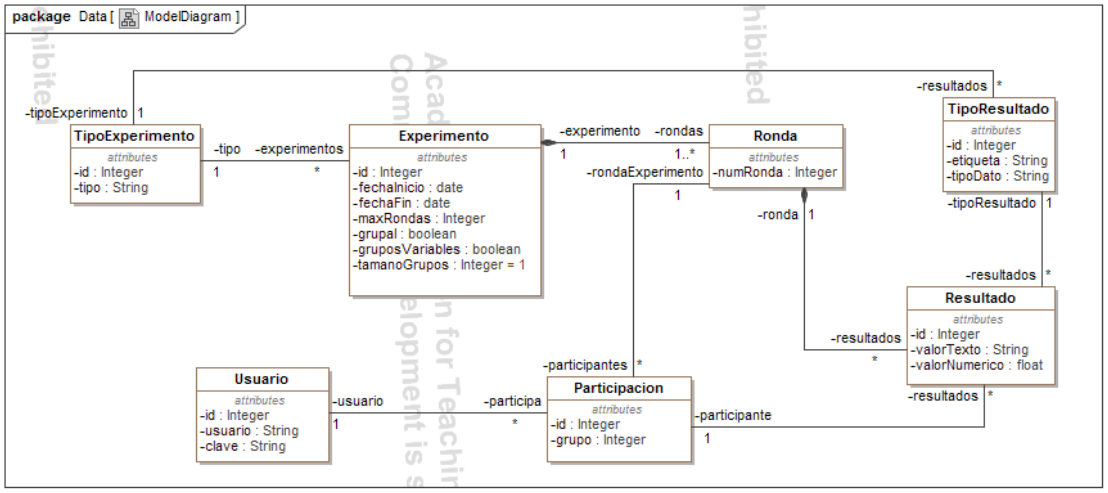

Figura 4. Modelo de datos del sistema

Para los modelos de datos de las aplicaciones, se dispone de un modelo parecido, con muy pocas variaciones, pero en los cuales se han excluido ciertos objetos que no son necesarios en la aplicación, como por ejemplo las rondas, ya que en las aplicaciones se tratan como números enteros simplemente.

5. Implementación

Una vez capturados los requisitos y especificado el diseño, es el turno de la implementación de los componentes software necesarios.

Al contar el equipo de desarrollo con una única persona, se han ido implementando los componentes software uno a continuación del otro, empezando por el servidor y terminando por la aplicación móvil.

5.1 Aplicación servidor

La codificación de la aplicación servidor, como se ha explicado anteriormente en el apartado 4.1, se divide en 3 capas. La estructura de paquetes puede observarse en la figura 5.

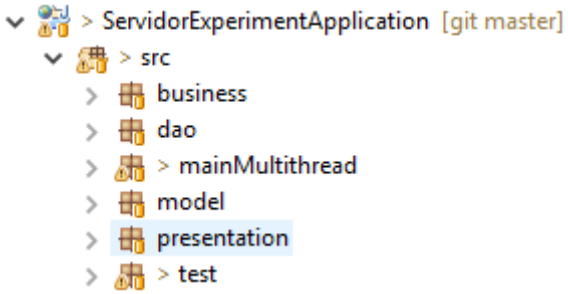

Figura 5. Organización de paquetes del servidor

Para la capa DAO, se ha hecho uso del driver JDBC para MySQL para llevar a cabo todas las consultas a la base de datos. En esta capa se han implementado todas las funciones e interacciones necesarias contra la base de datos para dar soporte a la aplicación.
La capa de negocio simplemente es la encargada de formatear los parámetros y llamar a los métodos DAO de la forma y en el orden en que sea necesario, así como confeccionar la respuesta necesaria para la capa de presentación.
La capa de presentación ofrece las interfaces para trabajar con la aplicación, proporcionando las operaciones a los clientes. A su vez, se encarga de invocar a los métodos de negocio adecuados, y aísla los métodos de administración de los métodos de los clientes "normales". Sobre esta capa, se ha añadido una interfaz TCP, que recoge las peticiones en los puertos designados e invoca a los métodos de presentación.

5.2 Cliente de administración

Para la implementación del cliente de administración se ha utilizado Java y la librería Swing para el diseño de interfaces gráficas.
Como se explicó anteriormente en el apartado 4.1, se construye una aplicación en dos capas. La estructura de paquetes del mismo puede observarse en la figura 6.

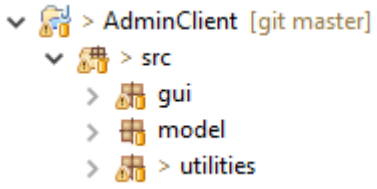

Figura 6. Organización de paquetes del cliente de administración

La capa de presentación contiene las interfaces gráficas que el usuario visualizará al utilizar la aplicación, mientras que la capa de conexión contiene la implementación de las operaciones utilizando comunicación TCP.

La interfaz de bienvenida se muestra en la figura 7. En ella el profesor podrá elegir si quiere crear un nuevo experimento o consultar la lista de los experimentos creados hasta la fecha.

