

APLICACIÓN MÓVIL PARA LA ENSEÑANZA DE GENÉTICA

MARÍA TERESA CALVO RAMÓN
GIAN MARCO DÍAZ MÁRQUEZ
ANA SANZ CEPA

TRABAJO DE FIN DE GRADO EN INGENIERÍA INFORMÁTICA
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID



Curso 2015/2016

Director: RUBÉN FUENTES FERNÁNDEZ

ÍNDICE DE CONTENIDO

ÍNDICE DE CONTENIDO	3
ÍNDICE DE FIGURAS	7
RESUMEN	9
DESCRIPTORES.....	9
ABSTRACT	11
KEYWORDS	11
1. Introducción: Antecedentes, Objetivos y Plan de Trabajo.....	13
1.1. Antecedentes	13
1.2. Objetivos que se persiguen	13
1.3. Plan de trabajo	14
Introduction: History, Objectives and Planning	17
History	17
Objectives.....	17
Planning.....	18
2. Estado del arte y requisitos.....	21
2.1. Descripción funcional de las aplicaciones existentes.....	21
2.2. Tecnologías iniciales	22
2.3. Trasfondo genético	23
2.4. Trabajo relacionado	25
E-learning	25
Diseño de interfaces.....	28
2.5. Requisitos del proyecto.....	29
3. Arquitectura de la aplicación	31
3.1. Arquitectura cliente-servidor	31
Cliente	31
Servidor	32
Comunicación cliente-servidor.....	33
3.2. Modelo de datos	33
Diagrama entidad-relación.....	33
3.3. Componentes software	37
Para la aplicación móvil.....	37

Para el servidor.....	39
3.4. Comportamiento dinámico	40
Cliente	40
Servidor remoto	41
3.5. Conclusiones.....	42
4. Manual de usuario	43
4.1. Login y registro	43
4.2. Menú principal y categorías	43
4.3. Navegación	44
4.4. Teoría.....	44
Lista de teoría.....	44
Detalle de teoría.....	45
4.5. Ejercicio	45
Lista de niveles	45
Lista de ejercicios	46
Ejercicios.....	46
4.6. Test	48
Preguntas	48
Resultados	49
5. Manual de administrador.....	51
5.1. Login	51
5.2. Pantalla principal.....	51
6. Conclusiones	53
6.1. Resultados del proyecto.....	53
6.2. Trabajo futuro	54
6.3. ¿Qué hemos aprendido?	54
Conclusions	57
Project outcomes	57
Future work	58
What have we learned?	58
DESCRIPCIÓN DEL TRABAJO INDIVIDUAL.....	61
María Teresa Calvo.....	61
Gian Marco Díaz	63
Ana Sanz	65

BIBLIOGRAFÍA.....	67
GLOSARIO	69

ÍNDICE DE FIGURAS

Figura 1. Plan de trabajo. 16

Figura 2. Diagrama entidad-relación de la base de datos..... 35

Figura 3. Arquitectura cliente-servidor 40

Figura 4. Vista de login. 43

Figura 5. Vista del registro..... 43

Figura 6. Vista de la pantalla de inicio “Home” 44

Figura 7. Vista donde se muestran las categorías de cada tema. 44

Figura 8. Botón navegación hacia atrás. 44

Figura 9. Botón de acceso directo al menú principal “Home” 44

Figura 10. Vista con listado de teoría..... 44

Figura 11. Vista con detalle de teoría..... 45

Figura 12. Vista con listado de niveles de ejercicios. 46

Figura 13. Lista de ejercicios de un nivel..... 46

Figura 14. Vista inicial de ejercicio. 47

Figura 15. Vista de ejercicio con calculadora desplegada..... 47

Figura 16. Vista de pregunta de test sin resolver..... 48

Figura 17. Vista solución a pregunta de test correcta..... 48

Figura 18. Vista solución a pregunta de test incorrecta..... 48

Figura 19. Vista de resultados del test. 49

Figura 20. Login de la aplicación de administración. 51

Figura 21. Pantalla principal de administración..... 52

RESUMEN

El presente trabajo tiene su origen en la necesidad de herramientas de apoyo al aprendizaje para los alumnos en las clases de Genética de la Facultad de Biología de la Universidad Complutense de Madrid. En esta asignatura, el equipo docente ha desarrollado aplicaciones para dispositivos móviles destinadas a los alumnos. Las aplicaciones les permiten trabajar con materiales relacionados con aspectos clave de la asignatura. Estas aplicaciones contienen apartados de teoría y ejercicios. Los ejercicios cuentan con asistentes automatizados que guían al alumno para su realización y autocorrección.

En su forma actual, las aplicaciones presentan limitaciones tanto desde el punto de vista de su diseño como de la funcionalidad que ofrecen. El actual diseño no aplica las técnicas comunes de Ingeniería del Software respecto a aplicaciones cliente-servidor. Ello las hace difíciles de mantener cuando se plantea abordar nuevas funcionalidades y plataformas, o facilitar la creación de nuevos materiales de la asignatura. Ello ha limitado su expansión para incorporar nuevos tipos de materiales (en particular diferentes tipos de ejercicios), integrarlas con otras herramientas (por ejemplo, el Campus Virtual de la universidad) o permitir un apoyo efectivo a la comunidad de aprendizaje formada por alumnos y docentes (por ejemplo, para que los docentes supervisen la evolución de los alumnos y estos puedan obtener información adicional de los profesores).

Para abordar esta situación se propone una aplicación móvil que englobe a todas las aplicaciones anteriores que se habían creado para las clases de Genética. Se utilizará un modelo cliente-servidor para mejorar sus capacidades funcionales, de modo que cumpla con los requisitos establecidos. Entre estos se incluye un control de los usuarios que utilizan la aplicación, y que se optimice la memoria local utilizada por la aplicación, permitiendo así el uso de imágenes más pesadas. Además, este modelo facilitará las tareas de mantenimiento de la aplicación, por ejemplo incluir nuevo material. Por otro lado, también se propone rediseñar la interfaz de la aplicación, de modo que sea más accesible desde el punto de vista de la usabilidad.

DESCRIPTORES

Aplicaciones móviles educativas; E-learning; Aplicación móvil; App; Bioinformática; Genética; Cliente-Servidor; Ionic Framework; Apache Cordova.

ABSTRACT

This work appeared because of the need of tools that support students in the course of Genetics in the Department for Biological Sciences in the University Complutense of Madrid. The lecturers of this subject have developed over the years several applications for mobile devices aimed at their students. These applications allow them to work with materials related to key aspects of the subject, including both theoretical and practical content. The practical exercises offer students guide to make the exercise and also support their auto-evaluation, so students can know whether their answers are the correct ones and why.

The current applications developed for the course have limitations regarding both their design and the functionality they provide. The design does not apply common patterns from Software Engineering for client-server models. This makes difficult to maintain those applications when considering the addition of new functionality and target platforms, or adding more materials of the existing types. This has also limited its growth: the incorporation of new types of materials (especially different types of exercises), the integration with other tools (for example, the Virtual Campus of the university), or the functionality to support effectively the learning community of teachers and students (for example, by allowing teachers to supervise their student' development and provide them additional information) have all been constrained.

In order to address this situation, this work proposes an application that encompasses all the previous applications created by Professor César Benito. It will use a client-server model that will improve functionality, so that all the elicited requirements are met. This includes having information and statistics about the application users, and to optimize the local memory that the application needs, thus allowing the use of heavier images. Moreover, that model will also facilitate maintenance tasks, for example adding new materials. Additionally, a cleaner interface, following modern design recommendations, is also proposed for the application, making it more accessible from the user's perspective.

KEYWORDS

Learning mobile application; E-learning; Mobile application; App; Bioinformatics; Genetics; Client-Server; Ionic Framework; Apache Cordova.

1.Introducción: Antecedentes, Objetivos y Plan de Trabajo

En la siguiente sección se especificará la motivación del proyecto, así como una primera aproximación al problema planteado y la presentación del autor inicial de las aplicaciones que vamos a mejorar. Por otro lado se describirán los objetivos perseguidos al realizar dicha mejora y se detallará el plan de trabajo llevado a cabo para la realización de este proyecto.

1.1.Antecedentes

Durante la realización del Grado en Ingeniería Informática (a partir de ahora nos referiremos a él simplemente como Grado) hemos tenido la oportunidad de conocer muchos aspectos de esta profesión. A la hora de elegir el Trabajo de Fin de Grado (TFG) tuvimos en cuenta tres factores: qué habíamos aprendido, qué ramas nos atraían más y cuáles son las tendencias tecnológicas hoy en día. Se nos presentó la oportunidad de hacer una aplicación para un caso real, con usuarios de la Facultad de Biología de la Universidad Complutense de Madrid (UCM), concretamente en el Departamento de Genética.

El profesor César Benito Jiménez, Catedrático de Genética de la UCM, había desarrollado varias aplicaciones móviles sobre la plataforma Android para sus alumnos de genética. Cada una de las aplicaciones es para un tema de genética (ej. one locus, two loci o mitosis) (Benito Jiménez & Espino Nuño, 2013) (Benito Jiménez, 1997) (Benito Jiménez, 2015).

Para comprender el problema y conocer las aplicaciones que el profesor César Benito Jiménez había desarrollado, concretamos una reunión con él. En la reunión, nos dio una breve introducción a las Leyes de Mendel, las cuales son “un conjunto de reglas básicas sobre la transmisión por herencia genética de las características de los padres a los hijos” (Wikipedia, 2016), y sobre mitosis. Asimismo, nos enseñó sus aplicaciones y las limitaciones que había encontrado al desarrollarlas.

Estas aplicaciones residen completamente en el terminal del usuario. Su estructura es muy similar: una parte con la teoría del tema, otra con ejercicios propuestos y un test final. Su diseño es poco flexible, de forma que no es fácil introducir nuevos tipos de material o instancias de los tipos ya existentes. Además, todo el material del curso debe residir en el cliente. Esto causa problemas con el uso de imágenes de alta calidad. Otro problema es que se optó por un desarrollo nativo en Android, lo que impide el uso de las aplicaciones por usuarios de otras plataformas.

1.2.Objetivos que se persiguen

Las aplicaciones del profesor César Benito Jiménez presentan varios problemas estructurales y de usabilidad. Los principales inconvenientes son: datos almacenados solo en la memoria del dispositivo cliente, lo cual implica aplicaciones muy pesadas; los datos (ej. teoría y ejercicios) se introducen en el propio código, lo que da como resultado tener que generar nuevas versiones de

la aplicación ante el más mínimo cambio en sus contenidos; y una aplicación mal diseñada desde el punto de vista de la usabilidad.

Este TFG busca abordar estos problemas mediante el desarrollo de una aplicación llamada Biogam siguiendo pautas de Ingeniería del Software y técnicas de desarrollo móvil. En particular, se plantea resolver el problema con una arquitectura cliente-servidor. El servidor proporcionará servicios centralizados (ej. almacén de información) y el cliente será una aplicación para dispositivos móviles (una *app*). Se usará un desarrollo multiplataforma que se adapte automáticamente a diferentes características del dispositivo del cliente.

Además, se busca facilitar a los usuarios en sus diferentes roles el uso de la aplicación: a los profesores la creación y modificación de contenido, así como el seguimiento de la evolución de los alumnos; a los alumnos el trabajo con los materiales y la realización de los ejercicios. Esto implica varios cambios sobre las aplicaciones anteriores.

Se persigue mejorar la funcionalidad para introducir nuevos contenidos. Como se ha señalado, en las versiones antiguas de la aplicación, esto se hacía en el código. Esto dificultaba la edición y el agregado de nuevos materiales a la aplicación. En la nueva versión esta información se almacenará en una base de datos relacional. La base de datos contemplará los elementos del curso (ej. temas, teoría, ejercicios y test). Sobre ella se añadirá a la aplicación funcionalidad que facilite la recogida, edición y agregación de datos.

También se plantea rediseñar la interfaz de la aplicación para adaptarla a las nuevas técnicas de diseño utilizadas actualmente en la creación de aplicaciones. De esta forma se garantizará que los usuarios obtienen una experiencia amigable con los contenidos de la aplicación. La nueva interfaz vendrá unida a un nuevo planteamiento de la aplicación, de forma que aparte de ser una aplicación educativa, también tenga elementos que la hagan divertida y que genere en los usuarios la necesidad de continuar aprendiendo.

1.3. Plan de trabajo

Para abordar los objetivos y requisitos previos se establece el siguiente plan de trabajo (ver resumen en Figura 1):

1. *Estudio en profundidad de la aplicación.* Esta tarea cumple múltiples objetivos destinados a comprender el problema y el estado de la solución actual. Por un lado, se trata de identificar y comprender los flujos de trabajo que se pretende soportar con las herramientas. También hay que analizar la complejidad de los algoritmos internos que se utilizan para realizar las diferentes tareas en la aplicación, por ejemplo, los cálculos que se hacen en las aplicaciones y muestran al alumno. Finalmente, se persigue detectar problemas de usabilidad y diseño de la aplicación, incluidos los previamente mencionados de almacenamiento y procesamiento de imágenes. Para realizar esta tarea se estima que el equipo tendrá que trabajar a lo largo de dos semanas.
2. *Estudio del estado del arte* en cuanto a técnicas de desarrollo de aplicaciones móviles. El objetivo es determinar qué plataformas móviles son las más utilizadas actualmente, para así saber a qué sistemas operativos enfocar el desarrollo de nuestra aplicación y qué herramientas utilizar para desarrollarla. Por otro lado, realizando este estudio se quiere identificar cuáles son las tendencias en diseño de aplicaciones en cuanto a interfaces de

usuario.

En esta etapa se procederá también a la selección de las tecnologías para la creación de la app y la persistencia de los datos. En todos los casos se obtendrá información tanto de la revisión de aplicaciones y plataformas reales como de la bibliografía existente. El tiempo estimado para la realización de esta tarea es de un mes, en el que cada uno de los miembros del equipo realizará una búsqueda de artículos y noticias relacionados con los distintos aspectos del estado del arte que se quieren investigar.

3. *Diseño de un prototipo inicial* con herramientas de prototipado rápido para rediseñar la interfaz de usuario teniendo en cuenta los detalles de usabilidad que previamente estaban ausentes. Después se procederá a validar el resultado con el cliente. Este proceso tendrá una duración de un mes. Se plantea realizar varias iteraciones de forma que el cliente pueda validar un diseño y forma de interaccionar para la nueva versión de la aplicación.
4. *Diseño de la base de datos*. Incluye el estudio de los datos con los que trabajará la aplicación, el diseño del modelo entidad-relación para los datos, y la inicialización de la base de datos en el gestor de base de datos seleccionado. El tiempo estimado para la realización de esta tarea es dos semanas, con la participación del equipo completo.
5. *Implementación de las vistas*, a partir del prototipo creado previamente. Esta fase se descompone en múltiples iteraciones en el marco de un proceso evolutivo de desarrollo con participación del cliente. El proceso inicial durará un mes, luego las iteraciones de desarrollo de las vistas se realizarán a la par que se realiza el desarrollo de la funcionalidad.
6. *Desarrollo de la funcionalidad de la aplicación*, teniendo en cuenta la información obtenida del estudio de la aplicación original realizado previamente. Para esta tarea emplearemos dos meses. Durante el primer mes se implementará la funcionalidad de las distintas partes de la aplicación y las pruebas convenientes se harán sobre un servidor local. El segundo mes se dedicará a terminar de implementar la funcionalidad y a realizar pruebas sobre un servidor remoto.
7. *Fase de elaboración de la memoria*. Se procederá a escribir el documento en el que se reúne toda la información relacionada con el proyecto realizado. El tiempo estimado para esta fase es de un mes y dos semanas. En este período se realizarán revisiones semanales del progreso de la memoria.

Plan de trabajo

* Período es equivalente a una semana excepto en el caso del primer período que equivale a un día

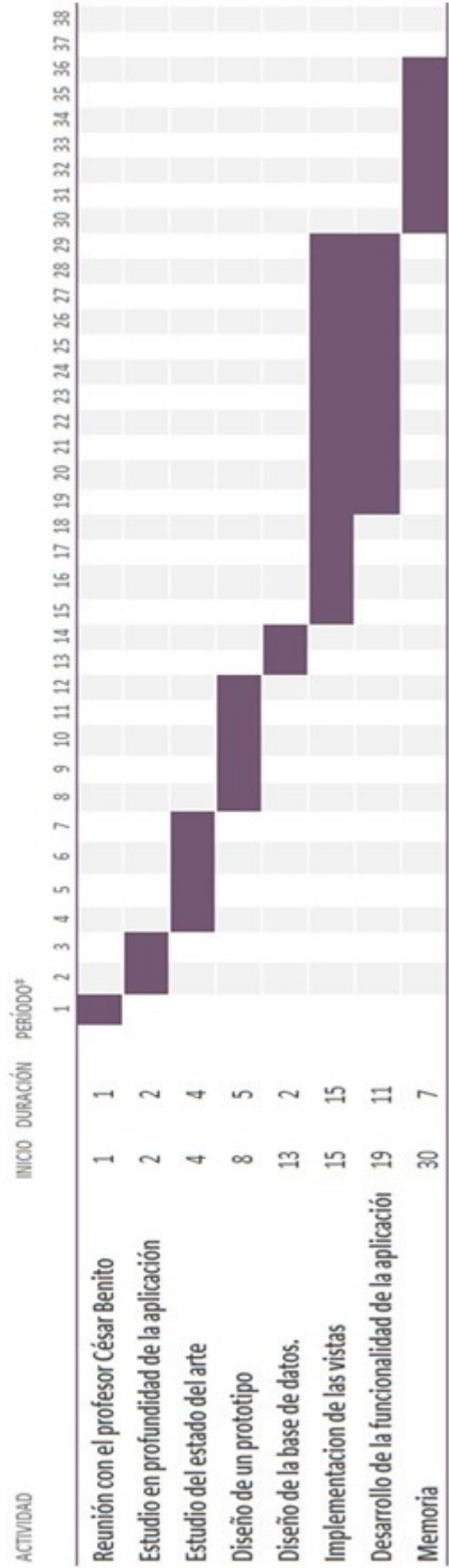


Figura 1. Plan de trabajo.

