

# GenLab

App multiplataforma para cursos de genética



UNIVERSIDAD  
COMPLUTENSE  
MADRID

**Facultad de Informática**

**Trabajo Fin de Grado 2017/2018**

**Tutor:** Rubén Fuentes Fernández

(Dpto. de Ingeniería del Software e Inteligencia Artificial de la Facultad de Informática)

## **Autores**

Pablo Arranz Ropero (GIS)

Juan Alberto Camino Sáez (GIS)

Carlos López Martínez (GII)



# Índice

Resumen	4
Abstract	5
Introducción	6
Diseño de la aplicación	7
Arquitectura de la aplicación	7
Aplicación servidor	7
Aplicación cliente	10
Diseño de la base de datos	12
Herramientas, metodología y planificación	13
Herramientas	13
Metodologías	14
Planificación	15
Especificación de requisitos	19
Proceso de desarrollo	24
Desarrollo de la aplicación de administración (Servidor)	24
Desarrollo de la aplicación móvil (Cliente)	27
Manuales	29
Manual de usuario	29
Aplicación móvil	29
Aplicación servidor	36
Manuales del desarrollador	42
Manual de extensión de la aplicación móvil	42
Manual de extensión de la aplicación de administración	46
Reparto del trabajo	49
Conclusiones y trabajo futuro	50
Conclusions and future work	52
Resultados de aprendizaje	54
Bibliografía	55
Glosario	56

# Resumen

El presente trabajo ha tratado de crear una aplicación para el desarrollo y despliegue de ejercicios relacionados con el campo de la genética. Los ejercicios han sido anteriormente desarrollados por César Benito Jiménez (profesor de la Facultad de Biología de la Universidad Complutense de Madrid) mediante MIT App Inventor. Esta solución presentaba varios problemas. Añadir nuevos ejercicios suponía una dificultad para hacerlos llegar a los alumnos, qué debían descargar de nuevo la aplicación. Además, estas aplicaciones eran demasiado pesadas, al contener todas las imágenes y textos necesarios. Para integrar todas las partes de estos ejercicios ha sido necesario estudiar su lógica y generalizarla, de modo que se facilita la creación y gestión de estas. Para facilitar estas tareas se ha creado una aplicación con una arquitectura cliente-servidor. La parte de servidor está basada en Java Spring Framework y permite gestionar los distintos contenidos que se muestran al usuario final. Gracias a la parte del servidor se consigue que el profesor no necesite conocimientos de programación para poder gestionar esta aplicación, solo de genética y de navegación web. La parte de cliente es una aplicación móvil que integra las distintas funcionalidades gestionadas por el servidor para que el alumno pueda interactuar con ellas. Dicha aplicación móvil se ha construido con distintas tecnologías web como HTML5, CSS3 y JavaScript. Todo ello se ha encapsulado con Cordova para permitir que fuese una aplicación multiplataforma (Android e IOs), y así poder permitir el uso de la aplicación al mayor número de alumnos posibles.

**Palabras clave:** Genética, Multiplataforma, E-learning, Aplicación móvil, Arquitectura cliente-servidor.

# Abstract

This paper has attempted to create an application for the development and deployment of exercises related to the field of genetics. The exercises had been previously developed by César Benito Jiménez (professor at the School of Biology of the Complutense University of Madrid) using MIT App Inventor, which was not an acceptable solution, since the addition of new exercises was a difficulty to make them reach the students and also these applications were too heavy as they contained all the necessary images and texts. In order to integrate all the parts of these exercises, it has been necessary to study their logic and to generalize them, so as to facilitate their creation and management. To facilitate this, an application has been created offering a server side based on the Java Spring Framework that allows managing the different contents that are shown to the end user. Thanks to the server part, the teacher does not need any knowledge of programming to be able to manage the contents of this application, only of genetics and web browsing. In addition, the client part, which is the mobile application itself, has also been created, bringing together the different functionalities managed by the server so that the student can interact with them. This mobile application has been built with different web technologies such as HTML5, CSS3 and JavaScript, and is all encapsulated with Cordova to allow it to be a multi-platform application (Android and IOs) and thus allow the use of the application by the greatest number of students possible.

**Keywords:** Genetics, Multi-platform, E-learning, Mobile application, Server-client architecture.

# Introducción

La importancia de las aplicaciones móviles es innegable en la actualidad. Su impacto ha llegado al campo de la educación, donde permiten distribuir y trabajar con contenidos interactivos complejos de forma fácil, al tiempo que suelen tener un impacto motivador sobre los alumnos. Dentro de la Facultad de Biología de la Universidad Complutense de Madrid decidieron realizar una serie de aplicaciones para permitir a sus alumnos mejorar en la materia de Genética. El uso de las aplicaciones mostró ciertos problemas inherentes a su diseño, que debían ser abordados para facilitar la extensión de su uso. De aquí surge el propósito de este trabajo: desarrollar una aplicación que aborde las limitaciones de las presentes aplicaciones. Ello implica facilitar el desarrollo y distribución de nuevos ejercicios, reduciendo la necesidad de conocimientos de programación para ello. Además, se persigue que el profesor tenga mayor capacidad para monitorizar la evolución de sus alumnos a través de los ejercicios. También, que los ejercicios estén disponibles para el mayor número posible de alumnos sin necesidad de programar versiones alternativas de los ejercicios para las diferentes plataformas. Asimismo, se ha realizado de tal forma que gracias a los manuales recogidos en este documento se pueda extender la aplicación con más ejercicios.

El resto de la memoria se organiza como sigue:

En primer lugar, se exponen los requisitos hallados durante la fase de especificación de la aplicación (Véase la sección Especificación de requisitos).

En segundo lugar, se presenta todo lo relacionado con el proceso de planificación y desarrollo del proyecto, donde se expone la metodología utilizada a la hora de realizar el proyecto (Véase la sección Metodologías).

Seguidamente se incluye el conjunto de manuales necesarios para comprender como usar la aplicación, así como otros manuales para facilitar la extensión tanto de la plataforma web como de la aplicación móvil (Véase la sección Manuales).

Finalmente, se muestra lo aprendido (Véase la sección Resultados de aprendizaje) a partir de la realización del trabajo y las conclusiones asociadas a estos resultados (Véase la sección Conclusiones).

# Diseño de la aplicación

## Arquitectura de la aplicación

En el siguiente apartado se muestra la arquitectura principal del proyecto.

Este proyecto sigue una arquitectura cliente-servidor. Esta arquitectura está compuesta por un cliente que realiza las peticiones, y un servidor que les da respuesta.

A continuación, presentamos la aplicación cliente y la aplicación servidor junto a las distintas características que les rodean y al mecanismo utilizado para la correcta comunicación entre ellas.

Además de esto, también se trata el diseño de la base de datos utilizada para el almacenamiento de datos.

### Arquitectura de la aplicación servidor

La aplicación servidor permite a los usuarios administradores puedan gestionar los contenidos que se mostrarán a los usuarios finales cuando accedan a la aplicación móvil. Es una aplicación web que permite crear, modificar y eliminar teoría, problemas, libros y tests, así como ver los resultados de los tests resueltos. También sirve de API para que la aplicación cliente haga peticiones. Está desarrollada siguiendo una arquitectura multicapa<sup>[8]</sup>, y se distribuye a grandes rasgos en los siguientes paquetes:

- **Controllers:** Contiene todos los manejadores de rutas de nuestra aplicación web y de la API REST. Forma parte de la capa de presentación.
- **Models:** Contiene todas las entidades y objetos de datos de la aplicación. Forma parte de las capas de integración y presentación.
- **Repositories:** Contiene los repositorios de la aplicación, es decir, son las clases que accederán a la base de datos cuando sea necesario. Forma parte de la capa de integración en nuestra arquitectura.
- **Services:** Contiene los servicios de aplicación donde se realizan las operaciones necesarias sobre los datos que llegan desde el controlador. Hacen de intermediarios entre la capa de presentación y la de integración. Forma parte de la capa de negocio en nuestra arquitectura.
- **Config, Interceptor y Utils:** Estos paquetes contienen utilidades necesarias para la configuración de las peticiones HTTP.

- **Resources:** Contiene las plantillas creadas en HTML (Thymeleaf<sup>[1]</sup>). También contiene los archivos CSS y JavaScript necesarios para la construcción de la vista. parte de la capa de presentación en nuestra arquitectura.

```

main
+---java
|   \---com
|       \---genlab
|           \---serverapplication
|               | ServerApplication.java
|               +---config
|               | ...
|               +---controllers
|               | ...
|               +---interceptor
|               | ...
|               +---models
|               | ...
|               +---repositories
|               | ...
|               +---services
|                   +---bookService
|                   | ...
|                   +---ctservice
|                   | +---Epistasia
|                   | | ...
|                   | +---Linkage
|                   | | ...
|                   | +---Onelocus
|                   | | ...
|                   | +---Polyhybrid
|                   | | ...
|                   | \---Twoloci
|                   | ...
|                   +---problemsService
|                   | ...
|                   +---sectionService
|                   | ...
|                   +---testsService
|                   | ...
|                   +---theoryService
|                   | ...
|                   \---userService
|                   | ...
|                   \---utils
|                   | ...
|                   \---resources
|                       | ...
|                       +---static
|                       | +---css
|                       | | ...
|                       | +---img
|                       | | ...
|                       | \---js
|                       | | ...
|                       | \---templates
|                       | | ...
|                       | +---fragments
|                       | | ...
|                       | \---layouts
|                       | | ...

```

Imagen 1. Estructura de carpetas de la aplicación servidor.



Uno de los patrones más importantes que hemos aplicado a nuestra aplicación es el patrón MVC<sup>[8]</sup>. Este patrón permite una comunicación modularizada entre las distintas partes de la aplicación:

1. El modelo, que estará formado por los paquetes `models`, `repositories` y `services`. Esta será la parte en la que se implementará toda la lógica de negocio. El modelo se comunicará con los controladores y nunca de manera directa con la vista.
2. El controlador, que en nuestro caso serán varios controladores, uno por cada “módulo” de nuestra aplicación. Estos controladores serán manejadores de rutas HTTP y se encargarán de recibir las interacciones con la vista (es decir, las peticiones HTTP) y de llamar a las partes necesarias del modelo para procesar estas interacciones. Está formado por el paquete `Controllers`.
3. La vista. Está formada por todos los ficheros HTML, CSS y Javascript que podemos encontrar en el paquete `resources`. Forma toda la interfaz gráfica de usuario y se comunicará con los controladores mediante peticiones HTTP ya sea mediante botones o enlaces, o mediante eventos recogidos con JavaScript.

Otros dos patrones muy utilizados en todo el diseño de la aplicación han sido el patrón builder<sup>[9]</sup> (para permitir la creación de objetos a partir de un objeto base) en todos los objetos del modelo, y el patrón de inyección de dependencias<sup>[10]</sup> (para permitir el uso de objetos por parte de una clase sin necesidad de que la misma los cree), cuyo uso facilita el framework elegido (i.e.: Spring Framework).