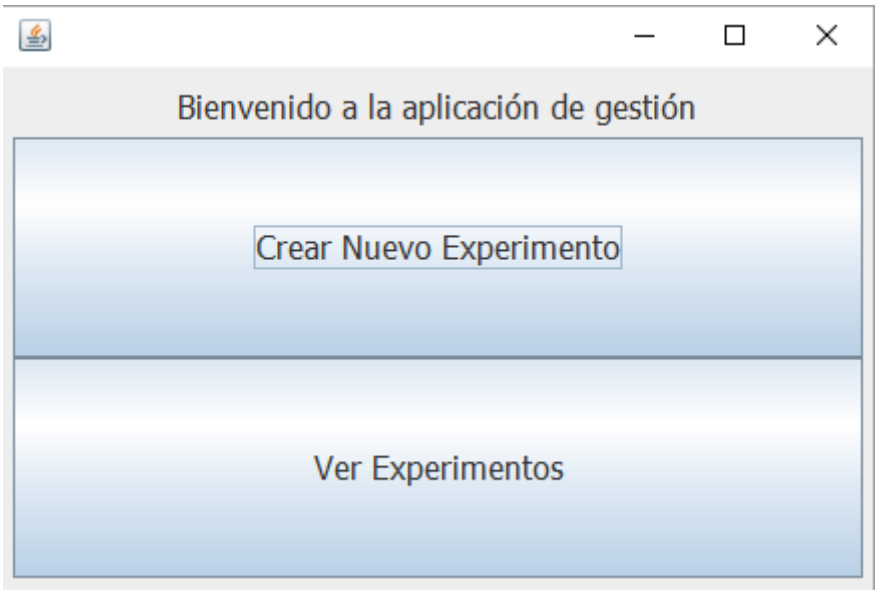

Figura 7. Interfaz de bienvenida, menú principal

Si selecciona "Crear Nuevo Experimento", lo redirigirá a la interfaz visible en la figura 8, la cual permite introducir los datos del nuevo experimento a crear. Si cambia de opinión, puede volver al menú principal haciendo clic en *Cancelar*, o confirmar la creación del nuevo experimento pulsando en *Crear*. Si ha pulsado crear, se le mostrará una nueva ventana para elegir el directorio en el que guardar los usuarios y contraseñas para los participantes de dicho experimento.

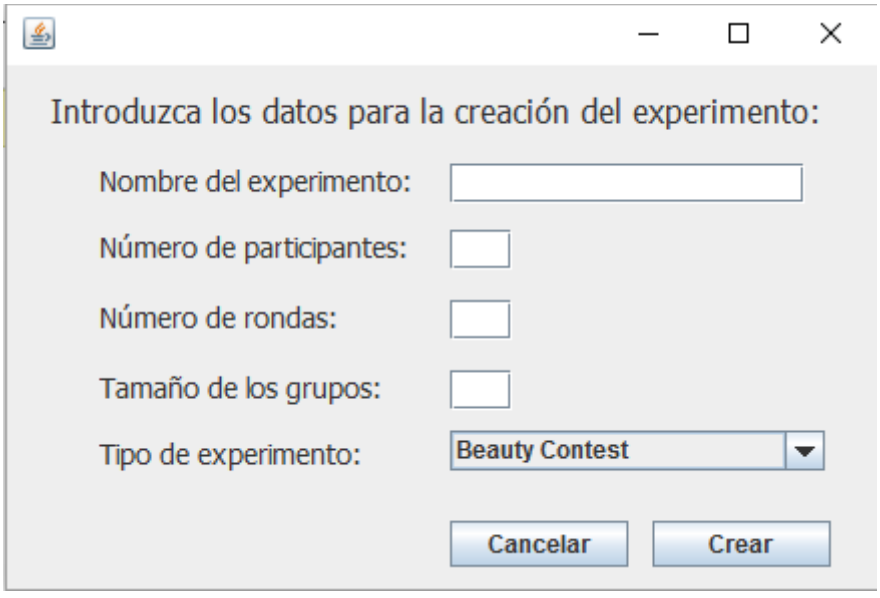

Figura 8. Formulario para la creación de un experimento

Si por el contrario, quiere ver los experimentos creados hasta el momento, debe pulsar en la opción "Ver Experimentos" del menú principal y aparecerá otra interfaz tal y como se muestra en la figura 9. Puede volver al menú principal haciendo clic en *Atrás*, o bien visualizar más información sobre un experimento pulsando sobre el mismo. Los experimentos están ordenados por fecha de creación, de más reciente a más antiguo para facilitar el acceso a los experimentos más recientes.

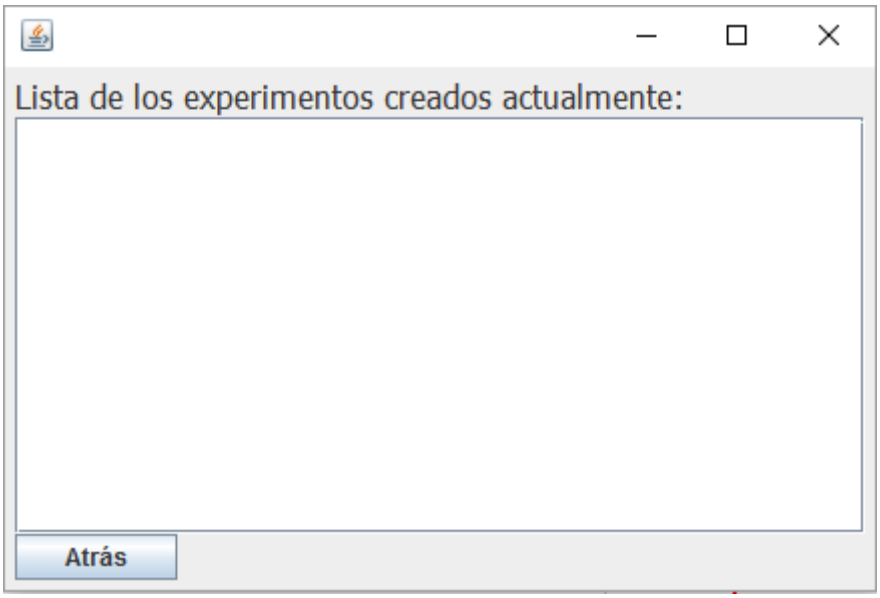

Figura 9. Lista de los experimentos (vacía)

En la figura 10 se puede apreciar cómo sería el detalle de cada experimento. Desde aquí el profesor puede o bien generar un informe con todos los resultados recogidos de dicho experimento hasta el momento, o bien volver a generar el listado de usuarios y contraseñas en caso de haberlo perdido o no haberlo generado anteriormente.