Introduction: History, Objectives and Planning

The motive for this project will be specified in the following section, as well as a first approximation to the given problem and an introduction to the initial author of the application that we are going to improve. On the other hand, there will be a description of the aims pursued in the upgrade. There will also be a detailed explanation of the planning done to organize the tasks that fulfil the present project.

History

During the fulfilment of the Computer Science Degree (from now on we will refer to it as CSD) we had the opportunity to learn about different aspects of this profession. When we had to choose the dissertation project we took into account three factors: what have we learned, which branches of Computer Science were more attractive for us and the latest technological trends. We were given the opportunity of developing an application that will later be used by the pupils of the University Complutense, specifically the students who study the subject of Genetics in the Department for Biological Sciences.

Professor César Benito from the department of genetics, developed Android mobile applications for his students. Each app covers a different topic of genetics (ex. One locus, two loci or mitosis). (Benito Jiménez & Espino Nuño, 2013) (Benito Jiménez, 1997) (Benito Jiménez, 2015).

In order to understand the problem and get to know the Apps developed by Professor César Benito, we arranged a meeting with him. In the meeting, he gave us a brief introduction to Mendel's Laws, these are "statements about the way certain characteristics are transmitted from one generation to another in an organism." (Science clarified), and about mitosis. Moreover, he showed us the Apps he had been working on and he explained to us the limitations he found when he was developing them.

These Apps are located in the user's device. The structure is very similar: a part with theoretical content, a part with practical exercises and a part with a final test. Its design is not as flexible as it could be, in such a way that it is not easy to add new material or update the types that are already in the app. In addition, all the material of the course reside in the client. This causes problems such as with the usage of high quality images. Another problem is that the chosen development method was native development in Android, this prevents users of other operative systems from using the application.

Objectives

The apps made by Professor César Benito Jiménez have structural and usability problems. The main disadvantages are: the client's device is the one in charge of storing all the data generated in the app, as a result the apps are heavier; the data (ex. Theory and exercises) is introduced in the code itself, as a result new versions need to be generated every time the contents of the app change; and a poor design from the point of view of usability.

This dissertation aims to address the existing problems in the Apps developed by César Benito Jiménez by means of the development of an app which follows the patterns established by

Software Engineering and by app development techniques. In order to solve those problems, our proposal consists in using a client-server model and a multiplatform development able to adapt automatically to the different characteristics of the client device. Moreover, we look towards making it easier for users in their different roles to use the app. Therefore, on one hand, we want to make it easier for teachers to make and edit content, as well as checking the students' evolution; on the other hand we want to facilitate students the work with materials provided and the completion of the exercises.

Another objective is the improvement of the techniques used in the old version of the app to store information. Data were introduced manually in core, which makes difficult editing and adding new material to the app. Henceforth, one objective is to design a relational database which encompasses the elements of the app, so that other characteristics can be added to enable the gathering, edition and addition of data.

Also, the project proposes redesigning the interface of the app to adapt it to the new designing guidelines used nowadays to develop apps. By these means we can ensure that users will obtain a friendly experience with the contents of the app. The new interface will be associated to a new approach of the learning experience in the app. This is intended to make it educative, but also motivating. It will contain elements that offer users an entertaining experience which encourages them to continue their learning process.

Planning

To fulfil the objectives and requirements, the following planning was made (also shown in Figura 1):

1. *In-depth study of the App.* This job accomplishes multiple objectives destined to understand the problem and the state of the actual solution. Therefore, this task is about identifying and understanding the workflows that we want to support with the tools. In addition, it is necessary to analyse the complexity of the algorithms used to execute the different tasks in the App, for example calculations made by the App and shown to the student. Finally, we aim to detect usability and design problems in the App including storage and image processing which were previously mentioned. To do this task we estimate that in total, the team needs to work during a period of two weeks.
2. *Study of the state of art* regarding mobile App development techniques. The objective is to determine which mobile platforms are the most used nowadays, so as to know in which operative systems we should focus when developing our App and also to know which tools we need. On the other hand the purpose of doing this study is to identify design trends related to user interface.
In this stage there is a selection of the technologies needed to create the app and the data persistence will also take place. In any case information will be obtained from the App revisions and real platforms as well as the existing bibliography. The time estimated to accomplish this task is one month. In this section each of the members of the team will search for articles and news related to the different aspects of the state of art that wants to be investigated.
3. *Designing an initial prototype* using rapid prototyping in order to redesign the user interface considering usability details that were absent before. After that, the result will be validated by the client. This process will last a month. Several iterations are proposed

so that the client can validate the design and the way of interacting for the new version of the application.

4. *Database design.* It includes data analysis, entity-relationship design, and the database initialization in the database manager selected. The expected time to do this task is two weeks, with the collaboration of every team members.
5. *Implementing the interface* using the previous prototypes. This stage is separated in several iterations in an evolving environment of development and with the participation of the client. The initial process will take a month, and then the interface development iterations will be done at the same time as the development of the functionality.
6. *Developing the app functionality* considering the information obtained in the previous study. During the first month, the functionality of the different parts of the application will be implemented and, also, the remote server will be tested.
7. *Memory document.* The document will be written using the information collected in the project development. The estimated time for this process is a month and two weeks. Weekly reviews will be made in this stage.

2.Estado del arte y requisitos

En la presente sección se detallarán aspectos importantes sobre la reunión llevada a cabo con el profesor. Además se explicará la funcionalidad de las aplicaciones creadas inicialmente, así como las tecnologías iniciales utilizadas para la creación de la antigua aplicación. También se abordará de forma breve el trasfondo genético que tiene la aplicación a desarrollar. Por otro lado, también se discutirán otras herramientas utilizadas en el ámbito de la educación. Estas herramientas persiguen una finalidad parecida a la del presente desarrollo, aunque a veces con temáticas diferentes. Por último se abordarán las guías y los métodos llevados a cabo para el diseño de la interfaz de la aplicación.

2.1.Descripción funcional de las aplicaciones existentes

Para conocer de primera mano las aplicaciones desarrolladas por el profesor César Benito Jiménez, concretamos una reunión con él.

La reunión duró 1 hora y 11 minutos y fue grabada con el consentimiento del profesor. La grabación se incluye en el material de la memoria. Preparamos una serie de preguntas para la entrevista, las cuales fueron:

- ¿De qué modo se incluyen las preguntas para los ejercicios y test en su aplicación?
- ¿Se tiene un seguimiento de las estadísticas de los ejercicios de los alumnos? ¿Por qué sí? ¿Por qué no? En caso afirmativo, ¿se tienen en consideración los aciertos y fallos o solo el número de respuestas contestadas?
- ¿Los usuarios de la aplicación se registran para acceder a ella?
- A la hora de plantear un ejercicio, ¿se permite deshacer? ¿Volver atrás? ¿Hay un número máximo de fallos permitidos? ¿Se puede pasar al siguiente ejercicio sin haber respondido correctamente el actual?
- ¿Ha recibido feedback de su aplicación por parte de sus alumnos? ¿Por qué sí? ¿Por qué no? En caso afirmativo, ¿cuáles eran los pros y los contras que encontraron?

En la reunión también se abordó el actual diseño de las aplicaciones. La estructura de todas ellas es muy similar desde el punto de vista del usuario. Toda la funcionalidad se accede a través de un menú que da acceso a las diferentes opciones.

En el menú principal tienen apartados de: teoría, ejercicios, test final y bibliografía. Ejercicios y test proponen actividades que los alumnos han de resolver para trabajar los contenidos del tema. En ambos casos se parte de un enunciado del problema, pero solo los test ofrecen funcionalidad para comprobar las respuestas, mientras que los ejercicios se limitan al enunciado.

En el caso de los test sobre las distintas formas de segregación de los genes, la funcionalidad para introducir y comprobar las respuestas se introduce mediante *calculadoras*.

Una *calculadora* es un componente de interfaz con elementos para introducir los diferentes valores que forman las respuestas de un problema. Los valores se recogen con cajas de edición. Una vez introducidos los valores, la calculadora aplica una serie de algoritmos y fórmulas para comprobar si son correctos de acuerdo con la segregación planteada. En caso de que no lo fueran,

habría que introducir nuevos valores o ir a la pantalla principal de la aplicación y elegir otro tipo de segregación.

Los test sobre mitosis son preguntas de selección de opciones. Se ilustran con imágenes que representan diferentes fases de la vida de la célula. Los estudiantes tienen que identificar la fase que se ve en la imagen.

Las conclusiones de la reunión fueron:

- Las aplicaciones desarrolladas hasta el momento están alojadas totalmente en local (en los dispositivos de los alumnos).
- Las aplicaciones son muy pesadas (ocupan mucho espacio de almacenamiento) debido a la alta calidad de las imágenes que incluyen en diferentes apartados.
- Las aplicaciones están mal diseñadas desde el punto de vista de la usabilidad. El feedback de los usuarios no es positivo en este aspecto.
- Se necesita una interfaz de administrador para el seguimiento de los alumnos.
- Las aplicaciones son de desarrollo nativo en Android. Los alumnos que utilizan otros sistemas operativos, como iOS o Windows Phone, no pueden utilizarlas.

A partir de dichas conclusiones, se elaboró un estudio más exhaustivo del estado de las aplicaciones y del trabajo relacionado. Esa fue la base para la elaboración de la lista de requisitos.

2.2. Tecnologías iniciales

Hoy en día existen varias plataformas móviles sobre las que desarrollar aplicaciones móviles. Entre ellas las más extendidas son iOS (Apple) y Android (Google), aunque también existen otras plataformas minoritarias como Windows Phone y Blackberry. Para abordar la variabilidad entre plataformas hay dos formas de desarrollar aplicaciones móviles:

- *Desarrollo nativo.* Consiste en desarrollar una aplicación para una plataforma en concreto utilizando su lenguaje nativo. Las aplicaciones desarrolladas de forma nativa tendrán acceso a características avanzadas del terminal, como por ejemplo la cámara, el *Global Positioning System* (GPS), lista de contactos, etc. Este tipo de desarrollo es más complejo, por tanto implica un coste mayor debido al tiempo que se tiene que emplear. Además, su código no es reutilizable para otras plataformas.
- *Desarrollo multiplataforma o híbrido.* Consiste en desarrollar una aplicación combinando tecnologías de las aplicaciones nativas (conjunto de API (*Application Programming Interface*) nativas de cada plataforma) y de las aplicaciones web (ej. HTML5 (*HyperText Markup Language*), CSS3 (*Cascading Style Sheets*) y JavaScript). Este tipo de desarrollo permite reutilizar gran parte del código para todas las plataformas. Además, el coste de su elaboración es menor que el de las aplicaciones desarrolladas de forma nativa. No obstante, ofrecen peor experiencia al usuario que las aplicaciones nativas.

Las aplicaciones desarrolladas por el profesor son nativas para Android, como se indicó anteriormente. La herramienta elegida para su desarrollo se llama “App Inventor 2”. Se puede acceder a ella a través del link <http://ai2.appinventor.mit.edu/> (Inventor, 2012) e iniciando sesión con una cuenta de correo. Es una plataforma que acerca el desarrollo de aplicaciones móviles nativas para Android a todos los públicos. Proporciona asistentes para diseñar tanto las pantallas de la aplicación (las vistas) como para asociarlas funcionalidad.

Para el diseño de las vistas de la aplicación la plataforma cuenta con una “Paleta” desde la cual se pueden añadir elementos. Estos elementos se clasifican en los siguientes apartados: interfaz de usuario, disposición, medios, dibujo y animación, sensores, social, almacenamiento, conectividad, LEGO® MINDSTORMS®, y experimental. Cuando elegimos añadir a nuestra aplicación alguno de los elementos contenidos dentro de estos apartados, podemos observar en el área “Componentes” del entorno la estructura visual que tiene la ventana sobre la que estamos trabajando. Además tiene un apartado “Propiedades”, donde se pueden modificar las características del elemento seleccionado en la vista. Por último, el apartado “Medios” muestra una lista de imágenes, videos o cualquier otro tipo de medio que contiene nuestra aplicación.

Para añadir funcionalidad a la aplicación, la plataforma cuenta con otro modo de trabajo al que denomina “Bloques”. Los bloques están representados con colores, según el papel que toman en la funcionalidad de la aplicación. App Inventor 2 nos proporciona bloques de control, lógica, matemáticas, texto, listas, colores, variables y procedimientos. Los bloques se parecen a las piezas de un puzle. Con ellos podemos elegir el comportamiento que queremos que tengan los elementos de nuestra aplicación, por ejemplo, si presionamos un botón y queremos que nos lleve a la siguiente vista. También podemos almacenar, temporalmente, información en variables. Para encadenar los bloques basta con arrastrarlos a la posición deseada, de esta forma podemos crear funcionalidad más sofisticada.

La plataforma permite emular la aplicación mediante un emulador o utilizando un dispositivo móvil conectado al ordenador por el puerto USB. También permite generar la aplicación guardando un archivo .apk en el ordenador o generando un código QR desde el que acceder a la dirección web en la que se podrá descargar el archivo .apk.

2.3.Trasfondo genético

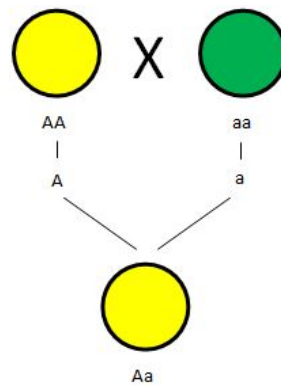
Las aplicaciones tienen como uno de sus objetivos introducir y asentar las bases de las leyes de Mendel. Las leyes de Mendel establecen los principios básicos de la genética moderna. La genética se define como “el área de estudio de la biología que busca comprender y explicar cómo se transmite la herencia biológica de generación en generación.” (Wikipedia, 2016).

Mendel fue un monje agustino católico que realizó experimentos que consistían en el cruce de distintas variedades de guisantes. Como resultado encontró que existen caracteres dominantes y recesivos.

Los *caracteres dominantes* son aquellos rasgos genéticos que se muestran visiblemente en el individuo. Por el contrario, un *gen recesivo* no es visible en los rasgos físicos del individuo. Es decir, los caracteres dominantes están presentes en el fenotipo (apariencia física) y en el genotipo (carga genética) del individuo, mientras que los recesivos se hallan en su genotipo pero no en su fenotipo.

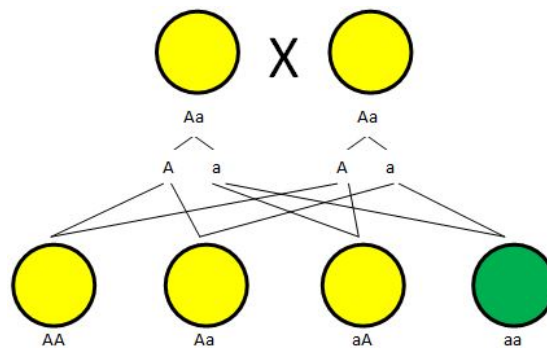
A raíz de estos experimentos, Mendel propuso tres leyes que explican y predicen cómo van a ser los fenotipos de un nuevo individuo (Mendel J. G., 2016). Éstas son:

1ª Ley de Mendel o Principio de la Uniformidad: Al cruzar dos individuos puros (AA y aa) para un determinado carácter, la primera generación descendiente tendrá la misma apariencia externa en ese carácter que uno de los progenitores.



En sus experimentos, Mendel realizó el cruce entre guisantes amarillos y guisantes verdes (ambos puros) y observó que todos los guisantes que nacían eran amarillos. Por esta razón, llegó a la conclusión de que el gen “color amarillo” predomina sobre el gen “color verde”.

2ª Ley de Mendel o Principio de la Segregación: Al cruzar dos individuos híbridos de primera generación, se da lugar a una segunda generación en la que 3/4 de los individuos obtenidos presentara un fenotipo dominante y 1/4 presentan un fenotipo recesivo (proporción 3:1).



La segregación de genes consiste en “la separación de cromosomas homólogos en gametos diferentes (materno y paterno)” (Wikipedia, 2016).

3ª Ley de Mendel o Ley de la Asociación Independiente: Existen rasgos genéticos que se heredan independientemente unos de otros y, por lo tanto, no afectan en el patrón de herencia de los otros.

Los genes involucrados en la herencia de un determinado carácter pueden aparecer en uno o varios lugares separados del cromosoma de la especie. Se conoce por *locus* a la ubicación que tiene un determinado gen en un cromosoma. Si hablamos en plural, nos referimos a *loci*.

Además de esto, otro tema que se quiere reforzar con las aplicaciones es el conocimiento de la *mitosis* y sus fases. La *Mitosis* es la división de una célula madre en dos células hijas genéticamente idénticas. Se divide el núcleo de la célula y el citoplasma (la parte de la célula que rodea al núcleo). La mitosis se realiza en varias fases: *profase*, *metafase*, *anafase* y *telofase*. Es posible hacer una división en fases más exhaustiva, pero estas son las principales.

Las aplicaciones planteadas por los profesores, así como la que se desarrolla en este TFG, engloban todos estos conocimientos sobre genética para reforzar el aprendizaje del usuario. Contienen materiales tanto sobre segregación de genes como sobre mitosis.