El flujo principal de la aplicación servidor depende de las distintas acciones que pueda ejecutar el administrador. No obstante, todas siguen el mismo esquema.

El flujo de trabajo siempre empieza cuando el usuario realiza la acción de iniciar sesión desencadenando una consulta para comprobar si los datos de acceso son correctos y así permitirle acceder a la aplicación.

Una vez dentro, puede elegir entre las diferentes secciones. Dentro de cada sección podrá realizar cada uno de los diferentes casos de uso expuestos en la imagen 7.

Al elegir cualquiera de los casos de uso se realiza una petición a la base de datos para obtener la información necesaria con relación a la sección de la aplicación elegida y al apartado seleccionado.

## Arquitectura de la aplicación cliente

Es una aplicación móvil disponible para todos aquellos usuarios que quieran usar la aplicación. Permite la visualización de teoría, problemas, libros recomendados, herramientas de cálculo y tests. También permite el uso de las herramientas de cálculo y la resolución de los tests mediante peticiones AJAX a la aplicación servidor, cuando éstas sean requeridas por el usuario.

La aplicación contiene la estructura básica de carpetas que se genera al crear un proyecto con Apache Cordova.

La carpeta *www* se encuentra dentro de esta jerarquía de ficheros y contiene la estructura de la aplicación, como se muestra en la imagen 2.

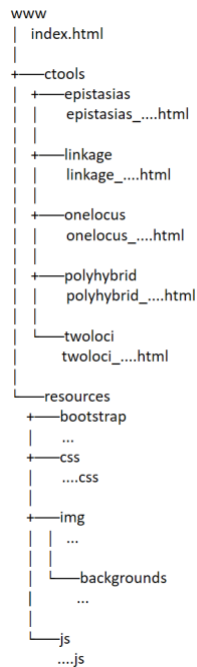


Imagen 2. Estructura principal de la carpeta /www

En cuanto a la arquitectura, se basa principalmente en una programación orientada a eventos, usando el modelo SPA.

El modelo SPA permite la carga de la página y de sus recursos en una única vez al iniciar la aplicación. De este modo reducimos los tiempos de carga para favorecer la interacción con el usuario. A medida que el usuario navega por las distintas secciones, se muestran u ocultan los elementos correspondientes.

Además de esto hay incluido un menú desplegable que contiene las distintas secciones de la aplicación (*One Locus, Two Loci, ...*) para poder navegar entre ellas. Una vez se ha elegido la sección que se quiere, sería pulsar sobre el apartado deseado.

Tras pulsar ese apartado se mostrará directamente la información (*Recommended Books, Problems y Theory*) o un listado de contenidos (*Calculation Tools y Tests*).

Si lo que se ha pulsado se muestra en forma de listado lo único que habrá que hacer es pulsar sobre el contenido que se desea consultar.

Si el contenido es una *Calculation Tool*, se muestran una serie de campos a rellenar, en dichos campos es donde el usuario final debe introducir los cálculos obtenidos durante el experimento en cuestión, una vez se hayan introducido, dándole al botón de calcular, la aplicación móvil hará una petición a la aplicación servidor para que calcule el resultado con los datos introducidos. Después de haber calculado el resultado será enviado al dispositivo móvil para su posterior presentación.

Si el contenido es un *Test*, se muestran las distintas preguntas a contestar. Cada vez que el usuario pulsa una respuesta, ya sea correcta o incorrecta se envía a la aplicación servidor para introducirla en la base de datos y así después se pueda mostrar en la parte de feedback, permitiendo que el usuario administrador observe dónde falla o acierta más la gente.

No obstante, para poder avanzar entre las distintas secciones es necesario haber completado antes los tests de las anteriores secciones de la aplicación.

En la imagen 3 mostramos de forma básica y esquemática el flujo principal de la aplicación móvil.

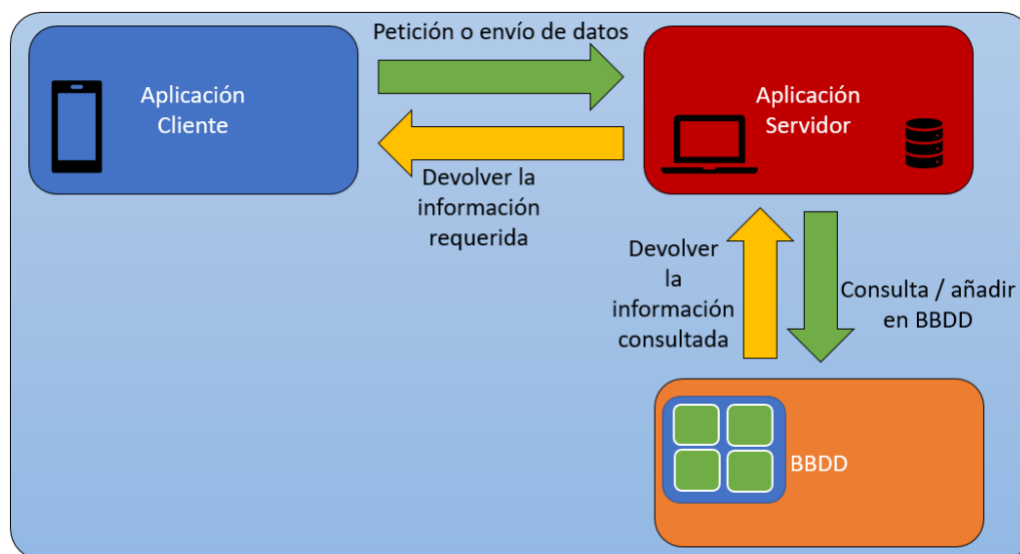


Imagen 3. Esquema del flujo de la aplicación móvil.

# Diseño de la base de datos

El diagrama mostrado en la imagen 4 es el esquema relacional de la base de datos en el que se representan únicamente las entidades que se almacenan en la base de datos.

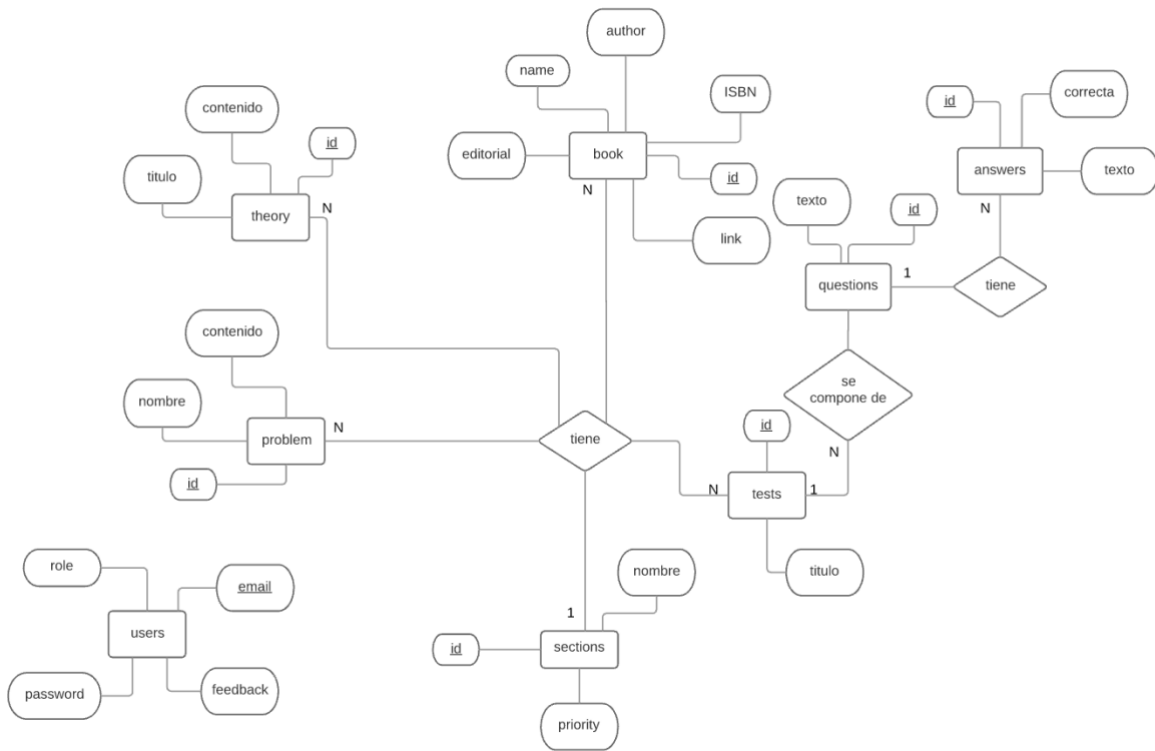


Imagen 4. Diagrama Entidad-Relación de la Base de Datos

# Herramientas, metodología y planificación

## Herramientas

Durante el desarrollo de este proyecto hemos utilizado distintas herramientas, tanto en la parte de cliente como en la parte de servidor. A continuación, se exponen los distintos instrumentos utilizados y su propósito.

En el lado cliente utilizamos:

- Sublime Text 3 y Visual Studio Code para el desarrollo del código, que será desarrollado utilizando HTML5, CSS3 y Javascript exclusivamente.
- Apache Cordova. Hemos escogido Cordova porque permite diseñar una aplicación móvil mediante tecnologías web, por lo que se tiene acceso a amplios recursos que HTML, CSS y JavaScript permiten (como el diseño Responsive, para permitir la correcta visualización independientemente del tamaño de la pantalla, fundamental para una aplicación móvil, donde existen múltiples tamaños del dispositivo) y, además, porque permite generar a partir del código los distintos ejecutables para cada sistema operativo (como Android o IOS).
- Un dispositivo móvil tanto Android como iOS para poder probar y depurar la aplicación.
- Las herramientas de desarrollador incorporadas en Google Chrome para poder depurar la aplicación de manera rápida.

En el lado servidor utilizamos:

- Spring Tool Suite para el desarrollo y depuración del código, que será desarrollado utilizando Java junto con Spring Framework (para la aplicación del patrón MVC) y el framework JPA (para el acceso a la base de datos) entre otros. Se escoge esta herramienta debido a su especialización en el desarrollo de la parte del servidor.
- Thymeleaf <sup>[1]</sup> para la creación de plantillas web, ya que permite generar código de forma dinámica a partir de los distintos objetos Java.
- Maven para la gestión de dependencias y construcción del proyecto.
- MySQL como base de datos donde alojar todos los datos de la aplicación.
- La terminal del sistema para acceder de manera remota a los dos servidores usados tanto como a la base de datos.

Para realizar el despliegue de la aplicación hemos usado dos servidores:

1. De manera temporal para realizar pruebas hasta tener acceso al servidor final, un servidor alojado en una Raspberry Pi 3 Model B.
2. De manera última y definitiva un servidor proporcionado por la Facultad de Informática de la Universidad Complutense de Madrid con Ubuntu Server 16.04.3 con el stack LAMP previamente instalado. Este servidor está alojado en la URL ingenias.fdi.ucm.es y se nos proporcionó acceso a los puertos en el rango 60070-60073.