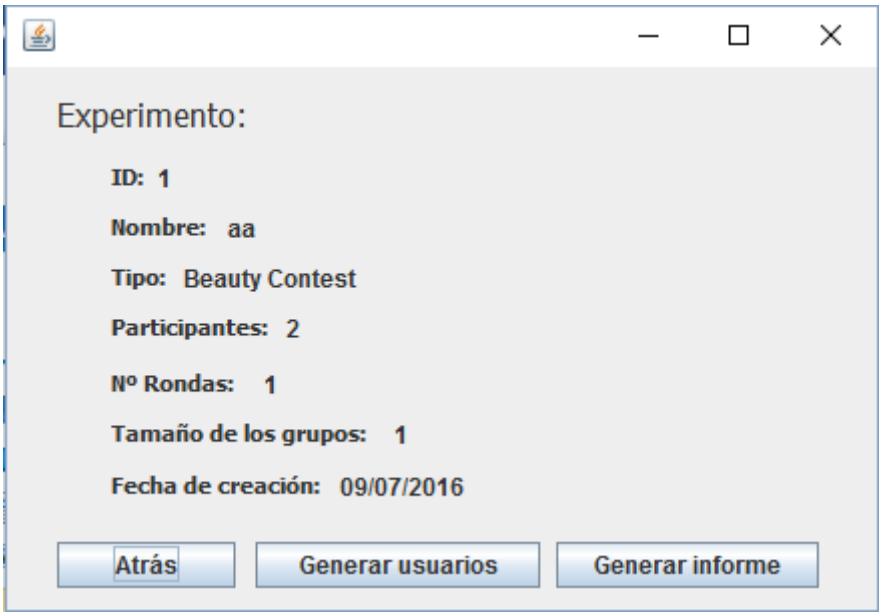

Figura 10. Información del experimento

5.3 Aplicación móvil

La aplicación móvil se ha implementado utilizando Xamarin, ya que es una tecnología emergente y orientada al desarrollo de aplicaciones móviles, utilizando el lenguaje C#. Al igual que la aplicación de administración, se ha seguido la misma arquitectura en dos capas, cuya estructura puede observarse en la figura 11.

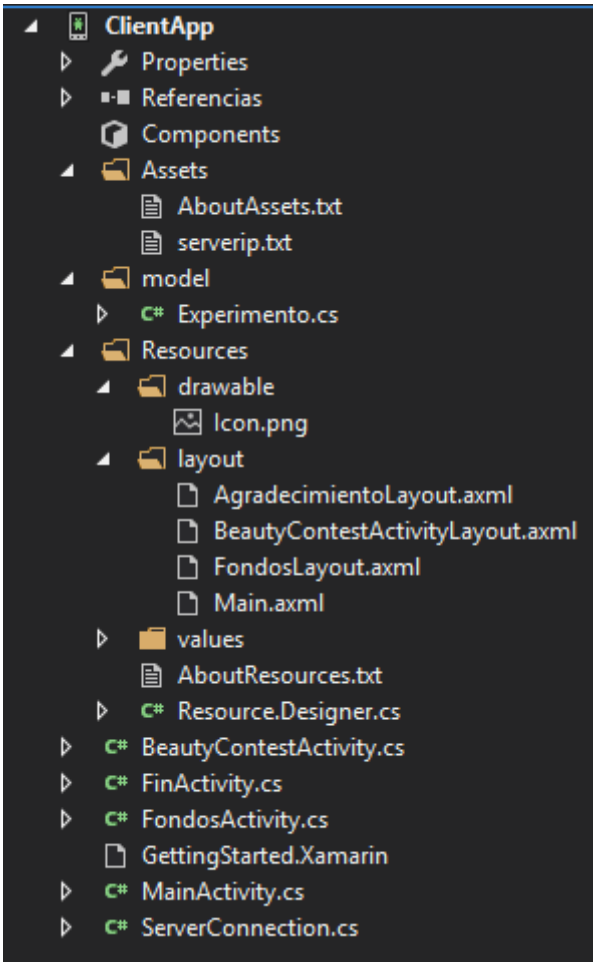

Figura 11. Organización de paquetes del cliente móvil.

Las interfaces se han diseñado de tal forma que resulten fáciles e intuitivas a los usuarios finales. Por este motivo, se ha diseñado utilizando los elementos de los formularios más adecuados en cada caso, y un tamaño de letra adecuado para la visualización de la información.

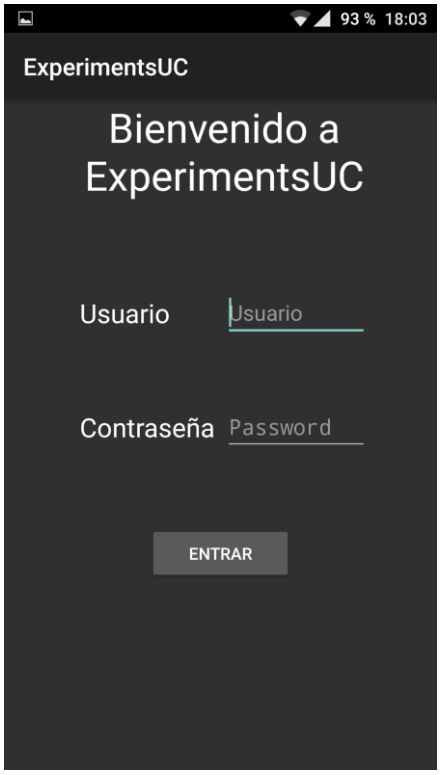

Figura 12. Pantalla de bienvenida/login

La figura 12 muestra la pantalla de inicio de la aplicación, en la cual los participantes solamente deben preocuparse de introducir las credenciales correctas, para acceder a los experimentos. Dependiendo del usuario, se dará paso a un experimento o a otro.