2.4.Trabajo relacionado

En este capítulo se explican las distintas tendencias que existen a la hora de utilizar la tecnología en la labor de enseñanza, así como las tendencias y heurísticas para el diseño de interfaces.

E-learning

El uso de la tecnología ha cambiado radicalmente nuestras vidas. En el ámbito educacional ha supuesto un gran cambio también. Si nos centramos en la universidad, podemos ver que las clases tradicionales, con un profesor explicando y los estudiantes tomando apuntes, son cada vez menos frecuentes. Como dice la doctora Pilar Sancho Thomas (Sancho Thomas, 2010): “a día de hoy, el uso de herramientas de aprendizaje a través de la web (denominadas genéricamente como e-learning) se ha convertido en algo habitual en cualquier universidad, bien como complemento de las clases presenciales (escenario que se denomina aprendizaje combinado o mixto, *blended learning*, en inglés), bien como parte de la oferta de educación a distancia.”.

En este ámbito, las herramientas más utilizadas son las conocidas como *Learning Management Systems* (LMS). Los LMS son herramientas de soporte y apoyo para las clases. Ofrecen un conjunto de funcionalidades orientadas al aprendizaje colaborativo, y que además permiten al profesor realizar un seguimiento de la evolución de sus alumnos. Algunas de las funcionalidades incluidas habitualmente son foros, chats y wikis.

Además de los LMS, existen los *Learning Content Management Systems* (LCMS), que serían una extensión de los anteriores. Los LCMS se caracterizan por tener la misma funcionalidad que los LMS incluyendo, también, la posibilidad de crear y administrar contenidos (Sancho Thomas, 2010). Un ejemplo de estas herramientas son las plataformas Moodle (Moodle, 2016), Sakai (Sakai, 2014) y aLF (UNED, 2016).

Moodle y Sakai son LCMS de código abierto que permiten crear entornos de aprendizaje colaborativo mediante foros, wikis y chats. En ellos el profesor puede publicar apuntes, enlaces para entrega de prácticas, exámenes online, etc. La UCM utiliza estas plataformas en su campus virtual como apoyo a las clases presenciales, siendo Moodle el elegido principalmente. Otras universidades que también utilizan Moodle en sus campus virtuales son la Universidad Autónoma de Madrid (UAM) y la Universidad Carlos III de Madrid (UC3M).

La Universidad Nacional de Educación a Distancia (UNED) utiliza la plataforma aLF en su labor de enseñanza no presencial. aLF es una plataforma LCMS desarrollada por la propia UNED para cubrir las necesidades específicas de la propia universidad. Tal y como la define la propia UNED, es una herramienta destinada a que “tanto el equipo docente como el alumnado, encuentren la manera de compaginar tanto el trabajo individual como el aprendizaje cooperativo” (UNED, 2016). El hecho de que sea un desarrollo propio hace que ofrezca una funcionalidad adaptada a las necesidades de la UNED, en particular con un desarrollado soporte a los ejercicios y prácticas no presenciales.

Los *Massive Open Online Course* (MOOC) también son herramientas para el e-learning. Los MOOC son cursos abiertos, gratuitos, que se imparten online y sin límite de participantes. Se basan en materiales y ejercicios en línea. Los ejercicios son auto-corregidos por el sistema en la mayor parte de los casos, para permitir dar soporte a un número muy elevado de alumnos (por encima de cientos y frecuentemente de miles). Los participantes pueden interactuar unos con otros a través de los foros. En muchos casos ofrecen un certificado de aprovechamiento y, en ocasiones,

la entidad organizadora puede ofrecer una prueba final para acreditar los conocimientos adquiridos y conseguir un certificado de mayor nivel. La UCM, la UC3M y la UAM tienen programas con MOOC en un amplio espectro de especialidades como informática, medicina, química, filosofía, etc.

Basadas en los MOOC, existen plataformas interactivas orientadas al aprendizaje. Estas permiten realizar diferentes MOOC, en ocasiones constituyendo itinerarios o especialidades, y obteniendo los certificados correspondientes. Un ejemplo es Codecademy (Codecademy, 2016), que ofrece una gran variedad de cursos para aprender lenguajes de programación como Python, JavaScript o PHP, así como el uso de diferentes API y Git. Todo de manera gratuita. Udacity es otro ejemplo de estas plataformas (Udacity, 2011). Udacity imparte una gran cantidad de cursos de informática, cada uno con varias unidades con videos explicativos y test para asentar lo aprendido.

La gran diferencia que presentan los MOOC con los LMS y los LCMS es que los LCMS y LMS son herramientas de apoyo al aprendizaje presencial o semipresencial. Pueden ser personalizados según las necesidades de los usuarios. Por el contrario, los MOOC son cursos con materiales cerrados en los que un usuario participa, pero que no puede modificar, y no se adaptan más que de forma (semi-)automática a sus necesidades.

Como hemos visto, las universidades están muy implicadas en fusionar la educación tradicional con las plataformas de e-learning. Las plataformas anteriores están orientadas principalmente a la web, pero también hay una gran selección de herramientas móviles orientadas a la enseñanza.

En el Top aplicaciones de Google Play (Google_Play, 2016), podemos encontrar la categoría “Educación”. Un repaso a las apps más populares en el “Top aplicaciones de Educación” proporciona información acerca de qué tienen las aplicaciones de éxito de este tipo. Aquí aparece con fuerza el concepto de “gamificación”.

En los últimos años, se ha incrementado el desarrollo de aplicaciones móviles, y, con ellas, han surgido tendencias para captar la atención de los usuarios. Una de estas tendencias es la gamificación. La gamificación es una estrategia que consiste en dar características de juego a las distintas actividades de una aplicación no lúdica para involucrar y motivar al usuario. Debido a esto, la gamificación está presente también en las aplicaciones móviles de aprendizaje. Un ejemplo de esto sería que las lecciones tengan distintos niveles de complejidad, y que se vayan desbloqueando a medida que se progresa en la lección. También, se puede premiar al usuario cuando logra un objetivo determinado (ej. monedas, medallas, etc.). De esta manera, los conceptos se asimilan mejor ya que el usuario se divierte a la vez que aprende.

Un ejemplo popular de aplicación de la gamificación en el contexto educativo es la app Duolingo, la más popular en esta categoría en junio de 2016. Se trata de una aplicación para el aprendizaje de idiomas. Cuenta con una versión móvil y otra para ordenador. La gamificación aparece a través de la división de cada tema en distintas lecciones, que hay que ir desbloqueando. Cada vez que se acaba un conjunto de temas, se sube de nivel. Además se pueden ganar “lingots” (la moneda de este juego), gracias a los logros que el usuario supera. Con esos lingots, el usuario podrá comprar en la tienda de la aplicación.

Otra app educativa que utiliza la gamificación es ClassDojo. ClassDojo es una herramienta para ayudar a los profesores a motivar a sus alumnos y mejorar el aprendizaje. Cada estudiante tiene un avatar, que gana puntos por buen comportamiento, compañerismo, atención en clase, tareas

bien realizadas, etc. Esto refuerza los comportamientos positivos de los estudiantes. Además, los padres también pueden tener un seguimiento de sus hijos. También es una red social privada, ya que se pueden subir fotos de los alumnos en clase y ver los avatares y logros de otros estudiantes, pero solo dentro de tu propia clase.

En el Top de aplicaciones educativas de Google Play (Google_Play, 2016), Duolingo es la más descargada y viene seguida de varias aplicaciones que también utilizan la gamificación.

En el cuarto puesto de esta categoría tenemos otra aplicación gamificada llamada *“Aprende inglés con ABA English”*. Esta aplicación permite al usuario aprender inglés con películas. Ofrece seis niveles de dificultad y un profesor particular para cubrir todas las habilidades del idioma.

En el séptimo puesto, podemos encontrar la aplicación *“Kahoot!”*, también gamificada. Ésta ayuda a reforzar los conocimientos de los estudiantes con una competición en tiempo real de preguntas y respuestas entre varios estudiantes.

Como podemos ver, la gamificación es una estrategia muy utilizada en este tipo de aplicaciones, aunque su uso no es universal. Existen también aplicaciones populares que no la emplean.

En el tercer puesto de esta categoría está *“Todo Test: Test de conducir”*, una app que no usa gamificación. Esta aplicación está orientada a las personas que quieran conseguir distintos permisos de conducir y ofrece al usuario test como los de las autoescuelas para que pueda practicar de cara al examen. No ofrece ningún tipo de reto ni tampoco una recompensa por hacerlo bien.

En el sexto puesto se encuentra *“Photomath - Cámara calculadora”*. Esta aplicación tampoco usa gamificación. Sirve para ayudar a resolver problemas matemáticos. Sacándole una foto a una ecuación muestra su resultado y los pasos detallados para hallarla.

En cuanto a las aplicaciones específicas para genética, la gran mayoría son aplicaciones que tienen solo información teórica sobre el tema. No ofrecen al usuario la posibilidad de reforzar sus conocimientos con algún tipo de ejercicio o de comprobar lo aprendido mediante un examen. Estas aplicaciones podrían asemejarse a libros electrónicos con un contenido muy específico. Algunos ejemplos son: *“Genetica”* (Anastore, 2015), *“Mendelian Genetics”* (Limited W. K., Mendelian Genetics, 2015), *“Molecular Genetics”* (Limited W. K., Molecular Genetics, 2015), *“Genetics 4 Medics”* (Limited A. , 2015).

Por tanto, en el campo de la genética, las aplicaciones desarrolladas en la Facultad de Biología de la UCM suponen una mejora sobre las asistentes. Un mejor desarrollo de las mismas podría suponer su popularización como herramientas de aprendizaje más allá de algunos grupos específicos.

En definitiva, la educación académica hoy en día se apoya cada vez más en herramientas como LMS o LCMS para poder ofrecer mayor cantidad de recursos y facilitar la comunicación entre alumnos y entre alumnos y profesores. Por otro lado, utilizando estas tecnologías es posible para muchos alumnos poder profundizar en temas de su interés sin necesidad de acudir a cursos presenciales gracias herramientas como los MOOC. Asimismo, también se utilizan aplicaciones móviles donde el usuario aprende de forma individual sobre algún tema de su interés. En las aplicaciones móviles educativas la tendencia a la gamificación va ganando terreno. Con ello, se busca que el usuario aprende a la vez que se motiva jugando.

Diseño de interfaces

Uno de los aspectos más importantes que debe de cuidar una aplicación para atraer y mantener a sus usuarios es un diseño visual en el que prime la usabilidad. Las aplicaciones interactúan con el usuario a través de la interfaz de la aplicación. Por ello es muy importante cuidar todos los aspectos que faciliten la interacción del usuario con la aplicación para que éste tenga una experiencia de uso satisfactoria.

Para lograr dicho resultado, existen guías de diseño. Las hay generales y específicas para plataformas. Entre las primeras, este proyecto considera las 10 heurísticas de usabilidad de Jakob Nielsen (Nielsen, 1995) , que se deben aplicar a cualquier aplicación. Entre las segundas están, por ejemplo, las guías de diseño para Android (Android, 2016).

Las heurísticas de usabilidad de Jakob Nielsen son normas básicas de usabilidad que toda interfaz de aplicación debe cumplir. Son las siguientes:

1. *Visibilidad del estado del sistema.* Toda aplicación debe informar en todo momento al usuario sobre en qué punto de la aplicación se encuentra. Lo debe hacer de forma que le sea sencillo poder retornar a cualquier otro punto de la aplicación.
2. *Relación entre el sistema y el mundo real.* Toda aplicación debe incluir vocabulario familiar para el usuario. Además, la información mostrada debe aparecer en un orden lógico y natural.
3. *Control y libertad de usuario.* El usuario siempre tiene que tener el control de poder volver a un estado anterior de la aplicación tras haber cometido un error. Además, también tiene que tener la posibilidad de rehacer una acción.
4. *Consistencias y estándares.* Los usuarios de la aplicación deben saber en todo momento qué significan las acciones, símbolos o palabras. Para ello hay que usar siempre las convenciones establecidas
5. *Prevención de errores.* Crear un diseño que sea cuidadoso y ayudar al usuario a que no cometa errores, por ejemplo a la hora de insertar información en un campo de texto.
6. *Reconocer antes que recordar.* Hacer visibles acciones e información de la aplicación para que el usuario no tenga que recordar cómo hacer cierta acción.
7. *Flexibilidad y eficiencia de uso.* Para que la aplicación sea utilizable tanto por usuarios avanzados como por usuarios expertos.
8. *Diseño estético y minimalista.* Evitar cualquier tipo de información innecesaria que sobre cargue la interfaz.
9. *Ayudar a los usuarios a reconocer.* Mostrar mensajes de error que vengan escritos en un lenguaje sencillo con instrucciones y soluciones al problema claros.
10. *Ayuda y documentación.* La aplicación deberá tener una lista fácil de encontrar con información sobre cómo llevar a cabo los pasos que componen la aplicación. El manual de usuario solo es imprescindible en aplicaciones complejas.

Dentro de las guías de desarrollo específicas tenemos *Material Design*. Según Android Developers, Material Design es “una guía integral para el diseño visual, de movimientos y de interacción en distintas plataformas y dispositivos” (Android_Developers, 2016). Android utiliza esta guía de diseño a partir de la versión Android 5.0. Material Design se caracteriza por la simplicidad de las vistas, nada recargadas, utilizando formas clásicas y sencillas (ej. círculos, rectángulos). Utiliza la luz y la profundidad para dar sensación de movimiento y atraer la atención del usuario.

Un aprendizaje llevadero y didáctico para el usuario no será posible sin una interfaz que le resulte amigable e intuitiva. Ello requiere tener en cuenta las consideraciones de diseño de interfaces existentes.

2.5. Requisitos del proyecto

Tras el análisis realizado en las secciones anteriores se plantean los siguientes requisitos para la aplicación a desarrollar en este trabajo:

1. Desarrollar una aplicación que englobe todos los temas de genética. Ello permitirá integrar y complementar los diferentes aspectos del curso y sus contenidos. En la actualidad estos aspectos están dispersos en varias aplicaciones.
2. La aplicación será un desarrollo multiplataforma. Así, se solventarán los problemas que tenían las antiguas aplicaciones, que sólo se podían utilizar en móviles Android.
3. La aplicación seguirá una arquitectura cliente-servidor. Con ello se logra una aplicación cliente más ligera, y se pueden gestionar de forma centralizada en el servidor datos de múltiples usuarios.
4. El usuario se registrará para acceder a la aplicación. De esta manera el profesor podrá llevar un seguimiento de sus alumnos.
5. Los temas tendrán un apartado de teoría, un apartado con los ejercicios propuestos, y un test final. Algún apartado podría no estar presente si no fuera necesario en un tema.
6. Los ejercicios son preguntas con enunciado y, opcionalmente, con imágenes. Los ejercicios se podrán realizar un número ilimitado de veces.
7. Los ejercicios están agrupados por niveles. Cada tema podrá tener uno o varios niveles de ejercicios.
8. Un nivel se desbloqueará cuando se haya completado el nivel anterior. De esta forma se aplica un aspecto de la gamificación para atraer la atención del usuario y motivarlo (ver sección 2.4).
9. Los enunciados de los ejercicios y las distintas calculadoras para resolverlos estarán en la misma pantalla. Una calculadora es un conjunto de cajas de edición que, introduciendo unos valores dados por el enunciado del problema correspondiente, nos indicará si la segregación es correcta o no lo es.
10. El test final es un conjunto de preguntas con un enunciado, con imágenes (opcional) y un conjunto de respuestas dadas, donde solo una será la correcta.
11. El test final de cada tema solo se podrá realizar una vez. Al permitir solamente hacer el test una vez, se obtiene información de si los alumnos han entendido las explicaciones de teoría y los ejercicios propuestos.
12. Al finalizar un tema, es decir, tras superar su test final, la aplicación presentará un resumen de las respuestas correctas, incorrectas y la puntuación. La puntuación es una media entre las preguntas correctas e incorrectas sobre 10. Asimismo, se podrá ver de nuevo el test con las respuestas correctas e incorrectas señaladas con carácter informativo.
13. El diseño de la aplicación tendrá en cuenta guías de diseño para que la aplicación sea lo más cómoda y accesible posible. En este caso se considerarán las 10 heurísticas de usabilidad de Nielsen. Además, se tendrán muy en cuenta las pautas de diseño de Material Design (ver sección “Diseño de interfaces”).

3.Arquitectura de la aplicación

En esta sección, se detallan los aspectos técnicos de la aplicación desarrollada en este proyecto y llamada Biogam. Inicialmente se explica la arquitectura de la aplicación indicando brevemente cuáles son los componentes de ésta, cliente y servidor. Después se explica el modelo de datos implementado para la aplicación, detallando también cada uno de los elementos que lo componen. También se realiza una explicación de los componentes software utilizados en la aplicación. Por último se describe detalladamente el flujo de la aplicación teniendo en cuenta la arquitectura cliente-servidor.

3.1.Arquitectura cliente-servidor

El proyecto adopta una arquitectura cliente-servidor a fin de facilitar la distribución de contenidos y reducir los requisitos computacionales de la aplicación en los dispositivos de los usuarios. Las principales características de las dos partes de la aplicación se resumen a continuación.

Cliente

El cliente es la aplicación híbrida e instalada en los distintos dispositivos móviles de los usuarios.

Para el desarrollo de la parte cliente se ha utilizado el patrón de diseño Modelo-Vista-Controlador (MVC). Este patrón permite separar la lógica de negocio de la interfaz de usuario. Esto favorece el bajo acoplamiento y alta cohesión del software, e incrementa las posibilidades de paralelizar el trabajo de desarrollo (Wikipedia, 2016).