A parte de las herramientas ya nombradas, hemos utilizado GitHub como repositorio para realizar el control de versiones y almacenar la aplicación durante el desarrollo. Se ha usado tanto para el desarrollo de la aplicación cliente como de la del servidor, manteniéndolas en repositorios separados.

## Metodología

A la hora de gestionar este proyecto hemos utilizado una metodología ágil basada en Scrum<sup>[11]</sup>.

Scrum es una metodología ágil en la que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener así el mejor resultado posible de un proyecto. Sobre esta metodología hemos realizado pequeños cambios debido a disponibilidades de horarios durante el desarrollo del producto y a fechas límite.

Esta metodología divide el tiempo total de desarrollo en *sprints*, que son fragmentos de duración variable, pero en torno a dos semanas, en los que se desarrollan funcionalidades acordadas.

Consideramos que esta metodología es la acertada ya que no tenemos unos requisitos que consideremos estables al comienzo del desarrollo y ante la posibilidad de estos de cambiar, Scrum puede ser de ayuda.

Otro motivo para elegir esta metodología es el hecho de poder presentar prototipos funcionales cada poco tiempo a nuestro tutor para de tal modo tener una opinión sobre cómo avanza el trabajo.

Además de todo lo anterior se ha tenido una serie de restricciones que nos limitaron y que nos ayudaron a decidirnos por Scrum como metodología a seguir. Estas restricciones fueron: las distintas fechas de entrega marcadas debido a las cuales tuvimos que estimar qué cantidad de tareas debían incluirse en cada *sprint*, la realización de reuniones tanto con el tutor del TFG (Rubén Fuentes Fernández) como con el profesor desarrollador (César Benito Jiménez), el estudio de las aplicaciones originales creadas con MIT App Inventor y su posterior traducción a tecnologías Java.

## Planificación

La planificación de este proyecto se ha realizado tomando en cuenta el tiempo disponible para la realización tanto del proyecto en sí como de la memoria requerida al final de este. Para ello se han marcado una serie de hitos a lo largo del calendario académico. Para el cumplimiento de estos hitos nos hemos ayudado de la aplicación *Trello* que nos permitió gestionar y estipular los distintos sprints del proceso, así como las tareas requeridas en cada uno de esos sprints.

En esta herramienta también se incluyó la recogida de la especificación de requisitos, y la realización de mockups como tareas a realizar en el proyecto.

En la primera etapa del trabajo determinamos los requisitos necesarios que deberían cumplir las distintas aplicaciones, para ello realizamos una primera reunión donde se determinaron tanto dichos requisitos como los objetivos principales que debíamos alcanzar para sacar el proyecto adelante.

En las siguientes etapas realizamos reuniones periódicas al final de cada sprint entre los distintos integrantes del trabajo para abordar los distintos aspectos de la aplicación, y también para mostrar los problemas surgidos y cómo resolverlos.

Durante estas reuniones hacíamos una puesta en común de lo que llevábamos realizado hasta el momento y qué íbamos a hacer en el siguiente sprint, así como la distribución del trabajo. Estas reuniones eran los viernes por cuestiones de disponibilidad de los distintos integrantes.

Además, hubo reuniones con Rubén Fuentes Fernández (tutor del trabajo) y en las cuales se mostraron los hitos realizados y los problemas surgidos durante el desarrollo de estos. Estas reuniones nos sirvieron de feedback para mejorar y extender la aplicación con relación a distintas funcionalidades.

No obstante, también realizamos reuniones con el César Benito Jiménez (profesor desarrollador de las aplicaciones originales) en la Facultad de Biología para obtener también feedback suyo, sobre todo de la aplicación de la parte del servidor ya que al fin y al cabo él era el usuario final.

Finalmente, tanto al final de cada hito como al final de la planificación revisamos que los requisitos especificados al principio se fueran cumpliendo dando lugar al correcto desarrollo del proyecto.

En la imagen 5 podemos observar un ejemplo del panel que contiene nuestra planificación en *Trello* incluyendo las fechas de los sprints.

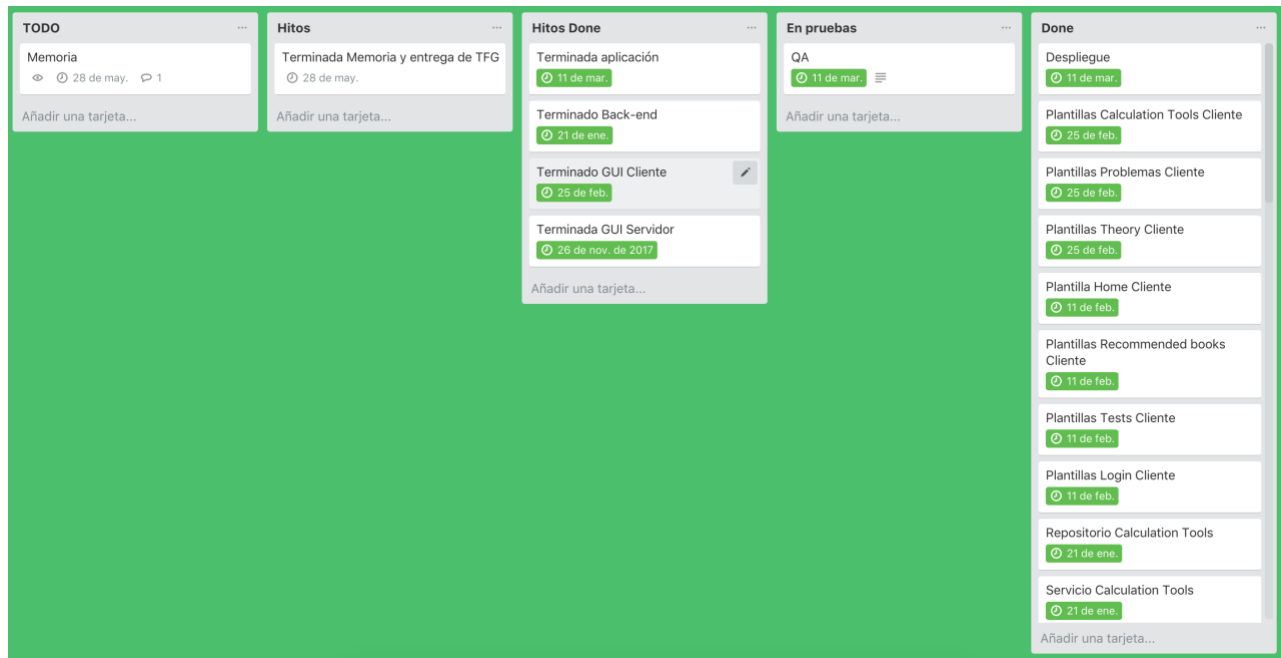


Imagen 5. Panel que contiene los sprints en Trello.

La planificación del desarrollo de nuestro proyecto se adjunta en un diagrama de Gantt junto con esta memoria, que por temas de tamaño y resolución no consideramos adecuado adjuntar como imagen aquí.

En cuanto a la ejecución de esta planificación resultó de la siguiente manera:

01/10/17 – 15/10/17:

Comienzan las reuniones con el tutor del TFG para aclarar conceptos y con César (el usuario final) para la extracción de requisitos. Realizamos la configuración inicial del servidor.

15/10/17 – 29/10/17:

Creamos y configuramos el repositorio donde iremos almacenando las distintas versiones del código de nuestro proyecto. Creamos distintos mockups y una especificación de requisitos en forma de borrador que van adjuntos junto con esta memoria y se los mostramos al cliente y usuario final, César Benito Jiménez, para que nos aclare qué tipo de producto es el que quiere.

Teniendo en cuenta su feedback rehacemos ciertos requisitos para que finalmente sean los mostrados en la sección Especificación de requisitos de esta memoria.



29/10/17 – 11/11/17:

Creamos la estructura base del proyecto con el esquema de paquetes y carpetas que siga nuestra arquitectura y comenzamos a desarrollar las plantillas básicas (sin funcionalidad por el momento) de `login`, `home` y `tests`.

11/11/17 – 26/11/17:

Seguimos con la creación de plantillas. En este sprint nos encargamos de maquetar y diseñar las plantillas de `problems`, `recommended books`, `calculation tools` y `theory`. Un problema que surgió durante este sprint fue el gran volumen de plantillas que era necesario crear para las Calculation Tools, por lo que la creación de estas se fue arrastrando durante los siguientes sprints.

26/11/17 – 03/12/17:

Comenzamos a desarrollar la capa de negocio de nuestra aplicación, creando pares de servicio de aplicación y repositorio por cada módulo. En este sprint se desarrollan el servicio de aplicación y el repositorio del módulo `theory`.

03/12/17 – 10/12/17:

Durante este sprint se desarrollan el servicio de aplicación y el repositorio del módulo `tests`.

10/12/17 – 17/12/17:

Durante este sprint se desarrollan el servicio de aplicación y el repositorio del módulo `problems`. El desarrollo de este módulo fue bastante sencillo debido a que, funcional y visualmente los módulos `problems` y `theory` son similares.

17/12/17 – 14/01/18:

En este sprint, que alargamos por cuestiones de descanso y estudio durante las vacaciones de navidad, desarrollamos el servicio de aplicación y el repositorio del módulo `recommended books`. También se implementó la lógica del login con Spring Security.

14/01/18 – 21/01/18:

Se terminan de crear las plantillas de las Calculation Tools y se crea el servicio de aplicación que realiza los cálculos de estas. Al ser esta una tarea complicada, que necesitaba de la extracción de las fórmulas de las anteriores aplicaciones, se alargó durante los siguientes sprints (que se dedicaron al desarrollo de la interfaz de la aplicación móvil).

21/01/18 – 11/02/18:

Se crea la interfaz de la aplicación móvil (y su comunicación con la API ofrecida por el servidor) del login, del home y de los módulos tests, recommended books.

11/02/18 – 25/02/18:

Se desarrollan las interfaces de los módulos theory, problems y de las calculation tools.

El despliegue de la aplicación y las pruebas fueron realizándose durante el desarrollo de la aplicación, pero con fecha límite a 11/03/18.

En nuestra planificación también marcamos diferentes hitos:

- |   |          |
|---|----------|
| 1. Finalización de la parte front-end del servidor →                            | 26/11/17 |
| 2. Finalización de la parte back-end del servidor →                             | 21/01/18 |
| 3. Finalización de la interfaz de la aplicación móvil →                         | 25/02/18 |
| 4. Finalización de todas las partes de la aplicación →                          | 11/03/18 |
| 5. Finalización y entrega del producto final, incluyendo memoria y aplicación → | 28/05/18 |

# Especificación de requisitos

El diagrama mostrado en la imagen 6 es el modelo de dominio, en el que se representan todas las entidades que participan en la aplicación.