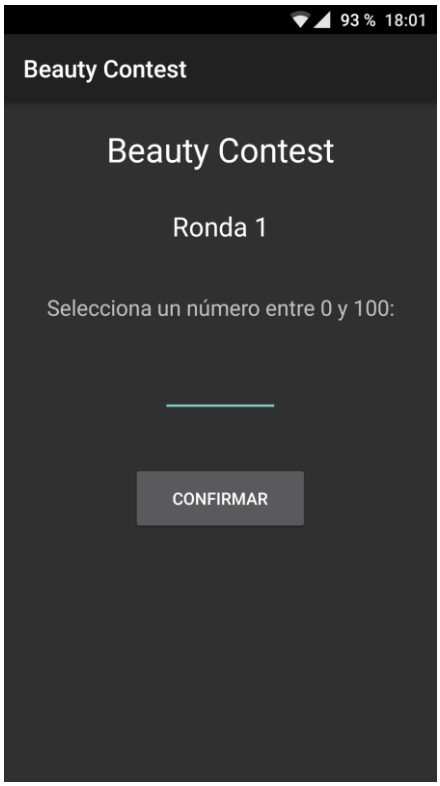

Figura 13. Interfaz del Experimento Beauty Contest

Si el usuario pertenece al tipo de experimento *Beauty Contest*, se le redirigirá a la interfaz visible en la figura 13, la cual permite introducir un número y enviarlo, así hasta completar las rondas del experimento. Una vez completado, se le redirigirá a la interfaz visible en la figura 15, mediante la cual se le agradece su participación en el experimento.

Figura 14. Interfaz del experimento Fondo público y privado

Si por el contrario, el usuario pertenece al tipo de experimento Fondo público y privado, se le mostrará la interfaz de la figura 14, la cual posee un selector para dividir su capital entre los dos distintos fondos. Al igual que con el experimento anterior, al terminar las rondas se le agradece su participación con la interfaz de la figura 15.

Figura 15. Interfaz de agradecimiento

5.4 Protocolo de comunicación

La comunicación entre los clientes y el servidor se ha implementado usando *sockets*, ya que están disponibles en la mayoría de los lenguajes de programación y es un método de comunicación básico y de bajo nivel.

Para llevar a cabo la comunicación y enviar los tipos de datos, se ha especificado un protocolo de comunicación específico.

Según el puerto al que se conecte la aplicación, accederá a las funciones de usuario normal (puerto 12003) o de administrador (puerto 12002). En ambos casos, los siguientes pasos son comunes.

Lo primero que se envía es un entero que identifica la operación que se quiere realizar, seguido de los parámetros, y posteriormente se podrá proceder a la lectura de los resultados. Por ejemplo, si desde el cliente móvil queremos verificar los credenciales y obtener el experimento al que pertenece dicho usuario, dicha operación es la número 1 de la interfaz TCP del cliente móvil, por lo que se crea una conexión en el puerto 12003. Lo primero que se envía es el código de la operación, en este caso el número entero “1”. Acto seguido, para esta operación se envía la longitud del usuario en número entero seguido de los bytes que forman el usuario, y a continuación lo mismo, pero con la clave. Tras realizar estos 5 envíos, se puede proceder a la lectura lo cual es por orden, primero un entero indicando el id del experimento (-1 o 0 en caso de ser inválido el usuario), seguido de otro entero que es el identificador del tipo, y por último el número de rondas del mismo.

Los parámetros y resultados se leen en un orden específico dependiendo de cada operación.

Para enviar tipos de datos simples (integer, float…) se envían tal cual con el tipo de dato indicado. Para enviar cadenas de caracteres, se puede o bien enviar como tipo de dato String (sólo compatible entre sockets de Java, usado para las funciones de administración) o bien enviar la longitud del String como entero, seguido de un array de bytes (usado para los clientes normales). Para enviar objetos, se envían uno a uno los atributos del mismo en un orden específico. Para enviar listas, se envía primero un entero indicando la longitud de la misma, seguido de los elementos que la componen.

5.5 Gestión de la configuración

La gestión de la configuración es una tarea imprescindible para llevar a cabo la implementación de software ya que permite disponer de un control de versiones, lo que conlleva la posibilidad de volver a anteriores versiones estables, en caso de ser necesario. También facilita el desarrollo entre varios programadores aunque en este proyecto no se ha dado esta circunstancia.

Por ello, todo el software que se ha desarrollado se encuentra disponible en GitGub (disponible en https://github.com/baldu13/TFG/), el cual su utilización ha sido muy cómoda y sencilla debido a la ayuda proporcionada por los IDEs utilizados para el desarrollo del sistema.

6. Pruebas

En esta sección, se documentarán las pruebas realizadas al software desarrollado, clasificadas en subsecciones según el tipo de prueba.

6.1 Pruebas Unitarias y Funcionales

Las pruebas unitarias permiten verificar el correcto funcionamiento de un módulo, función o método software.