Para la representación de la información y las acciones que se pueden realizar, la parte cliente cuenta con una interfaz gráfica con respuesta táctil de usuario. Está organizada por vistas y bloques:

- Bloque de *autenticación*. La autenticación cuenta con dos vistas principales, la vista de login usada por los usuarios dados de alta en el servidor remoto y una vista de registro para darse de alta.
- Bloque *Home*. Bloque principal en el que se puede elegir los distintos temas.
- Bloque *Categorías*. Cada tema es dividido en sub-bloques a los que hemos denominado categorías. Estas categorías dependen de las actividades que se pueden realizar dentro del tema. Por ejemplo, algunos temas se pueden dividir en sub-bloques ejercicios, teoría y test, mientras que otros solo en teoría. Cada sub-bloque cuenta con sus respectivas vistas.

Para tratar las acciones que realiza el usuario sobre la interfaz se han introducido controladores, encargados de la lógica que hay detrás de cada acción como pulsar un botón, introducir datos, responder ante gestos, etc. Se ha creado un controlador para implementarla lógica de cada vista antes definida.

En el cliente se ha creado una capa de servicios, en esta capa introducimos las reglas del negocio y definimos las funcionalidades que manipulan directamente los datos.

Los servicios y los controladores se definen con mayor detalle en el apartado “Para la aplicación móvil”

La aplicación cliente almacena datos en la memoria del dispositivo donde está desplegado, tales como información del usuario e información del estado y progreso de las actividades realizadas. Ésta funcionalidad entra en acción cuando el usuario se registra o realiza los test o realiza los ejercicios. Toda esta recopilación de datos va almacenada en una base de datos local creada al ejecutar la aplicación por primera vez.

El cliente necesita de conexión a Internet. Al arrancar, la primera acción que se hace al ejecutar es contrastar los datos de usuario almacenados en local con los datos de los usuarios alojados en el servidor remoto. También, periódicamente parte de la información almacenada en la base de datos local sobre el desempeño del usuario es trasladada al servidor remoto. En concreto, se copia la información de los test y los ejercicios resueltos.

Los frameworks usados para el desarrollo de este proyecto están orientados a crear aplicaciones que siguen este patrón. En particular, Ionic framework genera aplicaciones híbridas usando:

- AngularJS. Se trata de un framework que facilita el desarrollo de aplicaciones web siguiendo el patrón MVC. Con él se ha desarrollado todo el código fuente de la parte cliente.
- Apache Cordova. Encargado de interpretar, compilar y generar la aplicación nativa a partir de tecnologías web.

Los lenguajes en este framework son JavaScript, HTML y CSS. JavaScript se usa para implementar la funcionalidad. HTML y CSS se utilizan para el diseño de la interfaz. Las CSS son las “Hojas de Estilos en Cascada”. Se usan en desarrollo web para dar estilo y preparar la presentación de las vistas.

Servidor

Llamamos servidor al código desplegado en el host (aquí <http://ingenias.fdi.ucm.es/appbio>). Se encarga de interpretar y enviar las respuestas a las solicitudes hechas por los clientes.

Al igual que en la parte cliente el patrón de diseño utilizado para el servidor ha sido MVC.

La interfaz de usuario para el servidor cuenta con dos vistas: el login para la identificación del administrador de la aplicación, y la vista principal que muestra la información relevante de cada usuario como su información de cuenta y los datos recabados por cada aplicación cliente asociada.

La definición de controladores para el cliente es la misma para el servidor. Dentro de los controladores desarrollados para esta parte podemos distinguir dos grupos. En primer lugar están aquellos que serán usados para la representación de los datos de las dos vistas antes mencionadas: autenticación del administrador; representación de usuarios; tratamiento de datos para generar estadísticas. Por otra parte están los que serán usados para crear la respuesta a las peticiones de la parte cliente: autenticación y registro de los clientes; escritura y actualización de los datos enviados por los dispositivos clientes.

El servidor remoto cuenta con una base de datos SQL relacional. Su esquema es el mismo que el de la base de datos de la aplicación cliente. Usan diferentes sistemas gestores de bases de datos: la local usa SQLite y la remota MySQL.

El lenguaje de programación utilizado para el desarrollo del servidor ha sido PHP. Se trata de un lenguaje de código abierto ampliamente extendido para la creación de aplicaciones web del lado del servidor, y que facilita el acoplamiento con bases de datos MySQL.

Como herramienta para la creación del servidor, se usó Codeigniter Web Framework. Al igual que los frameworks para el desarrollo de la parte cliente, se centra en el uso del patrón MVC para crear aplicaciones web usando PHP. También brinda un conjunto de librerías que ayudan a agilizar el proceso. Codeigniter genera un proyecto totalmente organizado y configurado para aplicar el MVC.

Comunicación cliente-servidor

La aplicación móvil se comunica con el servidor en distintos casos:

- Envío de los datos del registro de un nuevo usuario.
- Comprobación de credenciales en el auto-login que realiza la aplicación.
- Envío de los resultados del uso de la aplicación.

Para la comunicación cliente-servidor se ha utilizado tecnología AJAX. La tecnología AJAX, acrónimo de “*Asynchronous JavaScript And XML*” es una mezcla de varias tecnologías. Utiliza JavaScript para hacer una petición asíncrona al servidor. Éste a su vez obtiene los datos y los devuelve en un lenguaje de intercambio de datos “universal”, antes XML y ahora cada vez más JSON.

3.2.Modelo de datos

Al tener una estructura cliente-servidor, la aplicación Biogam cuenta con dos bases de datos relacionales, una alojada en el servidor y otra en el cliente. Por comodidad en la transferencia de datos entre una y otra, son idénticas en estructura. Sin embargo, la base de datos del servidor almacena datos de todos los usuarios y la del cliente solo tiene los datos de su usuario.

La base de datos remota usa MySQL y por tanto soporta SQL (*Structured Query Language*) completo, el lenguaje de referencia para el trabajo con bases de datos relacionales. La base de datos local está desarrollada en SQLite, una versión más ligera del SQL clásico que resulta muy útil para dispositivos móviles.

En la Figura 2 se muestra el diagrama entidad-relación para la base de datos remota y local.

Diagrama entidad-relación

Para el almacenamiento de los datos de la aplicación Biogam se ha decidido emplear una base de datos relacional que contenga toda la información sobre los usuarios, la teoría, los ejercicios y los test. Esta información se detalla a continuación.

Un *tema* es un conjunto de conocimientos que se agrupan porque tienen un asunto común. Se compone por un identificador único y un nombre. En cada *tema* puede haber varios capítulos de *teoría*, *ejercicios* y preguntas de *test* (que conformarán un test global). También, puede ocurrir que no haya una de las anteriores categorías.

Un capítulo de *teoría* es un conjunto de explicaciones sobre una cuestión determinada. Se compone de un identificador único, un título y un cuerpo. Además, puede tener una, varias o ninguna *imagen*. Un capítulo de *teoría* pertenece a un *tema* en concreto.

Un *ejercicio* consiste en una prueba en la que el usuario podrá practicar sus habilidades sobre un tema en concreto. Se compone de un identificador único, un enunciado y puede tener una, varias o ninguna *imagen*. Cada ejercicio pertenece a un *nivel* y a un *tema* en concreto.

Un *nivel* representa el grado de dificultad. Se identifica por un identificador único y tiene un número de nivel. Un nivel puede tener varios *ejercicios*.

Una pregunta de *test* es una prueba en la que se evalúan los conocimientos adquiridos sobre una cuestión determinada. Se compone de un identificador único, el número de pregunta y un enunciado. Además, puede tener una, varias o ninguna *imagen*. Una pregunta de *test* pertenece a un *tema* en concreto. Cada pregunta además tiene un conjunto de *opciones* asociadas, donde solo una será la correcta.

Una *opción* de test representa una alternativa que el usuario puede escoger como respuesta a una pregunta de *test*. Tiene un identificador único y el texto con la opción en sí. Una *opción* puede estar presente en más de una pregunta de *test*, pero puede que en una pregunta sea la respuesta correcta y en otra no.

Una *imagen* representa un ejemplo visual para clarificar algo. Se identifica por un id y, además, se compone por una ruta a una imagen. Una *imagen* puede pertenecer a un capítulo de *teoría*, un *ejercicio*, un *test*, o a varios de los anteriores.

Un *usuario* es la persona que utiliza la aplicación. Se identifica por un nick, que será un correo electrónico, y tiene un nombre, una contraseña de acceso y, opcionalmente, una foto.

Un *usuario* puede resolver uno o varios *ejercicios*, las veces que quiera. Se guardará las veces que se realiza el ejercicio y las veces que se falla. Un *usuario* solo puede resolver una pregunta de *test* una vez y se tendrá en cuenta si se acierta o no al responder.

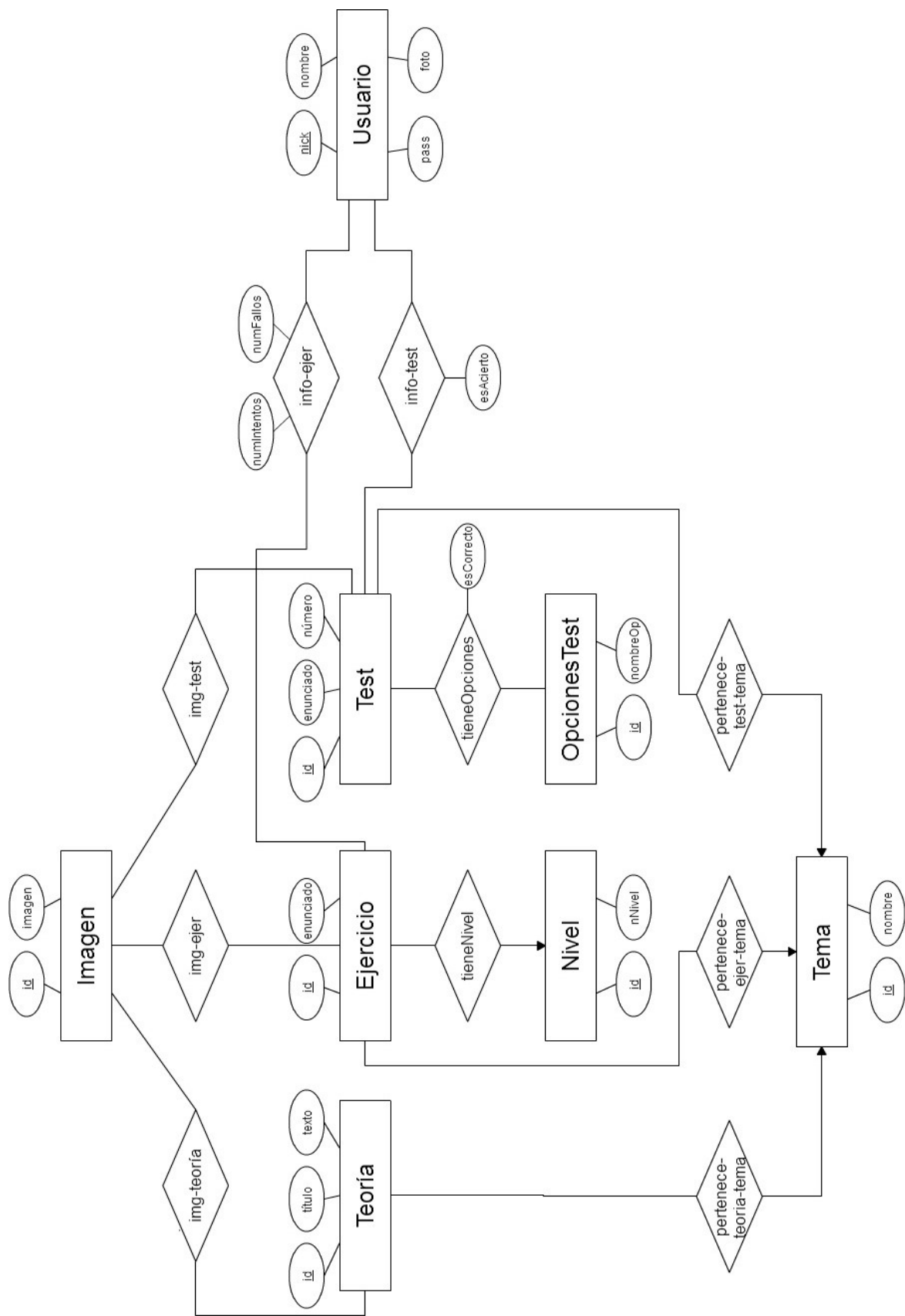


Figura 2. Diagrama entidad-relación de la base de datos

3.3.Componentes software

En esta sección se describen los distintos módulos que se han definido en la aplicación Biogam. Cada uno realiza un proceso específico acorde con los componentes necesarios para la construcción del patrón de diseño MVC. La descripción de los módulos aborda sus funciones generales y los detalles de las tecnologías usadas.

Para la aplicación móvil

Esta sección se centra en los componentes de la parte cliente. Esta es la aplicación para dispositivo móvil del usuario estudiante.

Templates

En este módulo hemos agrupado todas las vistas para la representación de la aplicación. Incluyen la pantalla de bienvenida, los formularios de registro y login, las etapas del juego, la representación de estados de los ejercicios y de los test, y la teoría.

Existe una vista principal llamada *index.html*. Todas las demás son vistas parciales de la vista principal. Aunque parezca que navegamos por distintas vistas, en realidad nos encontramos en una que va cambiando su contenido de manera asíncrona y parcial.

Las transiciones entre vistas parciales siguen una máquina de estados. Los estados más importantes de la aplicación móvil son:

- *Welcome*. Primer estado de la aplicación, se muestra un splashscreen con el logo de Biogam.
- *Login*. Segundo estado de la aplicación. En caso de no encontrarse la información de un usuario en la base de datos local, se mostrará un formulario para la autenticación manual del usuario.
- *Register*. En caso de no encontrarse la información de un usuario en la base de datos local, el registro será opcional si ya se posee una cuenta y obligatorio en caso contrario. La vista también será un formulario.
- *Home*. Se mostrará una lista de opciones con los temas disponibles.
- *Categories*. Dentro de cada tema podemos encontrar como mucho 3 categorías en ésta versión de la aplicación: *Theory*, *Exercises* y *Test*.
- *Theory*. Este estado está compuesto por dos vistas, una para la representación de la lista de todos los títulos de los conceptos de teoría disponibles para un tema, y otra para mostrar con mayor detalle el título de teoría elegido.
- *Exercises*. Este estado está compuesto por tres vistas. La primera muestra una clasificación por niveles de los ejercicios a manera de botones: nivel 1, nivel 2, etc. Cada uno de los botones de ésta vista muestra el número de ejercicios que componen el nivel, así como el número de ejercicios que llevamos hechos correctamente. Tras elegir uno de los niveles, la siguiente vista nos muestra una lista de ejercicios para hacer. Finalmente, otra vista nos presenta el ejercicio completo, con el enunciado y las opciones disponibles.
- *Test*. Este estado está compuesto por dos vistas. Una da acceso a las preguntas y las opciones de respuesta. Otra vista muestra la información relativa al test, como el número de aciertos y fallos.

- *Score*. Este estado está compuesto por una vista en la que se resume la calificación total obtenida en el test realizado. Aparece una vez se ha completado el test del tema. Esta vista tiene un botón que permite ver el test resuelto.

Cada estado, como ya se ha mencionado, tiene asociada una representación que es una de las vistas parciales. Todas corresponden con un HTML que se encuentra en el apartado 'Templates' de la aplicación cliente.

Controllers

El módulo donde se agrupan los controladores. En AngularJS también se puede definir un controlador para cada componente de la vista. En Biogam cada controlador está asociado sólo a una vista. Los controladores de los estados principales son:

- *WelcomeCtrl*. Encargado de determinar el tiempo en el que se representara el splashscreen de la aplicación al inicio de la ejecución. Es aquí donde se invocan los servicios para generar y poblar la base de datos según convenga.
- *LoginCtrl*. Encargado de recoger la información de usuario y verificar la cuenta en el servidor remoto.
- *RegisterCtrl*. Encargado de recoger la información de registro del usuario y añadirla en el servidor remoto.
- *HomeCtrl*. Encargado de pasar a la vista la lista de todos los temas disponibles en la aplicación. También almacena la elección del tema seleccionado por parte del usuario.
- *CategoriesCtrl*. Encargado de pasar a la vista la lista de todas las categorías disponibles en la aplicación. También almacena la elección de la categoría del tema.
- *TheoryCtrl*. Controlador para la vista de la lista de elementos de teoría que tenemos disponible para un tema. Obtiene la lista para el tema elegido y la representa.
- *CompleteTheoryCtrl*. Se encarga de representar la información detallada del concepto teórico antes seleccionado, pasando los enunciados y las imágenes a la vista.
- *LevelCtrl*. Se encarga de agrupar los ejercicios por niveles y pasar esta información a la vista junto con el número de ejercicios en cada nivel y el número de ejercicios resueltos correctamente para cada nivel.
- *ExerListCtrl*. Este controlador obtiene la lista de ejercicios correspondiente a un nivel.
- *ExerCtrl*. Se encarga de la representación y de la lógica para la resolución del ejercicio seleccionado en el estado anterior.
- *TestCtrl*. Se encarga de obtener y representar los test disponibles para un tema y de comprobar que no se haya resuelto antes.
- *ScoreTestCtrl*. Se encarga de facilitar los datos a la vista que se encarga de representar los resultados de los test.

Los controladores también se encargan de modificar el flujo de la aplicación dependiendo de los datos, llevándonos a estados distintos según convenga. También realizan las peticiones al servidor remoto.