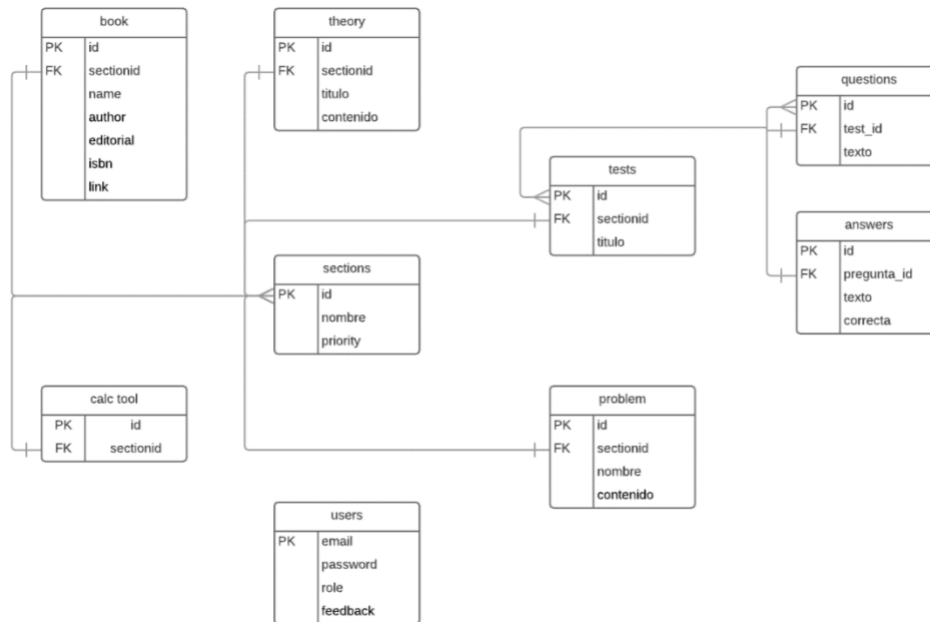


Imagen 6. Modelo de dominio.

En la parte del **servidor** el usuario podrá realizar las siguientes funciones:

**Login:** Se deberá acceder con un usuario y contraseña válidos para poder entrar a la aplicación.

**Gestionar sección:** Muestra los apartados disponibles dentro de cada sección de la aplicación. Estos apartados serán:

1. Calculation Tools: Herramientas que permiten al usuario introducir datos obtenidos en una situación para ver si el resultado es correcto acorde a un tipo de teoría.
  - a. *Ver y usar una Calculation Tool.*
2. Teoría: Teoría que abarca una sección en general.
  - a. *Añadir teoría*: especificar el título de la teoría e incluir texto, imágenes, ...
  - b. *Editar teoría*: editar el título o el contenido de la teoría.
  - c. *Borrar teoría*: borrar la teoría de manera irreversible.

3. Problemas: Enunciados de problemas disponibles para los alumnos (no se resuelven en la aplicación, son solo los enunciados).
  - a. *Añadir problema*: especificar un título y un contenido (texto, imágenes, ...) con el enunciado del problema.
  - b. *Editar problema*: editar el contenido de un problema.
  - c. *Borrar problema*: borrar un problema de manera irreversible.
4. Test: Preguntas tipo test disponibles para la resolución por parte de los alumnos.
  - a. *Añadir test*: añadir un nuevo test adjudicándole un nombre.
  - b. *Añadir cuestión*: añadir una nueva cuestión a un test determinado.
  - c. *Añadir respuesta*: añadir una nueva respuesta dentro de una cuestión perteneciente a un test.
  - d. *Editar test*: editar el nombre de un test, así como sus cuestiones o respuestas.
  - e. *Borrar test*: borrar un test junto a todo su contenido.
  - f. *Borrar cuestión*: borrar una cuestión determinada junto a todas sus respuestas asociadas.
  - g. *Borrar respuesta*: borrar una respuesta determinada.
5. Libros recomendados:
  - a. *Añadir libro recomendado*: añadir un libro completando los campos necesarios para ello.
  - b. *Editar libro recomendado*: editar la información de un libro.
  - c. *Borrar libro recomendado*: borrar un libro recomendado de manera irreversible.

**Ver resultados de los alumnos:** Muestra como feedback las distintas respuestas a los tests que han ido contestando los usuarios de la aplicación móvil.

**Establecer prioridad de las secciones:** Permite asignar una prioridad determinada a las distintas secciones de la aplicación. Esta prioridad será la que se tome en cuenta una vez se arranque la aplicación móvil.

Las funciones que puede realizar el usuario administrador en la aplicación servidor se muestran de una manera más visual en el diagrama de casos de uso de la imagen 7.

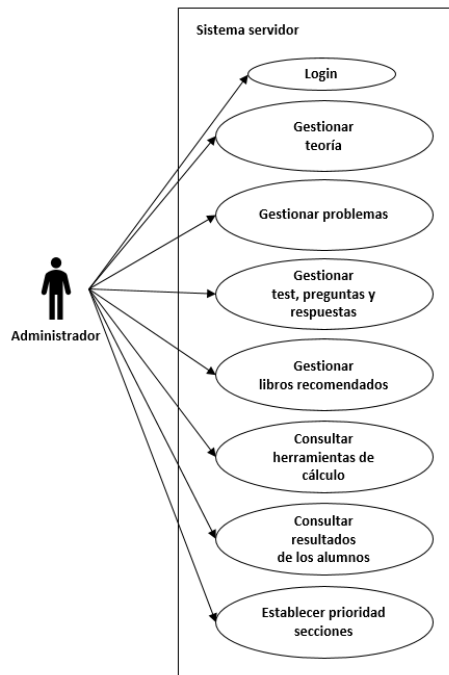


Imagen 7. Diagrama de casos de uso de la aplicación servidor

Algunos de estos casos de uso implican cambios permanentes en la base de datos y otros lo único que hacen es consultar información.

Los casos de uso que implicarán cambios permanentes en base de datos son: *Gestionar teoría*, *Gestionar problemas*, *Gestionar test/preguntas/respuestas*, *Gestionar libros recomendados* y *establecer la prioridad de las secciones de la aplicación*.

Sin embargo, los casos de uso de *login*, *consulta de resultado de los alumnos*, *consulta de las herramientas de cálculo* o *logout* no realizarán ningún cambio en la base de datos.

En la parte **cliente** el usuario podrá realizar las siguientes acciones:

**Login:** a través de la inserción de un nombre de usuario para poder registrar el feedback en el servidor y permitir monitorizar el avance en la aplicación gracias a la resolución de los distintos tests.

**Elegir sección:** dentro de la cual se incluyen las distintas subsecciones disponibles.

**Elegir subsección:** Se incluyen las mismas subsecciones que en la parte del servidor, pudiendo consultar la siguiente información:

1. *Calculation Tool* y usarla.
2. *Teoría*.
3. *Problemas*.
4. *Tests*, así como poder contestarlos viendo si las respuestas han sido correctas o erróneas.
5. *Libros recomendados* para una sección concreta.

Estas mismas acciones quedan recogidas de forma esquemática en el siguiente diagrama de casos de uso de la aplicación móvil:

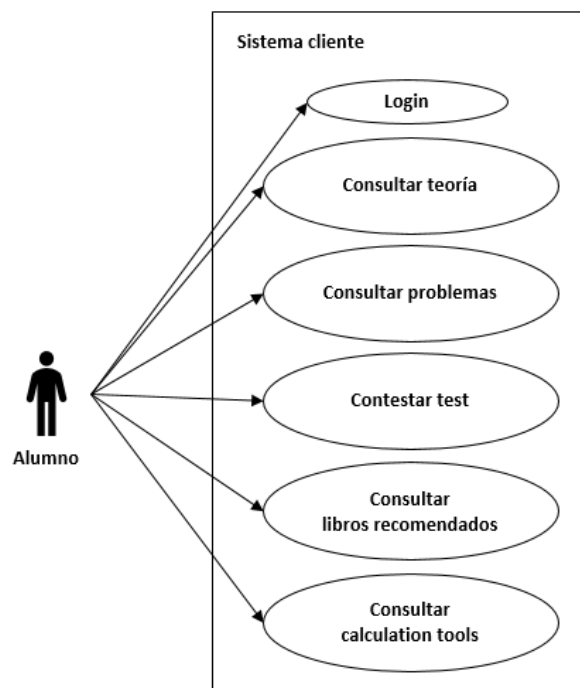


Imagen 8. Diagrama de casos de uso de la aplicación móvil.

El caso de uso que incorporará información a la base de datos es el de *Contestar Test* ya que añade a la base de datos la información de los resultados obtenidos en dicho test, para que el usuario administrador pueda obtener los resultados de los usuarios.

El resto de los casos de uso obtendrán los diferentes tipos de información directamente de la API en el lado servidor y una vez se haya obtenido esa información se presentará en el dispositivo móvil con diferentes formatos.

Además, la parte del cliente tenía que ser una aplicación para dispositivos móviles. Por ello, era necesario que la aplicación funcionase para cualquier sistema operativo del dispositivo o, al menos, aquellos que abarcasen un alto porcentaje del mercado. También debía adaptarse a las múltiples dimensiones de pantalla de estos dispositivos, y permitir su correcta visualización independientemente del dispositivo usado.

# Proceso de desarrollo

En este apartado del documento se explica el proceso que se ha seguido a lo largo de todo el año para desarrollar la aplicación, así como los problemas que se han ido encontrando durante el desarrollo, las distintas opciones que surgieron para su realización y la posterior solución utilizada.

## Desarrollo de la aplicación de administración (Servidor)

### Planificación del desarrollo

El desarrollo de la aplicación de administración comenzó tras haber realizado una planificación y entrega previa de los requisitos que debía soportar la aplicación, así como la arquitectura, el lenguaje y el framework que íbamos a utilizar durante el desarrollo de la misma. Además, también realizamos una serie de mockups interactivos que nos sirvieron de guía a la hora del desarrollo de la interfaz de la aplicación.

No obstante, también se pensó en la interacción cliente-servidor y qué tecnología sería la utilizada para llevar a cabo la aplicación.

Por otro lado, establecimos los distintos modelos de datos que serían utilizados en la aplicación, estos fueron realizados de forma básica debido a que durante el desarrollo se iteraría sobre estos modelos ya que podían surgir nuevas entidades.

### Proceso de desarrollo

Para desarrollar esta aplicación hemos utilizado la metodología de la que hablamos en la sección Metodologías de esta memoria.

Tras haber planificado todo lo anterior, lo primero que hicimos fue crear una aplicación mínima funcional sobre la que ir iterando. Esta aplicación consistía en las plantillas iniciales y los mínimos manejadores de ruta en los controladores para mostrar estas plantillas (vacías inicialmente) en las distintas URLs que estarían disponibles.

A continuación, maquetamos con ayuda de Bootstrap el *layout* de la página (ver imagen 9) por que al ser una cosa que se muestra en todas las páginas lo vimos prioritario.

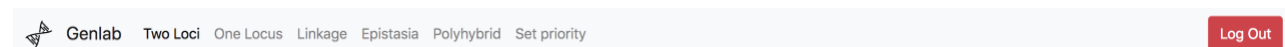


Imagen 9. Layout de la aplicación.