En el caso del sistema desarrollado, se han probado los métodos del servidor, en especial los de la capa de persistencia, al ser estos los que más potenciales *bugs* pudieran tener. No se ha podido aislar la prueba de la base de datos, ya que había que probar el correcto funcionamiento del conjunto, así que las pruebas se han llevado a cabo contra la base de datos en un estado inicial conocido, esto es, vacía, con únicos datos que añade el script de creación de la misma.

Se ha verificado el correcto funcionamiento de las interfaces proporcionadas por el servidor, así como una comprobación de que la base de datos posee la información que debería poseer.

Se ha testeado también toda la comunicación TCP llevada a cabo entre los clientes y el servidor. Dicha comunicación se ha probado también de forma conjunta, y se ha verificado el correcto funcionamiento tanto de los clientes como del servidor.

6.2 Pruebas de integración

Las pruebas de integración permiten detectar fallos, que no han podido ser detectados anteriormente en las pruebas unitarias, debidos en la mayoría de ocasiones por

incompatibilidades entre los diferentes módulos. Afortunadamente, nuestro sistema ha pasado dichas pruebas sin complicaciones tras haber pasado anteriormente las pruebas unitarias de cada componente.

6.3 Pruebas de aceptación

Las pruebas de aceptación permiten verificar, por parte del cliente, el correcto funcionamiento del software desarrollado según unas pruebas que el mismo cliente ha especificado.

Como se ha especificado anteriormente en el apartado de la metodología, se han ido entregando progresivamente diferentes versiones del software, en las cuales el cliente ha podido tanto verificar el correcto funcionamiento del mismo, como añadir nuevos requisitos.

En alguna ocasión se ha recibido alguna característica que debía cambiarse, como por ejemplo, que los resultados a obtener fueses números reales en lugar de enteros en los tipos del experimento fondo público y *beauty contest*, los cuales han sido corregidos y verificados en las pruebas de aceptación del siguiente entregable.

6.4 Medición de calidad

Para la medición de la calidad del software desarrollado, se ha utilizado la herramienta SonarQube (13).
A continuación, se expondrán los resultados obtenidos tras el primer análisis de calidad realizado a los tres componentes software desarrollados y las medidas correctivas que se llevaron a cabo.

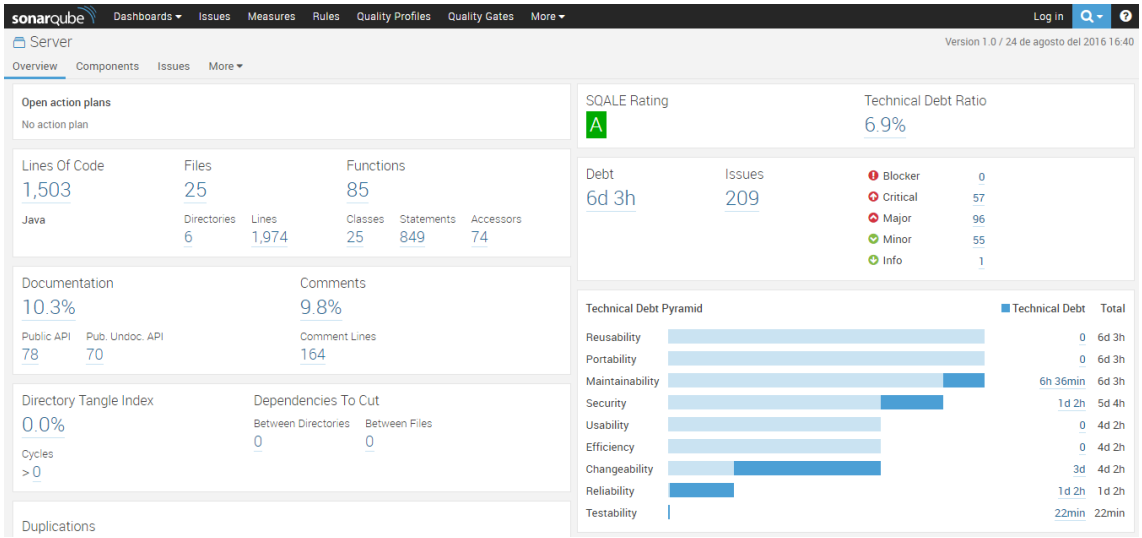

Figura 16. Resumen de calidad del servidor

Como se puede observar en la figura 16, el SQALE Rating del servidor es de nivel A, aunque una deuda técnica de 6 días se considera demasiado alta y por ello, será necesario reducirla. Tras analizar en detalle el informe, se puede observar que más del 50% de la deuda técnica proviene de la capa DAO, por lo que se realizará una refactorización de dicha capa, intentando eliminar la mayoría de *issues*. En su mayoría son cuestiones menores de fácil arreglo, como por ejemplo eliminar bloques de código comentados, reemplazar tabulaciones por espacios o eliminar variables repetidas.