Services

En este módulo agrupamos la lógica de negocio y tratamos con la información representada y con la que opera la aplicación. En Services se hacen las transacciones a base de datos, creando y

destruyendo datos en función de las necesidades del controlador que utiliza un servicio. Además contiene la lógica para realizar una petición al servidor remoto.

Todos los servicios están implementados en “services.js” y estructurados en factorías. Una factoría es un módulo JavaScript que agrupa funcionalidad según criterios definidos por el usuario. Por ejemplo una factoría que construya una casa tendrá funciones como: coloca puertas, coloca ventanas, construye techo, etc.

Para el servidor

Esta sección se centra en los componentes de la parte servidor. Estos prestan servicios centralizados como repositorio de información (ej. temas y ejercicios) a los clientes. Es también la parte donde el profesor puede revisar datos globales de sus alumnos.

Views

Esta parte es la encargada de representar toda la información necesaria para que el usuario administrador pueda interactuar con la parte del servidor.

Consta de dos vistas:

- *Login*. Formulario para la autenticación del usuario administrador.
- *Home*. Vista dinámica, que representa la información de cada usuario que usa la aplicación, usuario, nombre, apellido, estadísticas, etc.

Controllers

En este módulo se definen todos los controladores que rigen el comportamiento de las distintas rutas del servidor remoto.

- *Home*. Controlador de la página principal, se encarga de obtener los datos necesarios para su representación.
- *Register*. Controlador encargado de registrar a un nuevo usuario.
- *Login*. Controlador que comprueba las credenciales de cada usuario de la aplicación.
- *Store*. Controlador específico para almacenar la información que la aplicación tramita por cada usuario, progreso, nivel, categoría, fallos, intentos, etc.

Models

En este módulo agrupamos todos los modelos que serán utilizados por cada controlador para realizar las tareas que sean necesarias. Cada controlador contará con un modelo. Así pues, el controlador *Welcome.php* tendrá un *Welcome_model.php*, y así con los demás controladores.

Cada modelo contará con un grupo de funciones destinadas a resolver las tareas del controlador que lo usa. Por ejemplo *Register_model.php* tiene funciones como *insert_user()* y *get_user()*.

3.4. Comportamiento dinámico

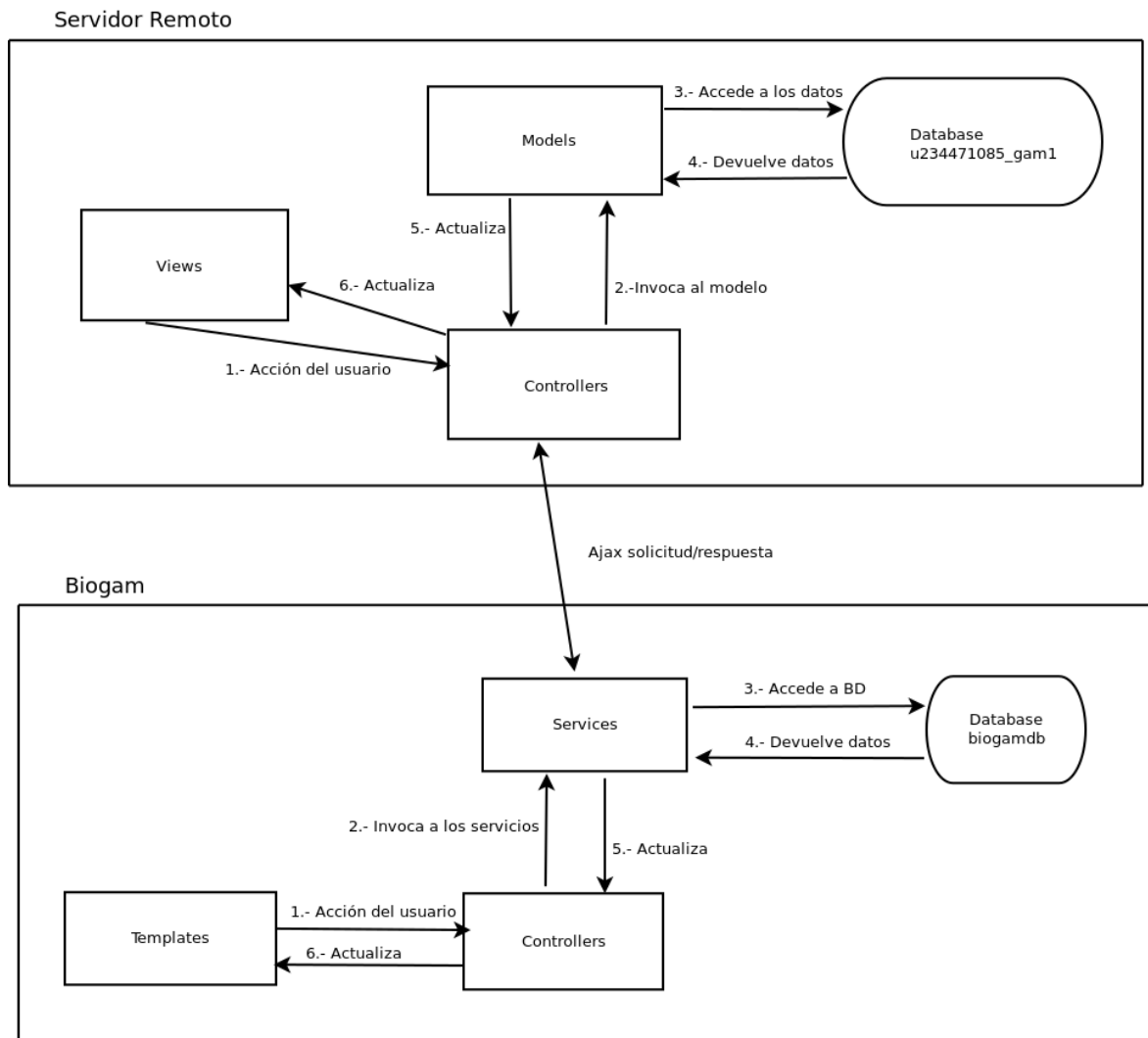


Figura 3. Arquitectura cliente-servidor

En esta sección se describe la interacción entre los componentes de la aplicación Biogam (ver sección 3.3) para proporcionar los servicios. Las siguientes secciones

Cliente

1. *Acción de usuario.* El usuario interactúa directamente con las distintas vistas:
 - **Introduciendo datos**
 - Registro en la aplicación.
 - Login.
 - Elección de tema, categoría y nivel.
 - Elección de apartado de teoría y de ejercicio en las listas que se proporcionan.
 - Introducción de datos en los ejercicios.
 - Elección de respuesta en los test.
 - **Recibiendo datos**
 - Teoría de los temas (información textual o en imágenes).
 - Feedbacks del estado de los ejercicios.
 - Resultados de los test realizados.

Estos datos son llevados y tratados en el controlador designado a la correspondiente vista. Éste proceso en AngularJS se realiza mediante el objeto `$Scope`, este objeto hace referencia a todas las expresiones que son alcanzables dentro de un mismo ámbito. Por ejemplo, dada una vista llamada `ejemplo.html` cuyo controlador es `ejemploCtrl.js`, todas las expresiones definidas dentro del objeto `$Scope` de este ámbito, serán alcanzables entre la vista y su controlador.

2. *Invoca a los servicios.* Una vez obtenidos los datos de entrada, el controlador hace uso de la capa de servicios de la aplicación para tratarlos y realizar las acciones pertinentes (cálculos, consultas a la BD, transformaciones, etc.).
Para usar un determinado servicio y las funciones incluidas en él, se tendrá que declarar antes en la cabecera del controlador.
3. *Accede a los datos.* Desde la capa de servicios (`services.js`), que actúa como el modelo de la aplicación, se realizan las transacciones a base de datos.
Los servicios son una de las piezas fundamentales de AngularJS. Se trata de definiciones que describen una funcionalidad que será reutilizada en varias partes de nuestro código. Cabe mencionar que en nuestra aplicación los servicios no sólo se limitan a hacer transacciones a la base de datos. Aquí también se encuentran otra serie de funciones útiles y agrupadas por objetivos:
 - **Funcionalidades de base de datos.** Aquí se encuentran las funciones para la creación, gestión y control de versiones de la base de datos.
 - **Creación de objetos complejos.** Una vez hecha la consulta a base de datos, con la respuesta obtenida se crean objetos complejos con la finalidad de retornar al controlador toda la información solicitada en una estructura de datos fácil de utilizar.
 - **Funcionalidades orientadas a la conexión.** Aquí se encuentran todas las peticiones de conexión al servidor remoto, que serán utilizadas por los controladores correspondientes.
4. *Devuelve datos.* La base de datos, responde a las peticiones realizadas por el modelo.
5. *Actualiza.* Una vez que la capa de servicios ha procesado u obtenido los datos, el resultado es devuelto al controlador que los solicitó.
6. *Actualiza.* Cuando el controlador obtiene los datos, estos son asignados a una expresión del objeto `$Scope`. De esta manera son alcanzables por la vista asignada a dicho controlador, pudiendo así modificar, escribir o eliminar datos que están siendo representados.

Servidor remoto

1. *Acción del usuario.* Al igual que en la aplicación móvil, el evento desencadenado por la acción del usuario es gestionado por un controlador concreto.
 - **Login.** Formulario de acceso para el administrador.
 - **Home.** La única vista donde se representa la información de todos los usuarios de la aplicación. En ella se puede observar por orden alfabético el nombre y las estadísticas de las categorías para cada tema.
2. *Invoca al Modelo.* El controlador asignado, dependiendo de si es de la vista del servidor o uno para gestionar las peticiones del cliente, invocará al modelo para cumplir con su tarea. Los controles de las vistas sólo invocarán funciones del modelo que se encargan de consultar en la base de datos. En cambio, los controladores para las peticiones del cliente

escribirán (alta de nuevo usuario), modificarán (actualizando los datos de la aplicación para cada usuario) y consultarán (comprobando las credenciales del login).

3. y 4. *Transacciones*. Los puntos 3 y 4 son una representación del proceso de transacción entre el modelo y la base de datos.
En el servidor el modelo sólo se limita a devolver la respuesta de las transacciones realizadas, y el controlador a llamar al modelo adecuado según convenga.
5. *Actualiza*. Los datos obtenidos en el modelo son devueltos al controlador que realizó la llamada.
6. *Actualiza*. Una vez el controlador obtenga los datos que solicitó, éstos serán devueltos a las vistas de la parte remota o a la aplicación móvil según convenga.
La respuesta por parte del controlador puede ser o bien los datos solicitados, o un mensaje del estado final de la consulta.

3.5.Conclusiones

El presente trabajo desarrolla la app objetivo siguiendo un modelo cliente-servidor. Éste facilita la distribución de los contenidos, y reduce las necesidades de almacenamiento en los dispositivos móviles clientes, que son más limitadas que en dispositivos convencionales. Así mismo, se incorpora una base de datos relacional donde se han tenido en cuenta los elementos de información necesarios para la app. Ello ofrece una estructura sólida y flexible de información, que favorece la consistencia a la hora de implementar la funcionalidad. Por otro lado la estructuración de los componentes software en concordancia con el comportamiento dinámico entre el cliente y el servidor remoto, hace posible que la aplicación tenga un flujo de funcionamiento lo más dinámico y estable posible. Esto facilita realizar futuras modificaciones de la funcionalidad ya proporcionada.

4. Manual de usuario

En esta sección se detallan los pasos que hay que seguir para navegar por la aplicación y realizar las distintas tareas. Se divide por pantallas y se muestra una captura de cada situación para facilitar la guía.

4.1. Login y registro

Una vez instalada la aplicación en el dispositivo móvil, al abrirla aparecerá una primera pantalla (ver Figura 4) en la que existen dos opciones: logarse y registrarse. El usuario se loga introduciendo su correo electrónico y contraseña. Para ello ha debido registrarse previamente en la pantalla de la Figura 5. El login se contempla para el caso de que el usuario desinstale la aplicación y posteriormente la vuelva a instalar.

Figura 4. Vista de login.

Figura 5. Vista del registro.

4.2. Menú principal y categorías

Una vez hecho el registro o el login, la aplicación nos llevará a la pantalla principal (ver Figura 6). En ella tendremos dos temas de genética para elegir, *One Locus* o *Mitosis*. En este manual se va a tomar como ejemplo el tema de *One Locus*. El tema de *Mitosis* es similar, aunque sólo cuenta con las categorías teoría y test.

Tras pulsar el tema *One Locus* la aplicación nos lleva a la siguiente pantalla (ver Figura 7). En ella podemos elegir realizar una de las categorías propuestas, teoría (ver sección 4.4), ejercicios (ver sección 4.5) o test (ver sección 4.6). Podemos movernos entre partes de la aplicación usando los controles de navegación (ver sección 4.3).

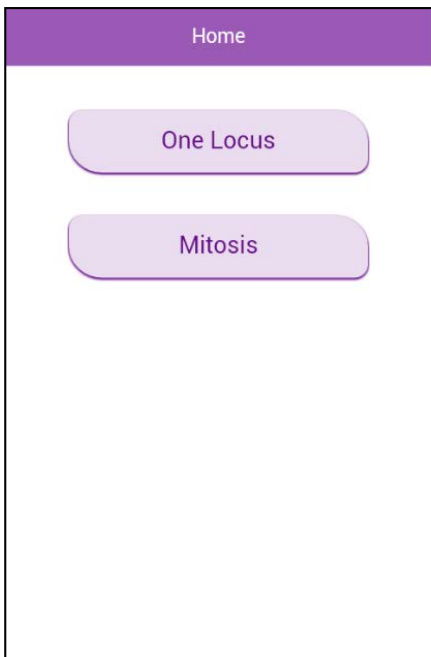


Figura 6. Vista de la pantalla de inicio “Home”.

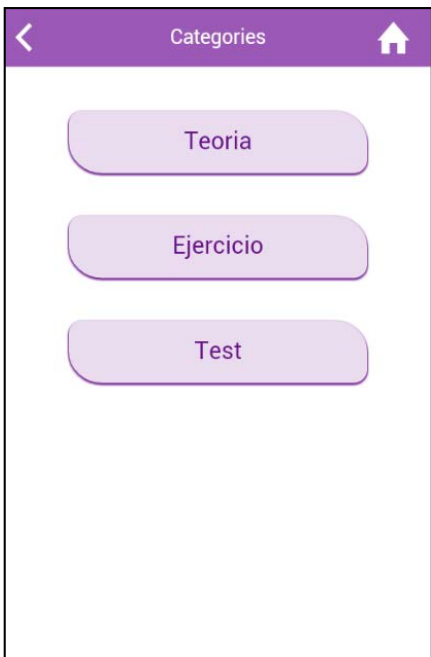


Figura 7. Vista donde se muestran las categorías de cada tema.

4.3.Navegación

Para navegar por la aplicación a vistas que ya se han visitado anteriormente hay que utilizar los botones de retroceso como muestra en la Figura 8, o bien el botón “Home”, como muestra la Figura 9.



Figura 8. Botón navegación hacia atrás.



Figura 9. Botón de acceso directo al menú principal “Home”.

4.4.Teoría

Lista de teoría

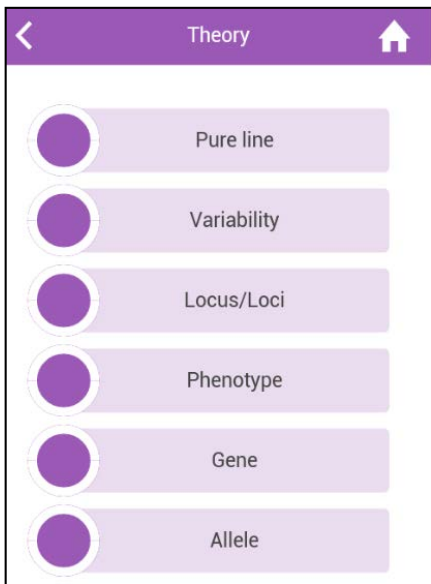


Figura 10. Vista con listado de teoría.

Desde la pantalla de categorías (ver Figura 7) pulsaremos en la opción teoría. Al hacer dicho gesto, la aplicación nos lleva a una pantalla como la que se muestra en la Figura 10. En esta pantalla aparece una lista con los distintos apartados de teoría que pertenecen al tema *One Locus*.

Detalle de teoría

Al pulsar sobre alguno de los apartados, que se muestran en la Figura 10, pasamos a una nueva pantalla. En ella se muestra la teoría completa como se muestra en la Figura 11. Si queremos pasar al siguiente apartado de la lista anterior, simplemente tendremos que deslizar el dedo hacia la izquierda. De esta forma nos desplazaremos de un apartado a otro de la teoría sin necesidad de volver una y otra vez a la pantalla donde aparece el listado de apartados. Para saber cuántos apartados de teoría quedan o cuántos se han visitado ya, podemos fijarnos en los círculos que aparecen en la parte superior de la vista.

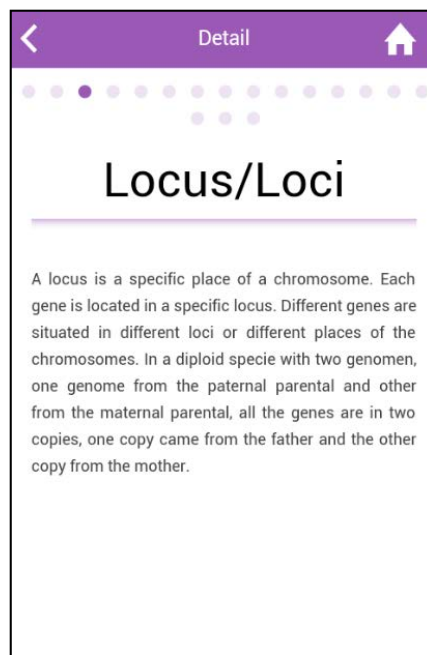


Figura 11. Vista con detalle de teoría.