Una vez teníamos lista nuestra aplicación en su forma mínima pasamos a diseñar el esquema definitivo de los datos, que podemos representar en dos diagramas (están incluidos en las secciones de *Especificación de requisitos* y *Diseño de la base de datos* de esta memoria). Estos diagramas fueron plasmados en Java gracias a las anotaciones JPA en las distintas clases objeto del modelo, ya que de esta manera podíamos encapsular los datos de la base de datos. No obstante, no todo fue guardado en base de datos, las *Calculation Tools* están almacenadas en la propia aplicación.

Tras todo lo anterior creamos los repositorios extendiéndolos a la clase `CrudRepository`, de esta manera proporcionamos a la aplicación de una serie de métodos con las operaciones básicas de consulta y manejo de la base de datos (e.g. `save`, `delete`, `findAll`, etc...), pudiendo nosotros crear más métodos aparte (por ejemplo, `findBySectionId`).

Después, desarrollamos los servicios que permitían conectar el controlador con los repositorios, haciendo así las operaciones necesarias para implementar las reglas de negocio, menos en el caso de las *Calculation Tools* ya que éstas no tienen ningún repositorio asociado.

Además de todo lo anteriormente explicado, cabe destacar ciertos aspectos que han sido desarrollados con Javascript:

1. El comportamiento dinámico de la página de edición de tests, permitiendo añadir/eliminar preguntas/respuestas sin recargar la misma.
2. La carga de fragmentos en el apartado Calculation Tools. Debido a que cada Calculation Tool es significativamente diferente de las demás, para hacer la aplicación de manera modular y mantenible, hemos desarrollado las Calculation Tools en forma de fragmento en un html diferente, de manera que gracias a Javascript se carga el fragmento pedido dependiendo de la Calculation Tool.
3. El uso de una herramienta WYSIWYG en los apartados de problemas y teoría. Estas herramientas nos permiten añadir texto con imágenes y diferentes formatos que se guardaran en la base de datos de manera que será posible mostrárselo de esa manera a los estudiantes.

En este momento y una vez finalizada la parte del back-end, presentamos la aplicación a nuestro tutor, dándonos a conocer su opinión y posibles avances o modificaciones que debíamos realizar. Estas mejoras fueron:

1. La obtención de los resultados en los tests hechos por los alumnos por parte del profesor acerca de los resultados de sus alumnos en los tests.
2. La posibilidad de otorgar cierta prioridad a una sección respecto de las otras, de manera que hasta que no se completan todos los tests de una sección no se pueden acceder a las secciones

con una menor prioridad. Esto se implementó usando Javascript para hacerlo de una manera más fácil (Drag and Drop).

Por último, añadimos la capa de seguridad usando Spring Security, proporcionando un mecanismo para evitar el intrusismo dentro de la aplicación.

## Seguimiento del desarrollo

Antes de empezar a desarrollar algún aspecto de la aplicación tuvimos que revisar y aprender la documentación de las distintas tecnologías que íbamos a utilizar durante el desarrollo, ya que algunas tecnologías eran nuevas para los integrantes del proyecto.

Durante el desarrollo de la aplicación de administración se solucionaron fallos de programación y gestión, algunos fueron más importantes que otros, pero se fueron solucionando en la medida de lo posible. Uno de esos fallos fue la gestión de imágenes en la herramienta WYSIWYG, debido a que a la hora de introducir imágenes se guardan en la base de datos en formato base64, decidimos solucionarlo ampliando el tamaño del campo en la base de datos. Otro de los fallos fue la forma de integrar los distintos subapartados dentro de la aplicación y su relación con las secciones a las que pertenecían, esto se solucionó utilizando un `sectionId` que identifica a una sección en concreto, así se pueden hacer peticiones a base de datos juntándolo con el id pasado por la url.

No obstante, también tuvimos un parón bastante considerable que nos ocupó gran parte del tiempo de desarrollo, debido a que tuvimos que traducir a Java la información almacenada en las distintas Calculation Tools dentro de MIT App Inventor (ver imagen 10).

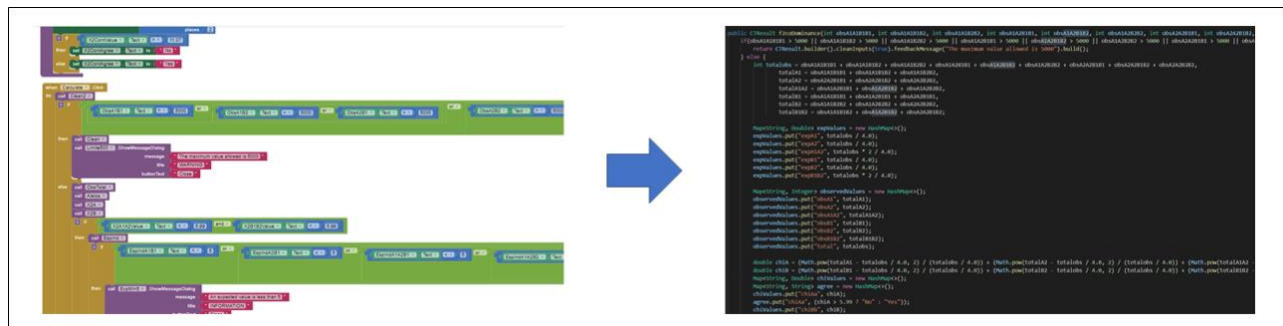


Imagen 10. Traducción de una Calculation Tool en MIT App Inventor a Java.

## Desarrollo de la aplicación móvil (Cliente)

### Planificación del desarrollo

El desarrollo de la aplicación móvil dio comienzo cuando la aplicación del servidor estaba prácticamente terminada, faltando algunos detalles de corrección o los aspectos menos importantes de las especificaciones.

Antes de comenzar el desarrollo de la aplicación móvil, fue necesario organizar y planificar cada una de las tareas, y decidir quién iba a dedicar más tiempo a esta parte. La planificación seguida se puede consultar en el subapartado de Planificación dentro del apartado de Herramientas, metodología y planificación.

### Proceso de desarrollo

En cuanto a la codificación, decidimos emplear tecnologías web para llevar a cabo la aplicación, usando los lenguajes HTML, CSS y JavaScript como base, apoyándonos en Apache Cordova. Esta decisión fue tomada porque con estos lenguajes puedes realizar un diseño Responsive que se adapte a cualquier dimensión de pantalla y con Apache Cordova podíamos crear distintos ejecutables a partir del mismo código para que se pudiese ejecutar la aplicación en cualquier sistema operativo de móvil, por lo que al usar esta alternativa solucionábamos los dos problemas principales.

Además, a partir de esta decisión, se podía comunicar la aplicación de móvil con el servidor realizando peticiones AJAX, de esta manera arreglando otro problema, el de la comunicación de la aplicación con el servidor. Es por ello por lo que, además de la aplicación de administración, se ha desarrollado una API en el servidor para atender las peticiones recibidas.

Para más información sobre el desarrollo de los contenidos que la API expone al exterior, debemos dirigirnos a la sección Desarrollo de la aplicación de administración (servidor).

Lo primero que se llevó a cabo fue la página principal de la aplicación, que contendría las distintas secciones y los apartados de *teoría*, *tests*, *problemas*, *calculation tools* y *libros recomendados*.

Una vez terminada, se procedió a desarrollar los distintos apartados de la misma. En primer lugar, se desarrolló los apartados de *teoría* y *problemas*, siendo éstos los más sencillos ya que la mayor parte de los datos venían del servidor, y se podrían probar las comunicaciones con el mismo. Una vez finalizados, se procedió con los apartados de *tests* y *libros recomendados*. En último lugar se dejó el apartado de *calculation tools*, ya que era el más complejo y largo y por ello se decidió dejar para el final.

Por último, se desarrolló una gestión de usuarios dentro de la aplicación para llevar un seguimiento de los tests que completaban, y también se añadió bloqueo a las distintas secciones de la aplicación, para que se fueran desbloqueando a medida que el usuario completaba la sección anterior.

Tras realizar las pruebas pertinentes y corregir algunos errores derivados de las mismas, se dio por concluido el desarrollo de la aplicación cliente.

## Seguimiento del desarrollo

En general, la codificación de la parte del cliente no tuvo problemas graves o parones significativos en el desarrollo. Como problema principal, se encuentra el tema de las *calculation tools*, debido a que al tener un modelo SPA y una única página HTML, no era conveniente introducir todas ellas en esa página ya que sería muy difícil de mantener. Se probaron distintas soluciones para el problema, eligiendo finalmente como solución la carga dinámica de ficheros HTML externos en la hoja principal para que de esta forma fuese más legible y mantenible. Además, debido a la gran cantidad de plantillas de las *calculation tools*, también llevó bastante tiempo realizarlas. Esto hizo que tuviéramos que alargar esta tarea en posteriores sprints.

Por lo demás, el desarrollo fue constante y se cumplió con la planificación establecida.

# Manuales

## Manual de usuario

### Aplicación móvil

En esta sección se van a explicar las distintas acciones que debe hacer un usuario para el correcto manejo de la aplicación móvil.

El flujo principal de la aplicación móvil depende de las distintas acciones que realice el usuario final (alumno), aunque a grandes rasgos sigue el siguiente esquema (que después detallaremos): el usuario introduce su nombre en la vista principal de login, una vez que pulsa el botón para loguearse accede a la vista principal donde están los distintos apartados (*Theory, Problems, ...*) asociados a una sección determinada de la aplicación. Que sea posible el acceso a esta sección depende del nivel de prioridad decidido por el usuario administrador en la aplicación servidor.

Al abrir la aplicación se nos mostrará la pantalla de bienvenida donde deberemos introducir nuestro nombre para que la aplicación recoja cierta información asociada a nuestro perfil (por ejemplo, para dar feedback al administrador dentro de la aplicación del servidor).

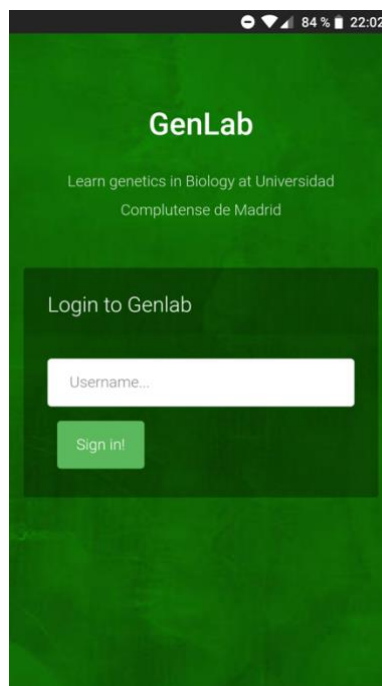


Imagen 11. Pantalla de login.

Una vez dentro de la aplicación, se nos mostrará una serie de secciones (herramientas de cálculo, problemas, tests, teoría y libros recomendados) adscritas a una aplicación en concreto y que podemos consultar pinchando en ellas.