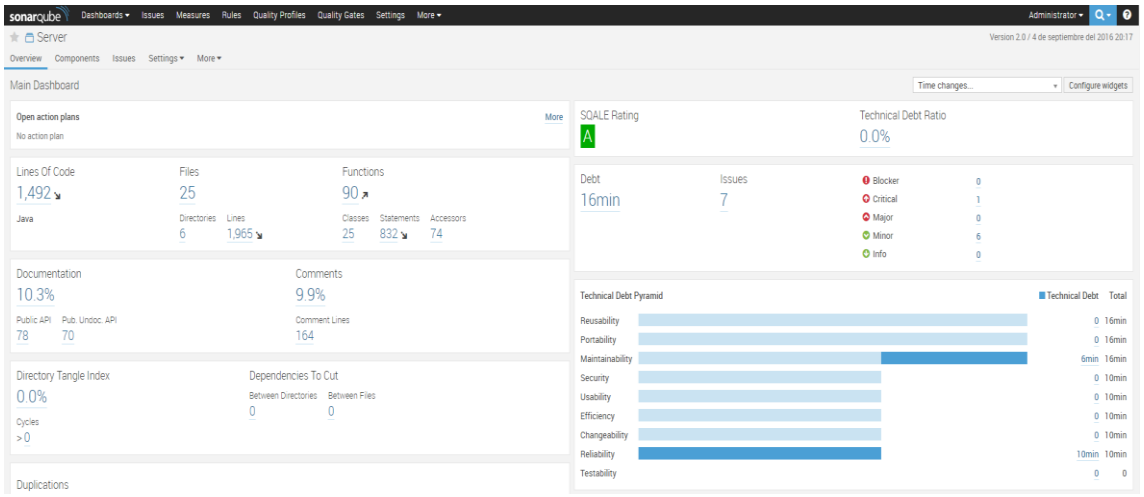

Figura 17. Resumen de calidad del servidor tras las refactorizaciones

Como se puede observar en la figura 17, tras aplicar las correcciones y refactorizaciones propuestas por Sonar, se ha conseguido reducir la deuda técnica de más de 6 días a tan sólo 16 minutos.

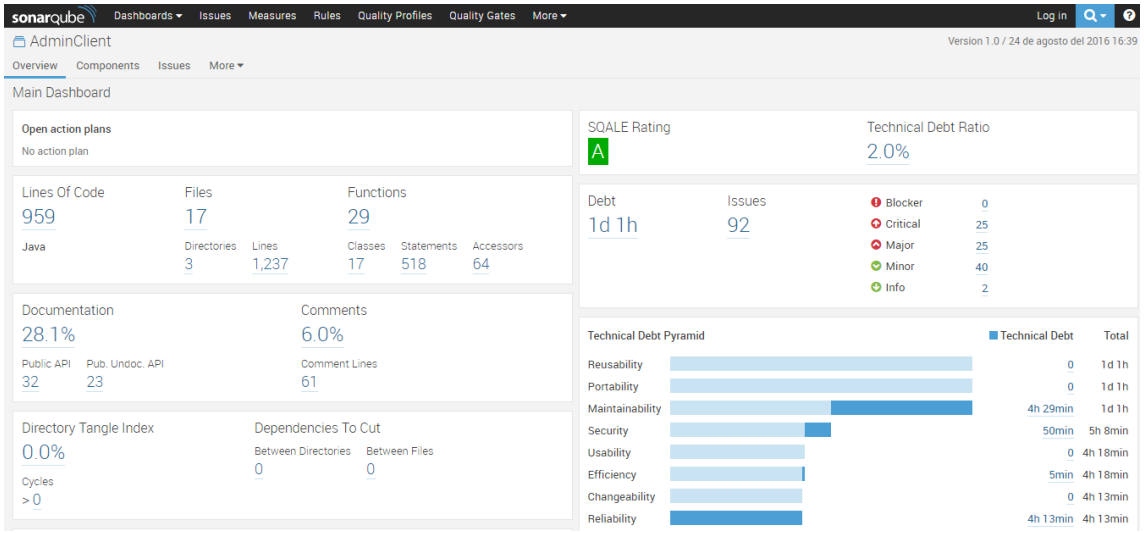

Figura 18. Resumen de calidad del cliente

Al igual que con el servidor, en la figura 18 podemos observar un resumen del informe de calidad del cliente de administración. Al igual que en el caso anterior, el SQALE Rating es de A, y la deuda técnica es mucho menor, aunque un día de deuda técnica sigue siendo muy alto, por lo que será necesaria la aplicación de correcciones en el código.

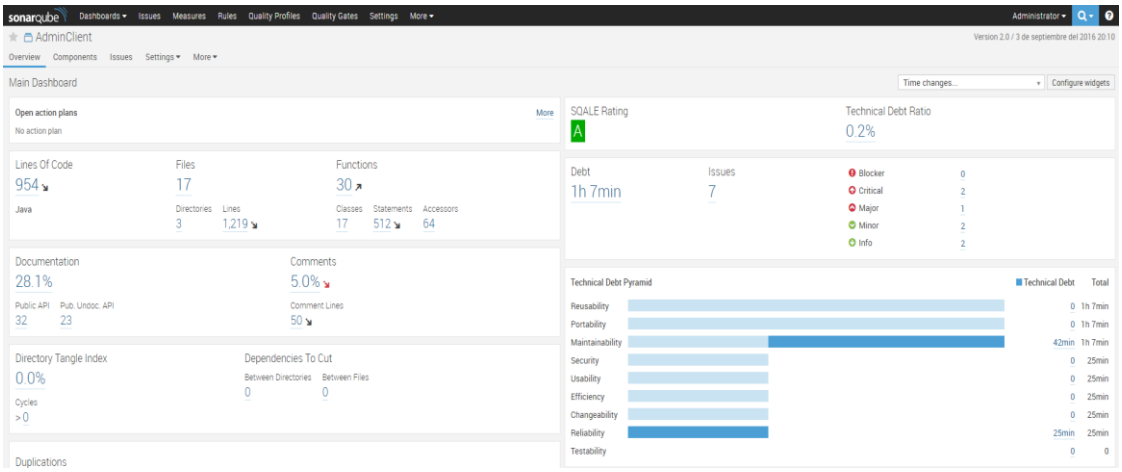

Figura 19. Resumen de calidad del cliente tras las refactorizaciones

Tras haber llevado a cabo la refactorización del cliente, se ha conseguido reducir un gran porcentaje de las *issues*, como se puede observar en la figura 19. De hecho la deuda técnica se ha reducido a una hora.