4.5.Ejercicio

Lista de niveles

Una vez en la pantalla, pulsamos en la opción “Ejercicio”. Esto nos llevará a una nueva pantalla como la que muestra la Figura 12, donde aparece una lista de niveles. Se podrá acceder a los niveles de manera incremental, es decir, a medida que vayamos resolviendo ejercicios se desbloquearan nuevos niveles.

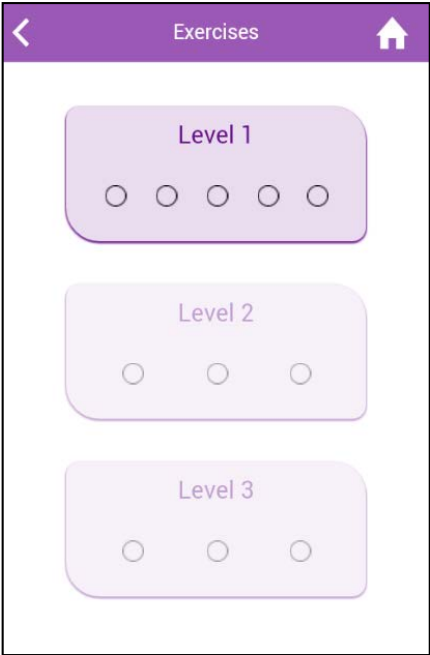


Figura 12. Vista con listado de niveles de ejercicios.

Lista de ejercicios

Al pulsar sobre algún nivel que esté disponible, la aplicación nos llevará a una nueva pantalla como la que muestra la Figura 13. Esta pantalla muestra una lista de ejercicios.

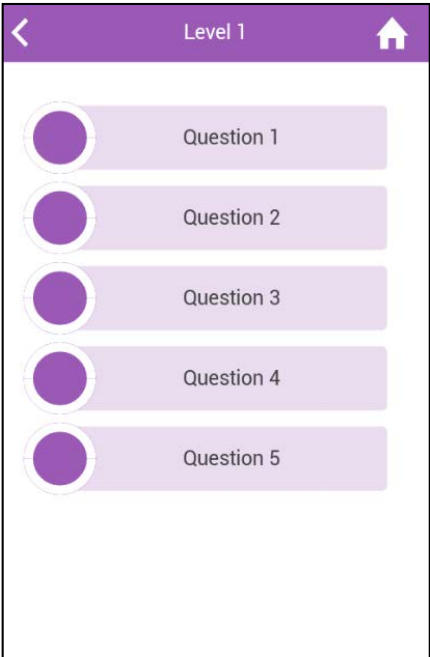


Figura 13. Lista de ejercicios de un nivel.

Ejercicios

Los ejercicios se pueden hacer en cualquier orden. Tras seleccionar uno de los ejercicios, la aplicación muestra una nueva pantalla como la de Figura 14. En ella aparece el título del ejercicio, un enunciado y un bloque de seis botones. Después de leer el enunciado, el usuario deberá elegir uno de los 6 botones. No se pueden seleccionar varios botones a la vez.

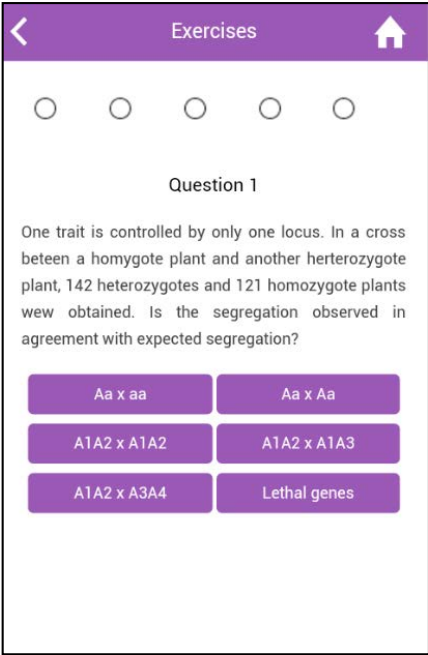


Figura 14. Vista inicial de ejercicio.

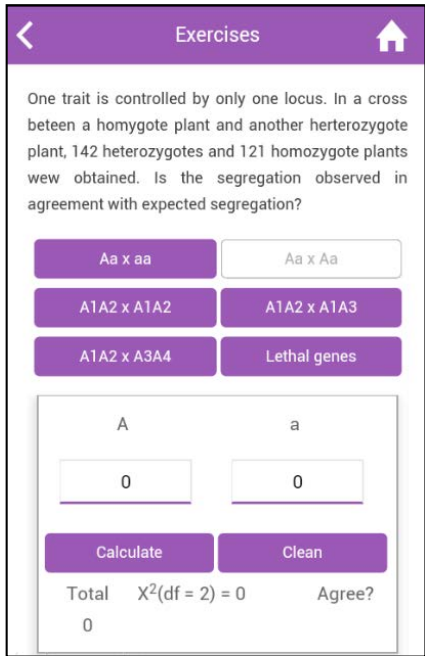


Figura 15. Vista de ejercicio con calculadora desplegada.

Cuando pulsamos en uno de los botones, aparece una caja con la calculadora de la aplicación como se muestra en la Figura 15. Se introducirán valores en los cuadros de texto. A continuación, hay que pulsar el botón “Calculate” para que la calculadora muestre un resultado. Los resultados aparecen más abajo, y dan información del cálculo realizado. Si el cálculo es incorrecto y se quiere limpiar la calculadora, basta con pulsar el botón “Clean”. De igual modo que con la teoría, si se quiere pasar de un ejercicio a otro, basta con deslizar el dedo sobre la pantalla hacia el lado izquierdo. Así mismo, en la parte superior de la vista aparecen unos círculos que indican en qué parte de la lista de ejercicios nos encontramos.

4.6.Test

Preguntas

Una vez en la pantalla de categorías (Figura 7), pulsamos en la opción “Test”. Esto nos llevará a una nueva pantalla como la que muestra la Figura 16, donde aparece una nueva ventana con la primera pregunta del test.

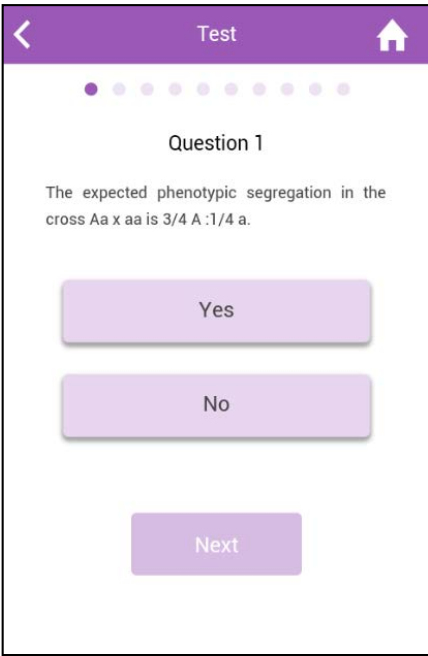


Figura 16. Vista de pregunta de test sin resolver.

Para realizar los test hay que elegir la opción que se considere correcta. En el momento de la selección se mostrará si la opción es correcta (ver la Figura 17) o incorrecta (ver Figura 18). Tras la corrección de la pregunta, para pasar a la siguiente pregunta pulsamos sobre el botón “Next”.

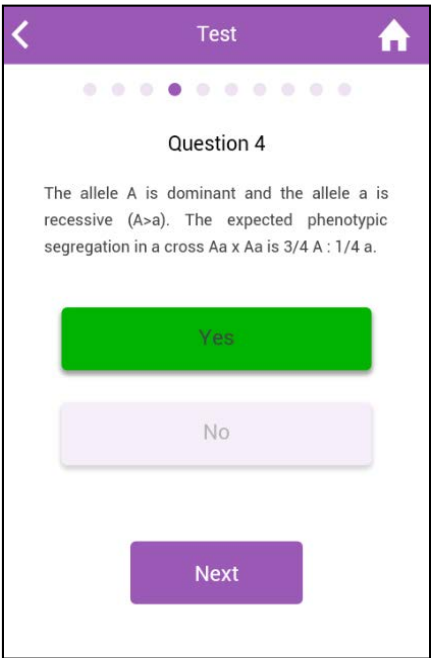


Figura 17. Vista solución a pregunta de test correcta.

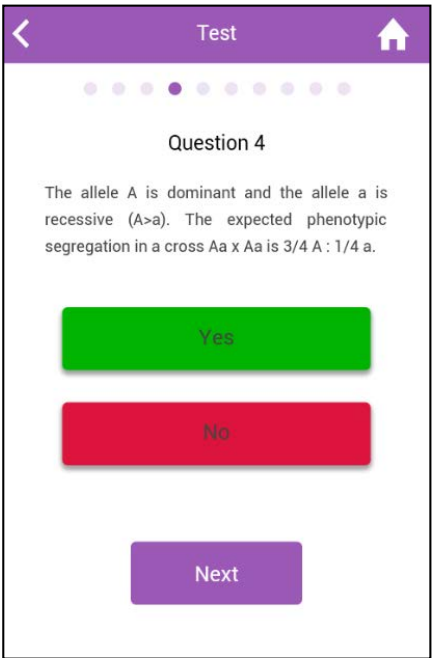


Figura 18. Vista solución a pregunta de test incorrecta.

Resultados

Al finalizar el test aparecerá una nueva pantalla como la que se muestra en la Figura 19. Donde se pueden ver los resultados obtenidos en el test. Además, si se desea revisar el test, puede hacerse pulsando el botón “View Test”.

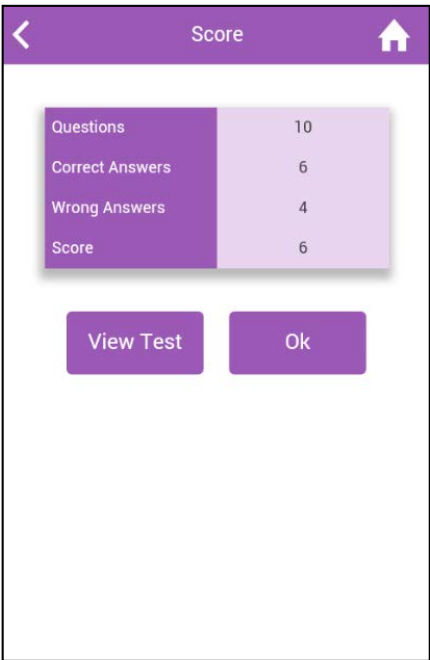


Figura 19. Vista de resultados del test.

En esta sección se ha descrito la integración del usuario con la app Biogam. Las funciones principales son el acceso del usuario a la aplicación, mediante el login o registro, y la interacción con los distintos contenidos de teoría ejercicios y test.

La interfaz de Biogam es uniforme siguiendo las líneas de diseño de Material Design (ver “Diseño de interfaces”).

5. Manual de administrador

En esta sección se detallan los pasos que tiene que seguir el administrador, en este caso el profesor, para poder ver la actividad de los usuarios de la aplicación Biogom. La aplicación de administrador es una aplicación web alojada en el servidor y está pensada para ser accedida a través de ordenador.

Se divide por pantallas y se muestra una captura de cada situación para facilitar la guía.

5.1.Login

En esta sección se muestra el login de acceso a la aplicación de administrador (Figura 20). Para poder entrar el administrador introducirá un usuario y contraseña predefinidos. Una vez logeado, accederá a la pantalla principal de administrador.

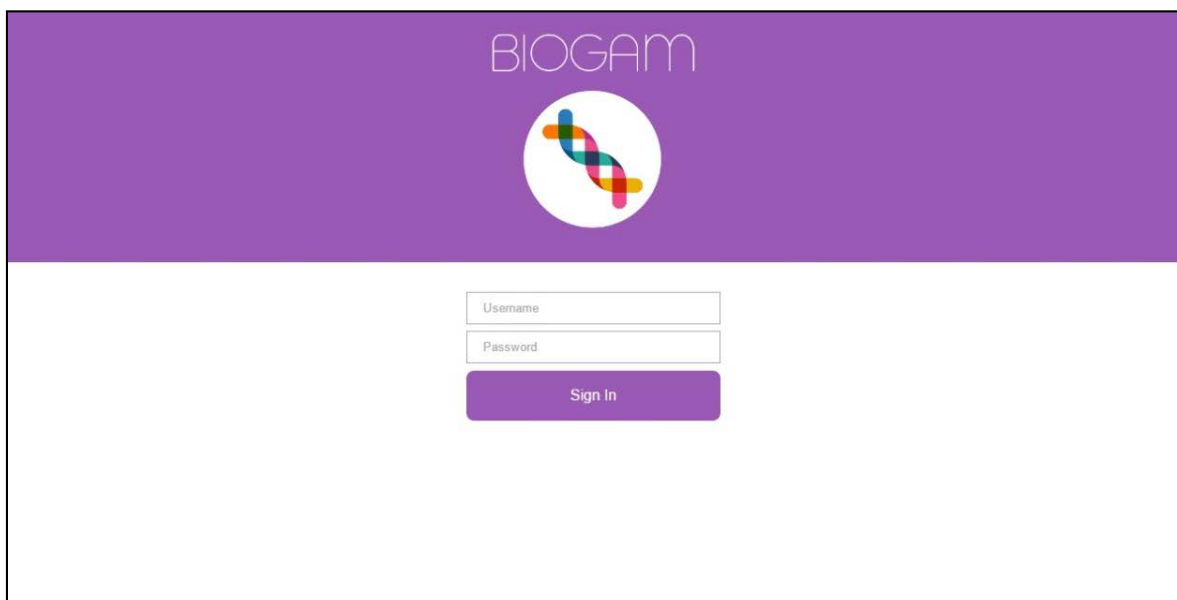


Figura 20. Login de la aplicación de administración.

5.2.Pantalla principal

En esta sección se muestra la pantalla principal de administrador. Como se puede ver en la Figura 21, el administrador puede seguir la actividad de los usuarios. Puede ver si un ejercicio se ha hecho o no, cuantas veces se ha intentado y el número de veces que se ha fallado. Además también se muestra la puntuación de los test que el usuario ha realizado.

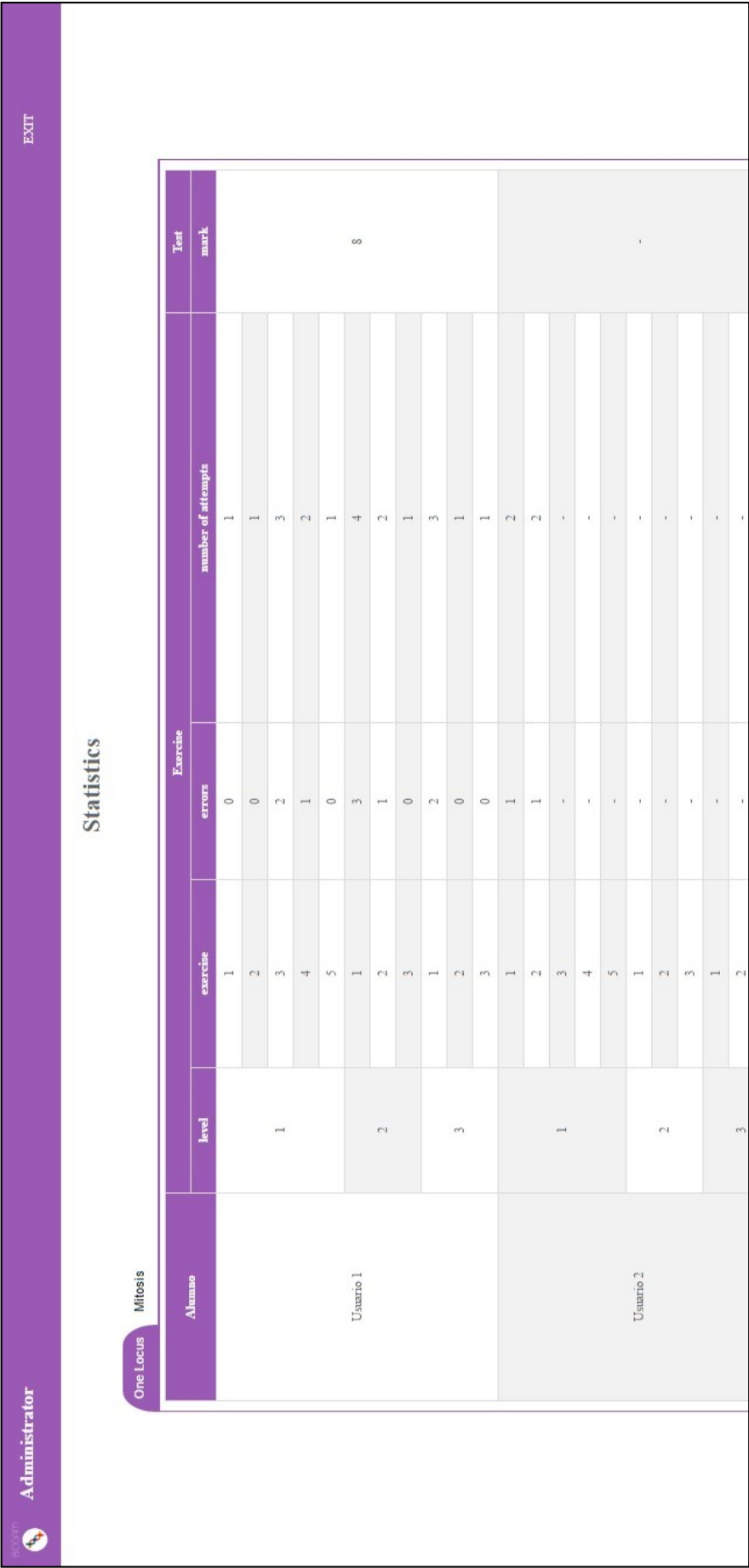


Figura 21. Pantalla principal de administración.