Las aplicaciones (secciones dentro de la aplicación) disponibles actualmente son:

1. One Locus
2. Two Loci
3. Polyhybrid
4. Linkage
5. Epistasias

En esta pantalla además se nos muestra al igual que durante toda la navegación, un menú desplegable para seleccionar una en concreto, dentro de la cual están las mismas secciones nombradas anteriormente.

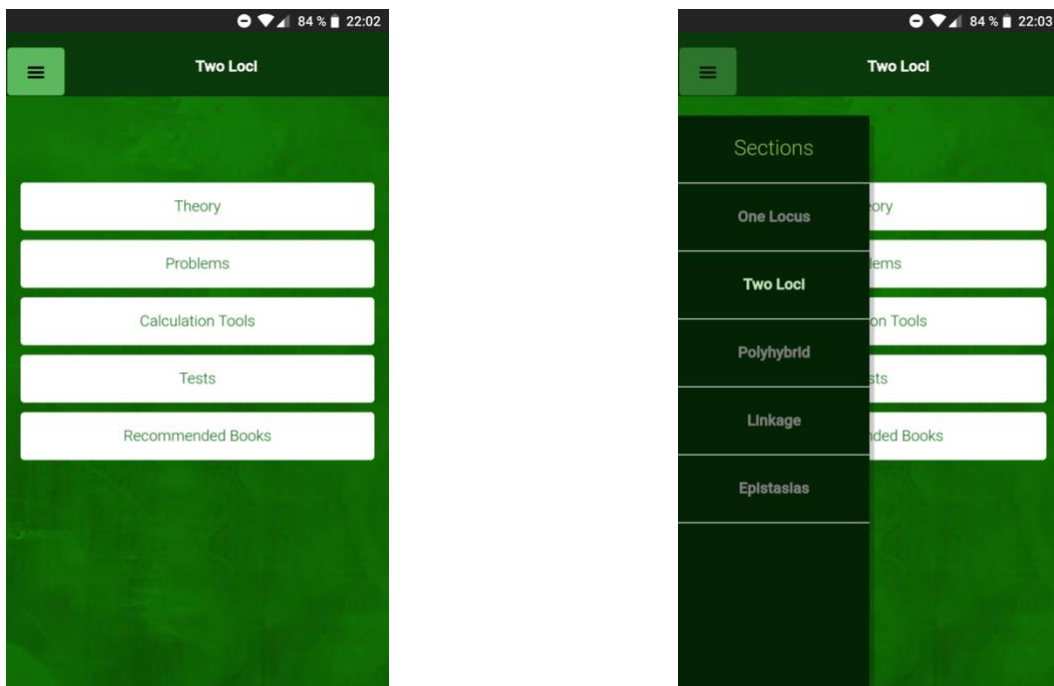


Imagen 12. Pantalla principal sin menú y con el menú desplegado.

Ahora trataremos los distintos apartados disponibles a consultar dentro de una aplicación específica:

Herramientas de cálculo (Calculation Tools):

En este apartado veremos un listado de las diferentes herramientas de cálculo disponibles para la aplicación seleccionada.

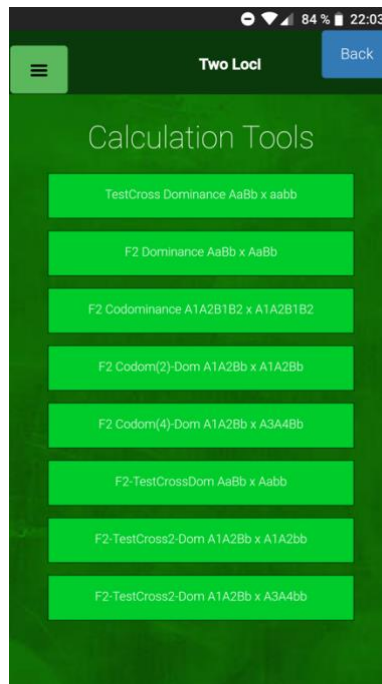


Imagen 13. Lista de Calculation Tools.

Al pinchar sobre una de ellas se nos mostrará una pantalla con la herramienta de cálculo, dentro de la cual estaremos nos encontramos con distintos inputs donde introduciremos los datos y una vez le demos a calcular se nos mostrarán los distintos valores de los resultados hallados.

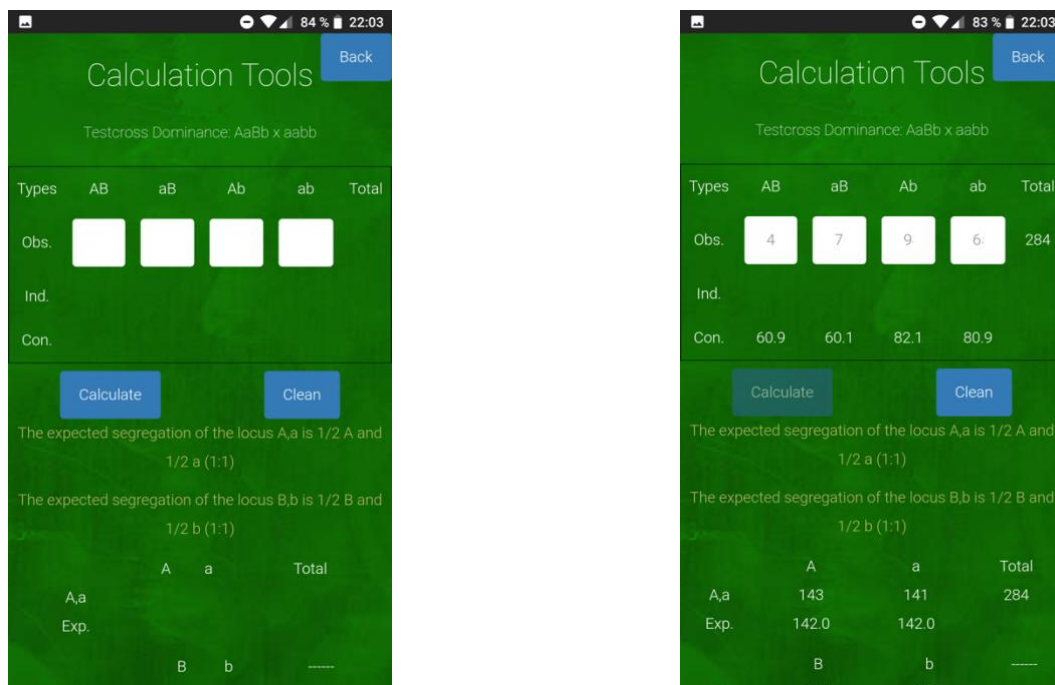


Imagen 14. Calculation Tool antes y después de realizar un cálculo.

Dentro de esta pantalla también hay un botón para borrar los resultados.

#### Problemas:

En esta pantalla se nos muestran listados todos los problemas propuestos dentro de la aplicación escogida, dentro de cada problema se plantearán distintas cuestiones que el usuario si lo desea deberá solucionar (no a través de la aplicación móvil).



Imagen 15. Ejemplo de vista de un problema.

#### Tests:

Al igual que con las herramientas de cálculo, aparecerá una lista de los distintos tests disponibles para la aplicación escogida. Al entrar dentro de uno de estos test se presentarán las distintas preguntas con sus respuestas a responder. El usuario deberá responder la respuesta que crea que es correcta pinchando sobre ella, dicha respuesta se mostrará en rojo si es incorrecta o en verde si es correcta.



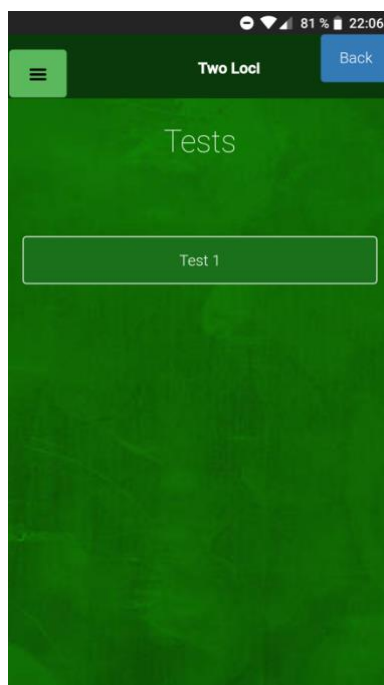


Imagen 16. Vista de la lista de tests.

Dentro de los tests puede haber la posibilidad de cuestiones multirrespuesta si el administrador decide que así sea, en ese caso el comportamiento será parecido a si fuese de una única respuesta.



Imagen 17. Interior de un test con una pregunta respondida erroneamente.

### Teoría:

En este apartado se mostrará la teoría correspondiente a la aplicación elegida junto con imágenes explicativas si el administrador así lo decide. Esta teoría explicará los distintos conceptos incorporados dentro de la aplicación. La vista es igual a la vista de problemas.

### Libros recomendados:

Aquí se presentarán los distintos libros recomendados para poder entender la aplicación elegida, ya que tienen relación con los temas tratados dentro de ella; la información mostrada es el título del libro, el autor, la editorial, el código ISBN y el enlace a dicho libro.



Imagen 18. Lista de libros recomendados.

En definitiva, estos son los distintos apartados que podemos encontrar dentro de cada aplicación.

A parte de lo tratado anteriormente, para elegir una aplicación dentro del menú desplegable deberemos haber completado los tests en las secciones con menos prioridad, por lo que si intentamos entrar en una sección sin haber completado las anteriores se mostrará un mensaje.

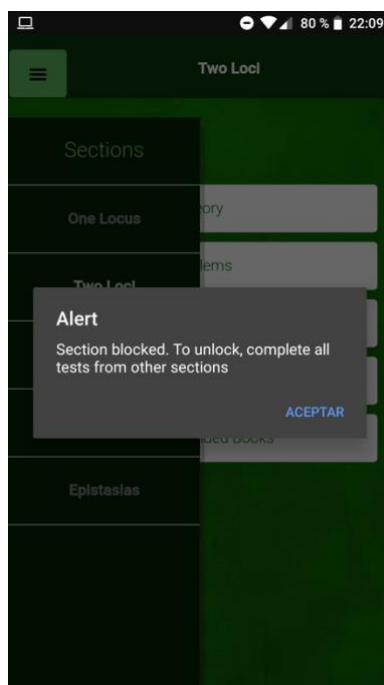


Imagen 19. Aviso que se muestra cuando intentas entrar en una sección no desbloqueada.

## Aplicación servidor

Aquí explicaremos las distintas acciones que debe hacer un usuario para el correcto manejo de la aplicación del servidor.

Al abrir la aplicación se nos mostrará la pantalla de login donde debemos introducir el nombre de usuario y contraseña autorizados para el rol de administrador (para añadir nuevos usuarios administradores, ver la sección *Añadiendo un nuevo administrador*)



Imagen 20. Pantalla de Login de la aplicación del servidor.