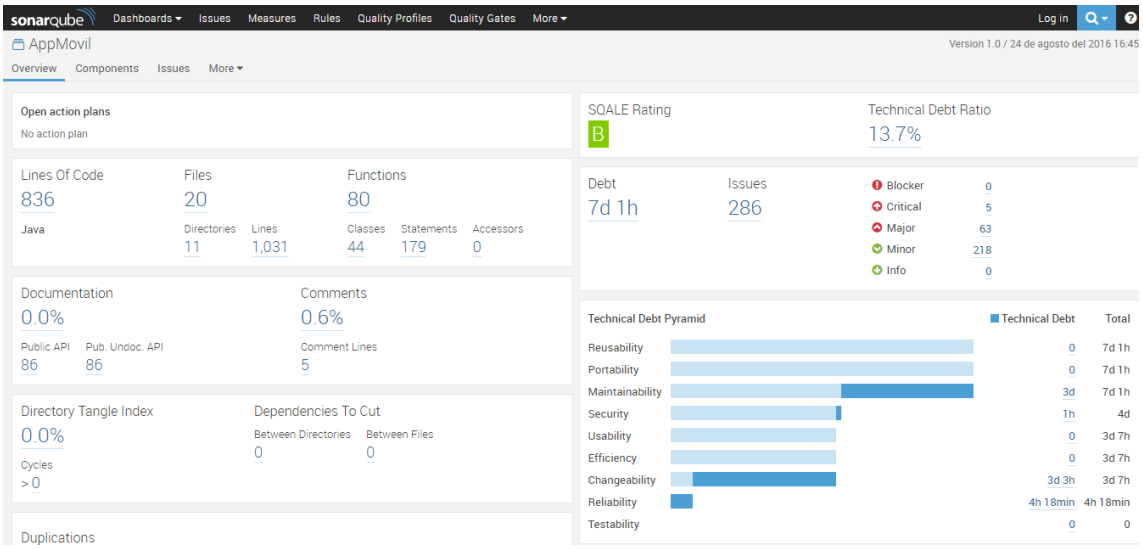

Figura 20. Resumen de calidad de la app

Por último, en la figura 20 se puede observar el resumen del informe de calidad de la *app*. En este caso el SQALE Rating es de B. Por ello, se le dio máxima prioridad a la mejora de su calidad. Tras analizar con detenimiento el informe de calidad de Sonar de la *app*, el 80% de las *issues* se resolvían renombrando variables y paquetes. El restante 20%, se debe, como se puede observar en la figura 21, a la autogeneración de código que Xamarin realiza en java a partir del C#. El propio Xamarin aconseja, mediante comentarios en las clases, no modificar dichos ficheros, por lo que las *issues* han sido marcadas como “won´t fix”. Automáticamente, el SQALE Rating ha subido a nivel A.

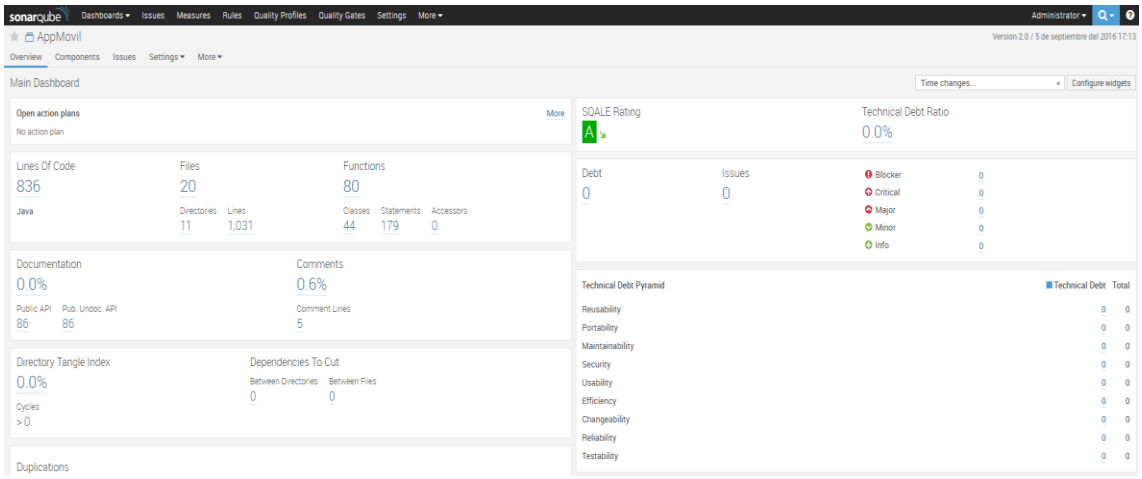

Figura 21. Resumen de calidad de la app tras refactorizaciones.

7. Conclusiones y trabajos futuros

En la presente sección del documento, se presentarán las conclusiones obtenidas tras la realización del proyecto, así como las tareas pendientes de desarrollo hasta la entrega definitiva del mismo.

7.1 Conclusiones

El presente proyecto tiene como principal finalidad la creación de un nuevo sistema mediante el cual se puedan realizar diversos experimentos relacionados con el ámbito de la economía utilizando para ello los *smartphones* de los participantes. El objetivo último es reducir el número de personal necesario en el aula para la ejecución de estos experimentos, así como eliminar la necesidad de acudir a la sala de informática de la Facultad de Ciencias Económicas y Empresariales, la cual presenta dos limitaciones: una, no permite acoger a todo el grupo de alumnos al tiempo y dos, su demanda es muy alta y por tanto no siempre está disponible.