6. Conclusiones

En esta sección se recogen las conclusiones del trabajo realizado. Se discuten las principales decisiones sobre la arquitectura de la aplicación y el diseño de la funcionalidad (ver sección 6.1). También, se habla del trabajo futuro sobre Biogam (ver sección 6.2). Finalmente, se resume lo que hemos aprendido gracias a la realización de este proyecto (ver sección 6.3).

6.1. Resultados del proyecto

Como producto final de todo este proceso de desarrollo se ha obtenido la aplicación Biogam orientada a enseñar genética. Con ella los estudiantes podrán visualizar teoría de forma sencilla e intuitiva, superar ejercicios de diferentes niveles de dificultad y, finalmente, poner a prueba sus conocimientos con los test.

La anterior funcionalidad se logra mediante una arquitectura cliente-servidor. El cliente es una aplicación móvil multiplataforma de tipo híbrido, con parte nativa y parte web. Envía datos periódicamente a un servidor. Éste los procesa para actualizar la información (ej. teoría, ejercicios y test) con la que trabaja cada usuario de la app, y mantener estadísticas.

Aunque se ha seguido la arquitectura cliente-servidor, la lógica de la aplicación móvil es totalmente independiente del servidor. El servidor es necesario para la comprobación de credenciales al iniciar la aplicación, y para actualizar de manera periódica la información con que trabaja el cliente. El resto del tiempo la app podría funcionar de manera autónoma.

Como se ha mencionado anteriormente, la aplicación también está pensada para que recoja datos estadísticos del progreso de los alumnos. Esta información se mostrará en una aplicación web donde el administrador podrá realizar un seguimiento de lo que los alumnos, usuarios de la aplicación móvil, han resuelto o no. De esta forma se podrá estudiar en qué apartados del test o de los ejercicios que se plantean han tenido más dificultades. Esta información será relevante a todos los niveles para el diseño de los nuevos materiales.

La aplicación se ha diseñado y desarrollado para que sea mantenible y se pueda incrementar su funcionalidad. En este sentido se ha adoptado la ya mencionada arquitectura cliente-servidor y seguido el patrón MVC. Con este patrón, agregar una nueva funcionalidad consistiría en añadir o modificar una vista y desarrollar un controlador acorde a las necesidades de la vista, y que se valga de un servicio adecuado para cumplir sus objetivos.

En la medida de lo posible se han seguido también los convenios para el uso de los diferentes frameworks empleados. Estas hacen referencia a aspectos diversos como directivas, funciones, declaración de variables, u organización del proyecto. Con ello se persigue obtener un código legible para futuras modificaciones.

El aspecto visual de la interfaz, tanto web como de la aplicación móvil, se ha realizado siguiendo las 10 heurísticas de Jakob Nielsen (Nielsen, 1995), para así poder ofrecer al usuario de la aplicación la mejor experiencia de uso posible. Además, el diseño es sencillo e intuitivo, siguiendo las pautas del *Material Design*.

La aplicación ha sido desarrollada a partir de la experiencia en la asignatura de Genética. Sin embargo, la estructura de los temas y el tipo de materiales son extrapolables a otras materias.

6.2.Trabajo futuro

Como trabajo futuro, se plantean mejorar ciertos aspectos de la aplicación e introducir nuevas funcionalidades. Se resumen a continuación.

En primer lugar, hay que incluir en la aplicación el resto de temas genéticos que quería el cliente. Esto no se ha abordado en esta versión debido a la falta de requisitos iniciales sobre ello. En los temas abordados, se disponía de versiones iniciales de las aplicaciones realizadas por el profesor César Benito Jiménez. Para poder realizar el resto de los temas, se deberá reunir el equipo con el profesor para determinar los nuevos requisitos que deberán cumplir las aplicaciones. Para ello se seguirá un proceso igual al que se ha seguido en el presente trabajo.

En segundo lugar, se debe completar la funcionalidad del administrador. Por el momento sólo puede examinar datos de los test que han realizado los alumnos. Se plantea introducir un apartado para que el administrador de la aplicación pueda añadir y modificar contenido a la aplicación, es decir nuevos ejercicios, nuevas preguntas tipo test, nuevos apartados de teoría. En la actualidad esto se hace introduciendo estos elementos directamente en la base de datos utilizando una interfaz genérica relacionada con el lenguaje SQL. La ampliación proporcionaría una interfaz específica adaptada al usuario profesor, que no tiene que ser experto en bases de datos.

Por último se plantea hacer pruebas con alumnos para poder recoger información sobre la usabilidad de la aplicación, y evaluar su satisfacción al usar la aplicación. Además, también de cara al futuro de la aplicación, la realización de pruebas con usuarios podría mejorar la funcionalidad de la aplicación. Por ejemplo, se podría concluir que hay que añadir nuevos elementos que previamente no se habían tenido en cuenta. Para poder realizar esta evaluación del usuario y obtener información para la mejora de la aplicación utilizaríamos métodos de observación. Estos consisten en observar y anotar los pasos que realiza el usuario para moverse por la aplicación, sin proporcionarle guía. Por último, entregaríamos al usuario un cuestionario con preguntas básicas sobre la experiencia obtenida al utilizar la aplicación.

6.3.¿Qué hemos aprendido?

Para este proyecto hemos aprendido nuevas tecnologías con las que desarrollar aplicaciones híbridas. En particular, todos los frameworks utilizados en el desarrollo de Biogam no los habíamos visto previamente.

Como punto de partida tomamos Ionic framework, que sería la capa superior de todo el proyecto. Utiliza AngularJS y Apache Cordova para generar la aplicación híbrida, y este mismo framework es el que establece los cimientos del proyecto.

Una vez generada la base de nuestro proyecto, tuvimos que profundizar para saber cómo funciona el desarrollo de una aplicación en AngularJS. AngularJS es un framework que sirve para desarrollar cualquier tipo de aplicación web que sigan las bases de diseño MVC. En esta parte de desarrollo de la aplicación se obtuvo conocimientos sobre la estrategia de diseño de aplicaciones web, el lenguaje JavaScript, el uso de directivas, y tecnologías para la creación de aplicaciones interactivas con AJAX.

También se puede mencionar el uso del lenguaje SQL. Éste se utilizó para la creación de bases de datos en MySQL y SQLite y la inserción de datos.

Para el aspecto visual se han adquirido conocimientos orientados al diseño responsive. También sobre usabilidad y los estándares fijados para este tipo de aplicaciones, por ejemplo el botón de volver siempre a la izquierda, los gestos al navegar, o los colores para indicar estados.

El contacto con Apache Cordova ha sido menor. Hemos tenido que indagar cómo funciona el framework, pasando por entender la estructura de directorios y manipular los ficheros de configuración JSON. Para algunos miembros del equipo, éste fue su primer contacto con este nuevo formato, que es el estándar actual para el intercambio de datos, especialmente en aplicaciones web. En general, el uso de Apache Cordova se ha limitado a instalar los plugins disponibles para tener acceso a los recursos del dispositivo que solo son accesibles de manera nativa, y generar el proyecto.

También hay que mencionar la implementación de la lógica que alberga el servidor. Ésta ha sido desarrollada en lenguaje PHP orientada a objetos.

En resumen, más que algunos lenguajes o frameworks, lo que se ha aprendido en este proyecto ha sido un marco general para el desarrollo de aplicaciones webs. Este marco sigue el patrón MVC, y utiliza tecnologías comunes en el desarrollo cliente-servidor con clientes móviles.

Conclusions

This section contains the conclusions of the project. The main decisions concerning the architecture of the App and the design of the functionality are discussed in this section (see section “Project outcomes”). There is also a section where the future work on Biogam is detailed (see section “Future work”). Finally, there is a summary of everything learned during the realization of this project (see section “What have we learned?”).

Project outcomes

As a result of the process performed, we have an application called Biogam which is focused on teaching genetics. With this application students can visualize theory in a very simple and intuitive way, do exercises with different levels of difficulty, and test their knowledge with the tests provided by the app.

The previous functionality is obtained following a client-server model. The client is a mobile application with a multiplatform solution; which includes a native and a web part. It sends data periodically to a server. These data are processed to update the information (e.g. theory, exercises, and test) with which each user of the app works, and to collect statistics.

Although the model followed is client-server, the logic of the mobile application is independent from the server. The server is necessary to check the credentials when opening the app for the first time, and to update periodically the information with which the client works. The rest of the time the app is able to work independently.

As mentioned earlier, the app is also developed to collect statistical data of the students' progress. This information will be displayed in a web app where the administrator will be able to track the solved exercises and tests done by students. In this way, the different sections of the tests and the exercises can be studied in order to spot the ones that were more difficult for the students. The information obtained from this study will be useful for the design of new materials.

The app has been designed and developed so it can be maintained over time and its functionality increased. The app has a client-server model and follows the MVC pattern. With this pattern, adding new functionality will consist in inserting or editing a new view and developing a controller according to the view requirements, and that uses an appropriate service to fulfill its objectives.

As far as possible, the use of the different frameworks followed their guidelines. These guidelines are related to several aspects like directives, functions, variable declarations or the project organisation. This is intended to obtain a legible code for future modifications.

The visual appearance of the mobile app interface and the web app, has been designed following the 10 heuristics of Jakob Nielsen (Nielsen, 1995). This allows providing users the best experience while using the app. In addition, the design is simple and intuitive and it follows the standards of *Material Design*.

The app has been developed based on the experience in the subject of Genetics. Nevertheless, the structure of the subjects and the type of materials can be used also for other subjects.

Future work

As future work we outline certain aspects of the app, as well as introducing new functionalities. We summarize these below.

In the first place, the remaining genetic subjects that the client asked for should be included. This couldn't be accomplished in this version of the app due to missing requirements that the client didn't specify in the beginning. In the subjects fulfilled, we had older versions made by Professor César Benito Jiménez from which we could guide our work. In order to finish the subjects that couldn't be developed, the team must call a meeting with Professor César Benito to determine the new requirements that the App must fulfil. So as to do this task the process will be the same as the one followed in the present project.

In second place, the functionality of the administrator must be completed. At the moment, the administrator can only examine the data obtained from the tests solved by the students. It states introducing a new section where the administrator can add and edit the App content that is, new exercises, new test questions and new theory. At present this is made by introducing these elements directly in the data base and using a generic interface related to SQL language. This expansion will provide a specific interface which adapts to the user teacher, because he doesn't need to be an expert in data bases.

Finally we state to do student tests in order to collect as much information as possible about the usability of the App and to evaluate their satisfaction when using the App. For the future of the App we consider that making tests will improve the functionality of it. For example, adding new elements that were not present in the previous version of the App. In order to obtain information to improve the app we will use observational methods applied to the students that are being evaluated. These methods consist in observation and annotation of the steps taken by the user to navigate through the application without guiding them. At the end of the test the user will be given a form with basic questions to determine how satisfactory the experience of the user while using the app was.

What have we learned?

For this project we have learned new technologies with which hybrid apps are developed. Particularly, we did not study previously none of the frameworks used to develop Biogam.

As a starting point we used Ionic framework for the upper layer of the project. It uses AngularJS and Apache Cordova to generate the hybrid app. This framework also establishes the foundations of the project.

Once generated the basis of the project, we had to analyze in depth the development of an AngularJS app to learn the way it works. AngularJS is a framework to develop any web app that follows the MVC design. In this part of the development, we got information about the designing strategy used in web apps, JavaScript language, and directive usage and technologies used to develop interactive AJAX apps.

We also learned the usage of SQL. This language was used to create the databases in MySQL and SQLite and to store data.

Regarding the visual appearance we have learned how to include Responsive Design in our views. We also gained knowledge in usability and the standards for this kind of apps, for example the navigation button used to return to the previous view that always has to be on the left of the view, the gestures used to navigate through the app, or the colors used to indicate states.

The contact with Apache Cordova has been less. We had to investigate how the framework works, which involves understanding the directory structure and manipulating the files to configure JSON. For some of the members of the team, this was the first contact with JSON, which is the actual standard used for the exchange of data, especially in web apps. Moreover, we used Apache Cordova to install the plugins necessary to access to the resources provided by the device, as these can only be accessed in a native way, and to generate the project.

We also have to mention the implementation of the logic hosted in the server. This logic has been developed in the object-oriented language PHP.

To summarize, other than using some programming languages or frameworks, we have learned a general framework for the development of web apps. This framework has followed the MVC pattern and it uses common technologies applied in the development of a client-server model combined with mobile clients.

DESCRIPCIÓN DEL TRABAJO INDIVIDUAL

María Teresa Calvo

Mi contribución a este TFG ha sido, sobre todo, en las bases de datos.

A la hora de decir cómo íbamos a almacenar los datos, llegamos a la conclusión de que una base de datos relacional era la mejor opción. Necesitábamos un sistema de almacenamiento en el que se tuviera en cuenta las dependencias de unos datos con otros (ej. un ejercicio pertenece a un tema y a un nivel, en un nivel hay varios ejercicios, etc.). El diseño e implementación de bases de datos relacionales ha sido algo que hemos abordado durante el grado.

Al tener una estructura cliente-servidor, necesitábamos una base de datos remota, alojada en un servidor, y otra base de datos local, alojada en el propio dispositivo cliente. Para que no hubiera problemas con la sincronización de datos, decidimos que ambas fueran idénticas en estructura. La única diferencia entre ellas es que la base de datos local guarda la información de un usuario, mientras que la base de datos remota guarda información de todos los usuarios de la aplicación.

El diseño de la base de datos duró aproximadamente dos semanas, con reuniones periódicas con mis compañeros para explicarles qué significaba cada entidad y las relaciones que las unían. Cuando estuvo terminada, tuvimos una reunión con el profesor para que le diera el visto bueno. Con la aprobación del profesor, se desarrolló el modelo entidad-relación definitivo.

Para la base de datos remota tuvimos varios inconvenientes. El primero fue la tarea de buscar un host para alojarla. Buscábamos un host gratuito, y encontramos uno llamado Hostinger (www.hostinger.es/). Con el paquete gratuito se incluye el alojamiento de una base de datos relacional y una aplicación, pero no permite conexión remota a la base de datos. Por esta razón decidimos utilizar como servidor una máquina virtual que nos proporcionó la facultad.

Previamente al diseño de la base de datos, todos por separado realizamos un estudio de las aplicaciones que había implementado el profesor César Benito para descubrir problemas más profundos que los que pudimos sacar de la reunión con él. Además, también por separado, hicimos un estudio de aplicaciones de aprendizaje para descubrir las tendencias en este tipo de aplicaciones y poder incluir sus principales hallazgos en la nuestra.

Después de esto, nos reunimos con el director del proyecto, el profesor Rubén Fuentes, para poner en común todo lo anterior y elaborar una lista de requisitos.

Por otro lado, en el desarrollo de la aplicación Biogam, todos los integrantes hemos participado en la elaboración de la funcionalidad. La única excepción ha sido la funcionalidad de los ejercicios y del estilo de la aplicación, implementados por Ana Sanz. En mi caso, he contribuido especialmente en el desarrollo de la funcionalidad de los test, teoría, login y registro. Además he implementado las consultas a la base de datos.

En cuanto al test, he creado las consultas para obtener las preguntas de test, para un tema determinado y con sus opciones. He colaborado en la implementación de las vistas de test para que se muestren las opciones correctas e incorrectas. Además, he implementado la funcionalidad para mostrar los resultados finales de un test: número de preguntas, número de aciertos, número

de fallos y puntuación. Una vez que se acaba el test, hay que almacenar los resultados en la base de datos con una query de inserción, que también he implementado.

Con respecto a la teoría, he implementado las consultas a la base de datos y parte de la visualización de los datos obtenidos.

En el registro y login, he implementado las consultas para insertar un usuario en la base de datos. También la parte del CSS y funcionalidad que se encarga de mostrar los errores en el registro o login.

Cuando casi toda la funcionalidad de Biogam estaba terminada, yo dejé el desarrollo para empezar a escribir la memoria. Mis secciones han sido: “Introducción: Antecedentes, Objetivos y Plan de Trabajo”, “Estado del arte y requisitos”, “Modelo de datos”, “Manual de usuario” y “Manual de administrador”. Además, he contribuido en “RESUMEN”, “DESCRIPTORES”, y en parte de la traducción de “ABSTRACT”, “KEYWORDS” y “Planning”.

Cabe destacar el estudio exhaustivo de aplicaciones móviles y herramientas educativas que hice para realizar el apartado “Trabajo relacionado”. En concreto hice los siguientes análisis.

Realicé pruebas con distintas herramientas LMS y LCMS para entender mejor sus características y funcionamiento. En el caso de estas herramientas, tuve la suerte de haber utilizado plataformas como Moodle o Sakai en la universidad, por lo que su estudio fue relativamente sencillo.

En cuanto a los MOOC, no había realizado nunca uno de estos cursos impartidos por las universidades, pero mi compañera Ana Sanz sí y me ayudó bastante a la hora de estudiarlos. Yo me centré más en las plataformas Codecademy y Udacity, buscando información sobre ellas y haciendo algún curso. Hice un curso en Codecademy sobre Git que nos resultó muy útil, ya que la mayoría no habíamos utilizado Git y lo necesitábamos para llevar un control de versiones para la aplicación Biogam.