Una vez dentro de la aplicación, se nos dará la opción de movernos a través de las distintas secciones (One Locus, Two Loci, Polyhybrid, Linkage o Epistasia) y en cada sección podremos administrar sus distintos apartados (i.e.: Calculation Tools, Theory, Problems, Tests, Recommended Books y Feedback).

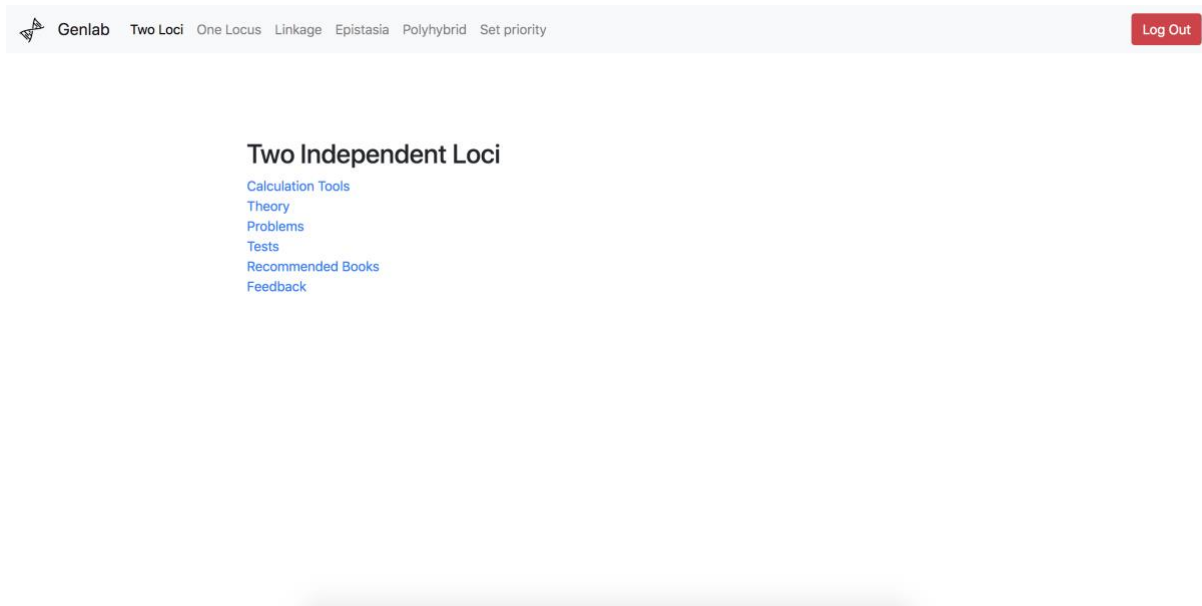


Imagen 21. Pantalla principal de la aplicación (sección Two Loci).

Dentro de una sección específica podremos administrar los siguientes apartados:

Calculation Tools:

En este apartado veremos un listado de las diferentes herramientas de cálculo disponibles para la aplicación seleccionada. Realmente no podremos administrar las Calculation Tools, sino simplemente verlas y probarlas para comprobar su eficacia.

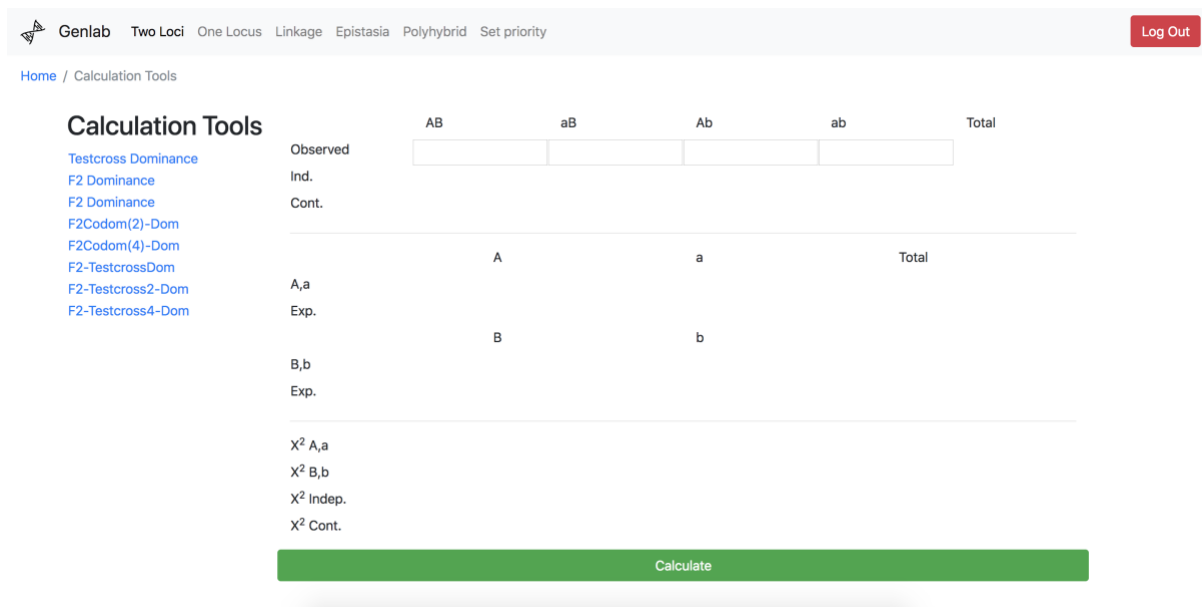


Imagen 22. Calculation Tool llamada Testcross Dominance.

## Problems:

En esta pantalla se nos muestran listados todos los problemas disponibles para una sección y se nos permite borrarlos, así como añadir uno nuevo o editarlos. Para que la edición sea sencilla disponemos de una herramienta WYSIWYG con herramientas para añadir imágenes, cambiar los tipos de fuente y colores, etc...

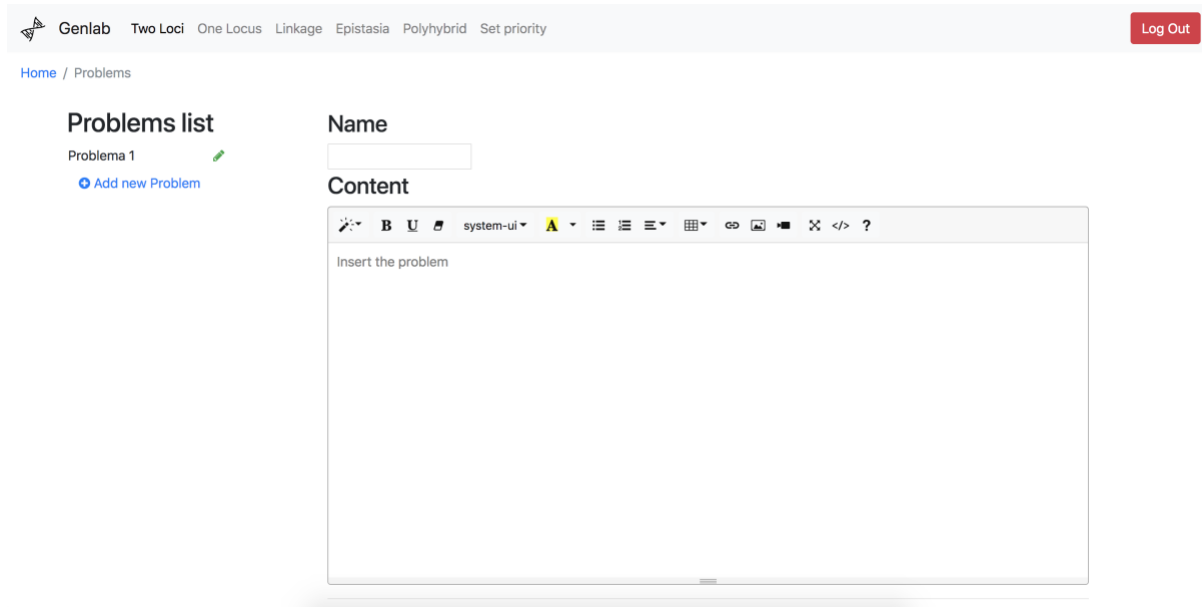


Imagen 23. Añadiendo un nuevo problema.

## Tests:

Al igual que con las herramientas de cálculo, aparecerá una lista de los distintos tests disponibles para la aplicación escogida. Podremos editar o eliminar tests existentes, así como crear un nuevo test.

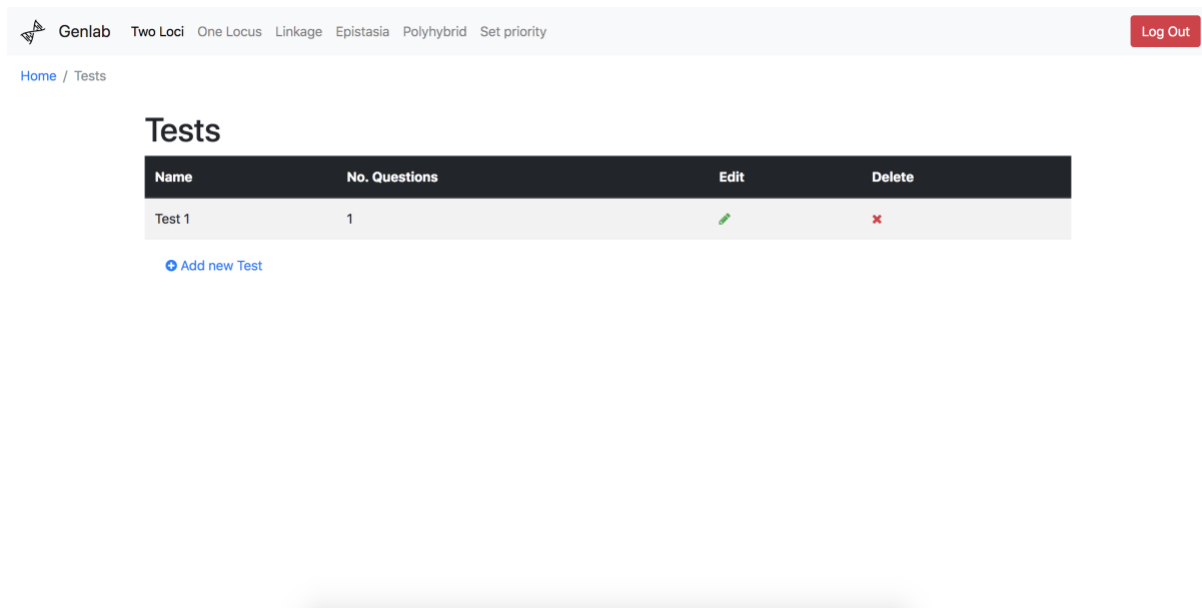


Imagen 24. Lista de Tests actualmente creados.

Al crear o editar tests podremos añadir dinámicamente tantas preguntas o respuestas como queramos y marcar qué respuesta (o respuestas) es la correcta.