El proyecto ha cumplido con todas las especificaciones definidas para construir un primer entregable, esto es, construir toda la infraestructura y programar los dos tipos de experimentos que fueron elegidos por los profesores de Economía. En los sucesivos entregables se incorporarán los dos tipos de experimentos pendientes de programar, lo que se estima se realice para principios de diciembre.

Se han encontrado varias dificultades a lo largo del proyecto, entre la que cabe destacar la obtención de un servidor en la UC para realizar las pruebas de aceptación, las cuales se demoraron más de un mes y aún está pendiente de estudiar cómo realizar su despliegue definitivo de forma que se cumpla con las exigencias del servicio de informática.

En cuanto a la visión personal, la realización de este proyecto me ha demostrado de forma práctica mucha de la teoría aprendida en ciertas asignaturas, como puede ser la realización de una captura de requisitos real, la dificultad de entender al cliente y de hacerme entender al ser desconocedores de la jerga informática. Por otra parte, he aprendido la importancia que tiene realizar un buen diseño de base de datos, con una perspectiva global, flexible y que permita el crecimiento del aplicativo sin incurrir en cambios de estructura. En una etapa muy

temprana de la aplicación, estaba diseñada de forma diferente, pero se ha conseguido adaptar fácilmente a la versión actual, la cual es más genérica y flexible, sin necesidad de variar para nada el diseño de base de datos. Así mismo me he demostrado mi capacidad para diseñar y desarrollar un producto software de inicio a fin, cumpliendo con todas las fases del ciclo software.

Como conclusión de este proyecto, lo más básico, pero no por ello menos importante que he aprendido, es que la realización de un proyecto no es algo trivial, que pueden surgir muchos contratiempos inesperados, y la capacidad para analizar y superar dichos contratiempos es una habilidad que un buen jefe de proyectos debe poseer, además de establecer e imponer muy buena organización y planificación.

7.2 Trabajos futuros

Como ya se ha mencionado brevemente en el apartado anterior, se han cumplido todos los requisitos iniciales propuestos para este trabajo fin de grado, pero aún quedan dos tipos de experimentos por incluir. Además, el cliente al ir observando el avance satisfactorio del mismo, ha solicitado incorporar otras funcionalidades, entre las que cabe destacar:

- Cambiar el sistema de generación de informes para que los genere en formato Excel en lugar de texto plano.
- Añadir la posibilidad de introducir el género del participante para filtrar los resultados.
- Añadir la posibilidad de ofrecer la aplicación también en lengua inglesa.
- Añadir la posibilidad de eliminar experimentos pasados para liberar espacio.
- Incluir la posibilidad de variar los grupos aleatoriamente entre rondas.

Bibliografía

1. **University Of Zurich. Zurich Toolbox for Readymade Economic Experiments. [Online] [Cited: Agosto 2016, 14.] http://www.ztree.uzh.ch/en.html.**

2. **Scrum, metodología ágil para el desarrollo de software. [Online] [Cited: Agosto 16, 2016.] https://es.wikipedia.org/wiki/Scrum_(desarrollo_de_software).**

3. Oracle. Java, lenguaje de programación. [Online] [Cited: Agosto 20, 2016.] https://www.java.com/es/about/.

4. Android, sistema operativo. [Online] [Cited: Agosto 20, 2016.] https://es.wikipedia.org/wiki/Android.

5. Microsoft. C#, lenguaje de programación. [Online] Agosto 19, 2016. https://msdn.microsoft.com/es-es/library/kx37x362.aspx.

6. MySQL, gestor de bases de datos. [Online] [Cited: Agosto 19, 2016.] https://es.wikipedia.org/wiki/MySQL.

7. Socket, tecnología de comunicación entre aplicaciones. [Online] [Cited: Agosto 22, 2016.] https://es.wikipedia.org/wiki/Socket_de_Internet.

8. Oracle. JDBC, API de comunicación. [Online] [Cited: Agosto 20, 2016.] http://www.oracle.com/technetwork/java/javase/jdbc/index.html.

9. Eclipse Foundation. Eclipse, IDE de desarrollo. [Online] [Cited: Agosto 24, 2016.] https://eclipse.org/home/index.php.

10. Xamarin, plataforma para desarrollo de aplicaciones móviles. [Online] [Cited: Agosto 16, 2016.] https://www.xamarin.com/.

11. Oracle. MySQL Workbench, herramienta para trabajar con MySQL. [Online] [Cited: Agosto 19, 2016.] https://www.mysql.com/products/workbench/.

12. JUnit, framework de pruebas unitarias. [Online] [Cited: Agosto 20, 2016.] http://junit.org/junit4/.

13. SonarQube. [Online] [Cited: Agosto 24, 2016.] http://www.sonarqube.org/.

14. Microsoft Project. [Online] [Cited: Agosto 22, 2016.] https://es.wikipedia.org/wiki/Microsoft_Project.

15. GitHub. [Online] [Cited: Agosto 19, 2016.] https://github.com/.

16. MagicDraw. [Online] [Cited: Agosto 19, 2016.] http://www.nomagic.com/products/magicdraw.html.

17. Garza, Pablo Brañas. *Economía experimental y del comportamiento.* s.l. : ANTONI BOSCH, 2011. ISBN 9788495348753.

18. Miller, Theodore C. Bergstrom / John H. *Experimentos con los principios económicos MANUAL DEL PROFESOR.* s.l. : Antoni Bosch, 2009. ISBN 978-84-95348-43-2.