Para el estudio de las aplicaciones móviles, recurrí a la categoría de educación de la tienda de aplicaciones Google Play (Google_Play, 2016). Me descargué y probé varias aplicaciones para tener una idea clara de lo que hay actualmente en el mercado y cuáles eran las tendencias más usadas. Probé también aplicaciones centradas en la genética, y me sorprendió el hecho de que ninguna de las que probé ofrecía al usuario algo que no fuera información teórica sobre el tema. Por ello, consideramos que Biogam es una aplicación novedosa dentro de las aplicaciones móviles para la enseñanza de genética, ya que permite al usuario interactuar con la aplicación y probarse a sí mismo.

Gian Marco Díaz

Mi trabajo comenzó con el estudio de partida sobre el funcionamiento del framework que se encargaba de generar el marco del proyecto, Ionic framework. Mi tarea en esta etapa fue probar la gama de opciones que ofrecía el framework y obtener una idea general de cómo funcionaba. Entre todas las opciones disponibles por Ionic hicimos un uso mayoritario de:

- “add” que añadía la plataforma para la cual estaba destinado la aplicación (Android, IOS, etc);
- “serve” que montaba un servidor ligero para hacer las pruebas del código desarrollado con Angularjs;
- “build” que compilaba el proyecto web y generaba la aplicación nativa para las plataformas añadidas previamente; y
- “service” que nos permitía gestionar los paquetes y plugins que íbamos necesitando a lo largo del desarrollo.

El estudio de AngularJS fue una actividad que desarrollamos por separado. En este caso el centro de mi estudio fue la organización de los proyectos con este framework, sus herramientas de desarrollo y las funcionalidades que ofrecía.

Después que el grupo de trabajo creara los primeros prototipos de la aplicación en papel, colaboré con un primer prototipo funcional haciendo uso de la herramienta Ionic Creator. Ésta es una herramienta que permite crear prototipos basados en Ionic framework a través de un entorno gráfico. El prototipo funcional nos ayudó a tener una idea más clara de la configuración de la aplicación. Sin embargo, la herramienta fue descartada por sus limitaciones en el proceso de diseño, que a partir de ese momento recayó sobre el miembro del grupo Ana Sanz.

El diseño de la base de datos fue casi en su totalidad un trabajo realizado por María Teresa. Mi aportación fue la identificación de entidades con sus atributos, e ideas sobre las relaciones que deberían existir.

Dentro del diseño de las vistas de la aplicación, mi trabajo consistió en crear los componentes reutilizables. Algunos de ellos fueron el marcador de los ejercicios, los sliders para la navegación, o los fragmentos de código que generan los botones (no el diseño).

Para la implementar la funcionalidad de la aplicación, la primera tarea que realicé fue la configuración de los estados de la aplicación. Los iniciales fueron: Home; Categorías; Teoría; Ejercicios y Test. Cada estado contaba con su respectivo controlador y una vista asociada. Posteriormente, la complejidad de los estados crecería y sería modificada por los otros miembros del proyecto.

El desarrollo de estos estados, sus controladores y las vistas, fue un trabajo en paralelo. Nos distribuimos el trabajo entre las vistas, la lógica del controlador y los servicios de los que hacía uso e implementaban la funcionalidad. Aquí me ocupé principalmente del apartado de teoría, con la carga de texto e imágenes, y el control de la navegación a través de todos los puntos de la teoría de cada tema. También colaboré en gran parte del desarrollo del controlador de los test, principalmente abordando la lógica para la visualización de los aciertos y fallos. Por otro lado hice la lógica para mostrar los ejercicios que se habían hecho.

Para poder trabajar con la base de datos local de la aplicación móvil, se utilizó un plugin de Apache Cordova, *Cordova-sqlite-storage*. Éste permite abrir y/o usar bases de datos SQLite en Android, IOS y Window con HTML5. Me encargué de desarrollar el script para la creación de la base de datos partiendo del esquema entidad-relación ya generado y la inserción de datos.

En la conexión con el servidor remoto, mi tarea fue implementar las peticiones AJAX para los controladores del servidor y la representación de las respuestas por parte del servidor. También en la lógica del servidor, cree los controladores para las peticiones de la aplicación móvil y los propios de la web del administrador.

Además, he contribuido desarrollando los modelos e implementando la funcionalidad de la que harían uso los controladores para realizar sus tareas. También he adaptado la base de datos que usaría el servidor remoto.

La configuración del servidor remoto ha sido otra de mis tareas. He usado herramientas que trabajan estableciendo conexiones SSH como puTTY y FileZilla. Entre las tareas realizadas aquí está crear la jerarquía de carpetas para el correcto uso del servidor desplegado en el host *ingenias.fdi.ucm.es*.

Dentro del trabajo de la memoria participé principalmente en la introducción y descripción de la arquitectura usada para el desarrollo del proyecto. Esto incluyó los componentes identificados y el su flujo de trabajo en la aplicación.

Ana Sanz

Primero realizamos el análisis en profundidad de la aplicación. Me centré en localizar los puntos débiles de usabilidad en la interfaz de la aplicación inicial propuesta por el profesor. Para ello tuve en cuenta las tendencias en diseño de interfaces actuales (como Material Design), y apliqué conocimientos obtenidos a lo largo de la carrera. Además estuve pensando cuál era la forma más eficiente de traducir el código que había generado el profesor utilizando la aplicación App Inventor 2. Esta aplicación no genera código en ningún lenguaje en concreto, por lo que había que estudiar el significado de la unión de los elementos con los que describe el código.

A continuación estuve haciendo el estudio del arte. Cada uno de los miembros buscó información para luego hacer una puesta en común. Busqué información sobre las tecnologías que se utilizan en el presente trabajo.

Al ser la primera vez que íbamos a trabajar con Apache Cordova, estuve mirando documentación sobre las capacidades de esta tecnología, que tipo de flujo de trabajo utilizar, ventajas y desventajas. Además estuve mirando cuáles eran los requisitos necesarios para la instalación de Apache Cordova. Realicé una instalación de Apache Cordova de prueba en mi ordenador. Primero utilicé como herramienta de desarrollo Visual Studio. Con esta herramienta no coincidían las rutas, que se creaban de forma automática con el programa, con las rutas que se creaban sin utilizar Visual Studio como herramienta mediadora.

En el proceso de estudio también miré las tecnologías que se utilizaban para la parte visual de una aplicación. Encontramos que Ionic framework nos permitía hacer visualización en tiempo real de las vistas que se generan. También nos ofrecía una amplia lista de directivas AngularJS, y de estilos con los que podemos aproximar la interfaz de la aplicación a una interfaz lo más “nativa” posible. Esto permite mejorar la experiencia que tiene el usuario con la aplicación.

Teniendo en cuenta las notas tomadas del análisis de la aplicación, referentes a las vistas. Hice los prototipos de la aplicación. Dibujé cada una de las pantallas de la aplicación móvil fijándome en los elementos que tenía la aplicación inicial y cambié el estilo adaptándolo a unas vistas más modernas e intuitivas. En los dibujos fui indicando cuál era el orden en el que aparecía cada pantalla, teniendo en cuenta el flujo de la aplicación. Así mismo, en cada una de las vistas del prototipo fui indicando lo que representaba cada elemento. Ello nos permitió a la hora de implementar la funcionalidad tener en cuenta la forma en la que se interactuaba con cada elemento, y qué es lo que quería representar con ello.

Por otro lado, también hice los prototipos en papel para la parte de administrador de la aplicación. Inicialmente esta parte de la aplicación la habíamos pensado para dispositivos móviles. Sin embargo, como el objetivo era que el administrador pudiera editar, añadir y eliminar ejercicios y test, pensamos que desde un dispositivo móvil la capacidad de realizar estas acciones era limitada. Por eso el diseño se hizo para una aplicación web. En éste diseño el administrador podía ver datos estadísticos de la actividad de los usuarios de la aplicación móvil. Esta aplicación sigue un patrón de diseño visual parecido al de la aplicación móvil, donde prima la sencillez y la modernidad. También he considerado que es importante seguir la misma línea de estilo para que no se pierda la esencia de la aplicación inicial.

Una vez terminados los prototipos los puse en común con el resto del equipo y con el director del proyecto para corregir fallos y para hacer puesta en común de ideas sobre elementos que añadir

o quitar de la aplicación. Tras la reunión, añadí los cambios que se plantearon y volvimos a reunirnos.

El diseño de la base de datos la realizó María Teresa Calvo Ramón. Yo colaboré en el diseño inicial del diagrama entidad-relación. En esta tarea ayudé a definir las distintas entidades que tendría nuestra aplicación, así como a determinar los atributos y las relaciones que tendrían cada una de ellas.

La fase de creación de las vistas consiste en un proceso evolutivo con varias iteraciones. Inicialmente plasmé de manera sencilla las vistas que se habían creado a raíz del prototipo en papel aprobado por el cliente. Para esta primera versión de las vistas de la aplicación me ayudó Gian Marco Díaz, enseñándome ejemplos de login y menú principal que había hecho con Ionic Creator. De esta forma pude hacerme a la idea de cómo tenían que usarse los elementos que proporciona Ionic framework. Una vez tuve la implementación sencilla de las vistas, revisamos el trabajo con el cliente para que aprobara también los colores elegidos y la disposición que iban a tomar ahora los elementos. Para esta primera aproximación utilicé los elementos CSS proporcionados por Ionic framework. Estos son cabeceras, botones, navegación, tablas, disposición en rejilla “grid layout” para responsive design y creación de listas entre otros muchos.

Una vez revisadas las vistas con el cliente, recogimos información de mejoras que se podían hacer al aspecto de la aplicación. Tras esto, di comienzo a la implementación de unas vistas más personalizadas de la aplicación, cambiando los elementos por defecto que proporciona Ionic framework, y utilizando para ello CSS3. Tras revisarlo varias veces con el cliente conseguimos una aplicación que se ajustaba a sus requisitos.

También para la aplicación hice el logotipo utilizando la herramienta Adobe Illustrator. Primero realicé unos bocetos en papel. Teniendo en mente dichos bocetos cree varias opciones en Adobe Illustrator. Presenté los bocetos al cliente y finalmente, tras su aprobación, se realizó el logotipo actual de la aplicación.

La fase de desarrollo de la funcionalidad del proyecto la hemos hecho todos en conjunto. Yo me centré en la parte de “Ejercicios” de la aplicación. Primero traduje a JavaScript el código que se había generado con App Inventor 2 para implementar los algoritmos que se empleaban para el apartado de la calculadora de la aplicación del profesor César Benito. Estos algoritmos calculan la probabilidad de segregación de los genes a partir los valores introducidos en los campos de texto. Los uní al código de nuestra aplicación y realicé las pruebas necesarias para validar que el resultado obtenido al realizar el cálculo era correcto. Además, al igual que en la aplicación del profesor César Benito Jiménez, añadí indicadores visuales acerca de si los cálculos realizados eran correctos o no. Estos indicadores son dos iconos, uno verde con un tic y uno rojo con una cruz.

La fase de la elaboración de la memoria la hemos hecho entre todos. Yo he colaborado en la realización de las partes de “RESUMEN”, “ABSTRACT”, “Introducción: Antecedentes, Objetivos y Plan de Trabajo”, “Estado del arte y requisitos”, “Arquitectura de la aplicación”, “Manual de usuario”, “Manual de administrador” y “Trabajo futuro”. También he traducido los apartados “Introduction: History, Objectives and Planning” y “Conclusions”.

BIBLIOGRAFÍA

- accensit. (s.f.). *Comparativa de tecnologías para el desarrollo de aplicaciones Móviles*. Obtenido de accensit: <http://www.accensit.com/index.php/en/accensit-blog-en/150-mobile-platforms.html>
- Anastore. (2015). *Genética*. Recuperado el 11 de junio de 2016, de <https://play.google.com/store/apps/details?id=com.andromo.dev474745.app459246>
- Android. (2016). *Principios de diseño para Android*. Recuperado el 15 de junio de 2016, de Principios de diseño para Android: <https://developer.android.com/design/get-started/principles.html?hl=es#enchant-me>
- Android_Developers. (2016). *Android Developers*. Recuperado el 15 de junio de 2016, de Material Design para Android: <https://developer.android.com/design/material/index.html>
- Benito Jiménez, C. (1997). *360 problemas de Genética resueltos, paso a paso*. Síntesis.
- Benito Jiménez, C. (2015). *141 problemas de Genética resueltos, paso a paso*. Síntesis.
- Benito Jiménez, C., & Espino Nuño, F. (2013). *Genética, Conceptos esenciales*. Editorial Médica Panamericana.
- Codecademy. (2016). *Codecademy*. Obtenido de Codecademy: <https://www.codecademy.com/>
- Finelli, F. (28 de Junio de 2011). *Braintive*. Obtenido de 10 REGLAS HEURÍSTICAS DE USABILIDAD DE JAKOB NIELSEN: <http://www.braintive.com/10-reglas-heuristicas-de-usabilidad-de-jakob-nielsen/>
- Google_Play. (2016). *Google Play*. Recuperado el 11 de junio de 2016, de Top aplicaciones de Educación: https://play.google.com/store/apps/category/EDUCATION/collection/topselling_free?hl=es
- Inventor, M. A. (2012). *MIT App Inventor 2*. Retrieved from <http://ai2.appinventor.mit.edu/>
- Limited, A. (2015). *Genetics 4 Medics*. Recuperado el 11 de junio de 2016, de <https://play.google.com/store/apps/details?id=com.genetics4m>
- Limited, W. K. (2015). *Mendelian Genetics*. Recuperado el 11 de junio de 2016, de <https://play.google.com/store/apps/details?id=wwk.wikikids.com.mendeliangenetics>
- Limited, W. K. (2015). *Molecular Genetics*. Recuperado el 11 de junio de 2016, de <https://play.google.com/store/apps/details?id=wwk.wikikids.com.moleculargenetics>
- Los experimentos de mendel*. (2000). Obtenido de <http://pendientedemigracion.ucm.es/info/genetica/grupod/Mendel/mendel.htm#Inicio>
- Mendel, J. (1865). *Experimentos en hibridación de plantas (original en alemán "Versuche über Pflanzenhybriden")*.

- Mendel, J. G. (2016). *Leyes de Mendel*. Obtenido de <http://leyesdemendel.com/>
- Moodle. (14 de Marzo de 2016). *Moodle*, 3.0.3. Recuperado el 23 de Mayo de 2016, de Moodle: <https://moodle.org/>
- Nielsen, J. (1995). *Nielsen Norman Group*. Recuperado el 14 de Junio de 2016, de 10 Usability Heuristics for User Interface Design: <https://www.nngroup.com/articles/ten-usability-heuristics/>
- qode. (3 de Agosto de 2014). *¿Qué es una App Nativa?* Obtenido de qode: <http://qode.pro/blog/que-es-una-app-nativa/>
- robotics, A. (5 de Junio de 2013). *TIPOS DE APPS: NATIVAS, HÍBRIDAS Y WEB APPS*. Obtenido de appio: <http://appio.es/tipos-de-apps/>
- Sakai. (2014). Obtenido de <https://sakaiproject.org/>
- Sancho Thomas, P. (2010). *Nucleo: Un Sistema para el aprendizaje virtual colaborativo escenificado a través del rol multi-juego*. Obtenido de <http://eprints.ucm.es/9823/1/T31477.pdf>
- Science clarified*. (s.f.). Obtenido de Advameg, Inc: <http://www.scienceclarified.com/Ma-Mu/Mendelian-Laws-of-Inheritance.html>
- Udacity. (2011). *Udacity*. Recuperado el 11 de junio de 2016, de <https://www.udacity.com>
- UNED. (2016). *aLF*. Obtenido de Estés donde estás...: https://www.innova.uned.es/register/?return_url=%2fdotlrn%2fmiuned
- UNED. (2016). *UNED*. Obtenido de Estés donde estás...: https://portal.uned.es/portal/page?_pageid=93,571972,93_20535476&_dad=portal&_schema=PORTAL
- Wikipedia*. (2016). Obtenido de Wikipedia: <https://es.wikipedia.org/wiki/Gen%C3%A9tica>
- Wikipedia*. (2016). *Wikipedia*. Obtenido de Leyes de Mendel: https://es.wikipedia.org/wiki/Leyes_de_Mendel
- Wikipedia*. (2016). *Wikipedia*. Obtenido de Segregación de genes: https://es.wikipedia.org/wiki/Segregaci%C3%B3n_de_genes
- Wikipedia*. (2016). *Wikipedia*. Recuperado el 15 de junio de 2016, de Modelo–vista–controlador: <https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador>
- Zambrano, R. (2014). *Desarrollo de aplicaciones móviles ¿nativas, multiplataforma, HTML5 o híbridas?* Obtenido de mobidoo: <http://www.mobidoo.es/desarrollo-movil/desarrollo-de-aplicaciones-moviles-nativas-multiplataforma-html5-hibridas/>

GLOSARIO

API	<i>Application Programming Interface</i>
APK	<i>Application Programming Interface</i>
App	<i>Aplicación Móvil (Application)</i>
CSS	<i>Cascading Style Sheets</i>
GPS	<i>Global Positioning System</i>
HTML	<i>HyperText Markup Language</i>
JSON	<i>JavaScript Object Notation</i>
LCMS	<i>Learning Content Management Systems</i>
LMS	<i>Learning Management Systems</i>
MOOC	<i>Massive Open Online Course</i>
MVC	<i>Model View Controller</i>
PHP	<i>Hypertext Preprocessor</i>
QR	<i>Quick Response</i>
SASS	<i>Syntactically Awesome Stylesheets</i>
SQL	<i>Structured Query Language</i>
TFG	<i>Trabajo de Fin de Grado</i>
UAM	<i>Universidad Autónoma de Madrid</i>
UC3M	<i>Universidad Carlos III de Madrid</i>
UCM	<i>Universidad Complutense de Madrid</i>
UNED	<i>Universidad Nacional de Educación a Distancia</i>
USB	<i>Universal Serial Bus</i>