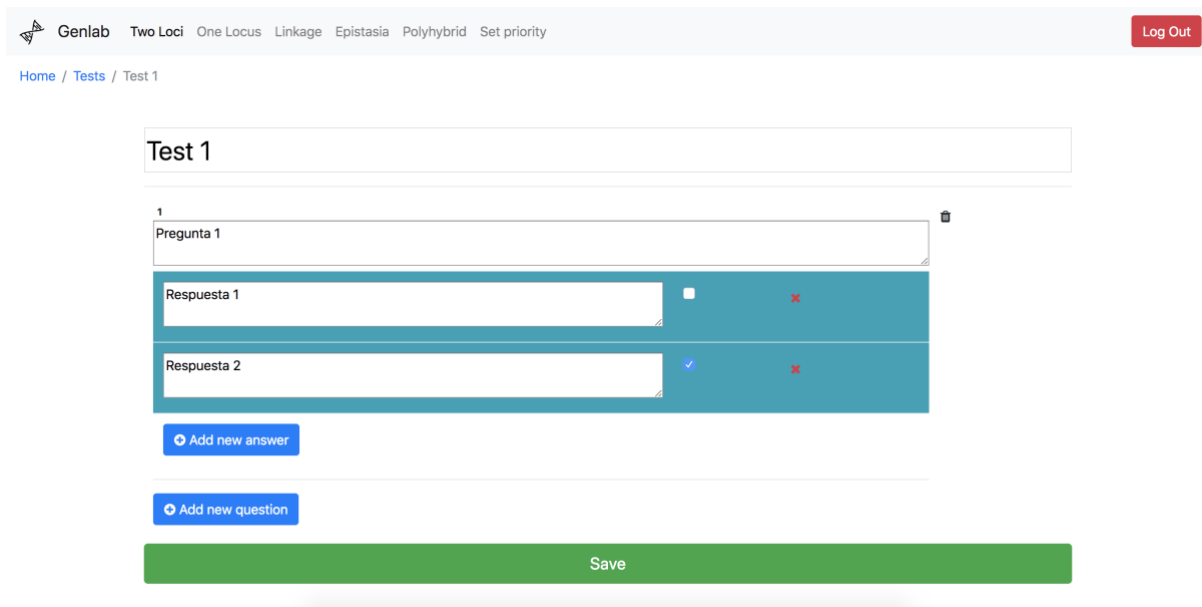


Imagen 25. Creando o editando un test.

Theory:

El uso de este apartado será igual que el uso del apartado problems, por lo que si existe alguna duda respecto de su funcionamiento deberemos remitirnos al manual de uso del apartado Problems.

## Recommended books:

Aquí se presentarán los distintos libros recomendados añadidos por el momento, teniendo la posibilidad de modificar o eliminar los existentes, o añadir uno nuevo.

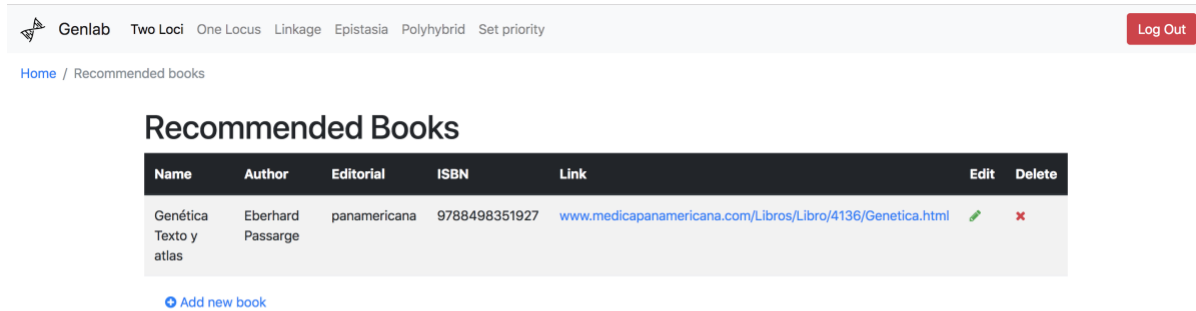


Imagen 26. Lista de libros añadidos.

GenLab Two Loci One Locus Linkage Epistasia Polyhybrid Set priority Log Out

Home / Recommended Books / Genética Texto y atlas

Name:

Author:

Editorial:

ISBN:

Link:

Don't include "http://" in your link.

[Save](#)

Imagen 27. Añadiendo o modificando un libro.



## Feedback:

En este apartado podremos ver el feedback automático generado por la aplicación sobre los tests que van resolviendo los diferentes usuarios. Se mostrará para cada usuario los tests que ha resuelto y cuantos fallos ha tenido en dicho test.

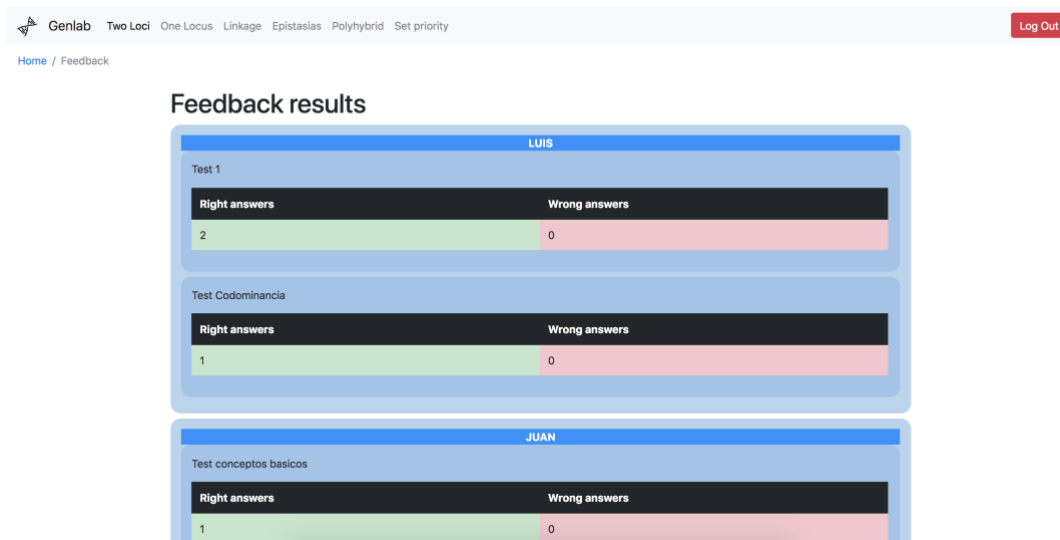


Imagen 28. Pantalla de feedback.

## Set priority:

En este apartado, que podemos encontrar en la barra superior de navegación, podemos reordenar arrastrando las diferentes secciones para asignarles una prioridad, de manera que una sección no se desbloquea hasta que no se han completado todos los tests de la anterior.

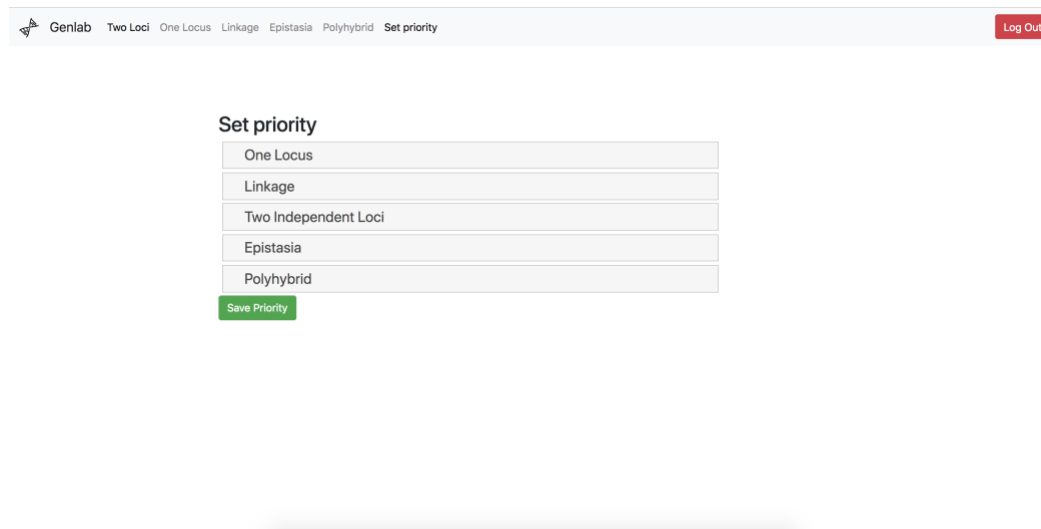


Imagen 29. Asignando prioridad a las secciones.

# Manual del desarrollador

## Manual de extensión de la aplicación móvil

En este apartado se procede a explicar las instrucciones necesarias para llevar a cabo una ampliación o extensión de la aplicación. Los detalles arquitectónicos de la misma están definidos en la sección *Arquitectura de la aplicación cliente*.

### Extensión general de la aplicación

Es posible implementar mejoras o añadir nuevos elementos en un futuro a la aplicación. Para ello, es necesario diferenciar si se quiere añadir elementos generales, o una nueva Calculation Tool. En este último caso es necesario seguir una serie de pasos adicionales para facilitar la mantenibilidad y la modularización de las mismas.

#### Añadiendo nuevos elementos

Para añadir nuevos elementos, simplemente hay que añadirlos en el archivo *index.html*, identificados mediante la etiqueta `id` de HTML, ya que como se explica anteriormente, se usa el modelo SPA y se controla la visibilidad de los elementos mediante los scripts a medida que el usuario navega a través de la aplicación. Es posible, si ese es su deseo, el añadir nuevos ficheros HTML y llamarlos mediante la etiqueta `href`, aunque no es recomendable.

#### Añadiendo nuevos estilos y scripts

Para manejar su estilo, se pueden crear nuevas hojas de estilo CSS y añadirlas mediante la etiqueta `link` al HTML principal, y poner los estilos que se deseen, aunque es mejor si se utilizan algunas de las hojas de estilo ya creadas anteriormente. Hay tres hojas de estilo creadas: *styles.css* tiene todos los estilos correspondientes a toda la aplicación en general, *form-elements.css* contiene estilos propios del formulario de acceso a la aplicación y *ctools.css* tiene todos los estilos referentes a las Calculation Tools que existen en la aplicación.

Para los scripts, sucede igual que en caso anterior. Es posible añadir nuevos scripts mediante la etiqueta `script` al HTML principal, aunque es recomendable usar los scripts ya creados anteriormente. En este caso, tenemos el fichero *home.js*, donde se encuentra el comportamiento de la aplicación en general y las peticiones AJAX que recogen del servidor los problemas, los libros recomendados, la teoría y los tests, y *ctools.js* donde se encuentra el comportamiento de las

Calculation Tools, así como las peticiones AJAX para pedir los resultados al servidor (donde se encuentran alojadas las fórmulas de cálculo).

En estos casos, es posible añadir, eliminar o editar los elementos sin ningún tipo de limitación, respetando la estructura de diseño proporcionada para permitir que nuevos desarrolladores puedan modificar la aplicación.

## Añadiendo una nueva Calculation Tool

En el caso de las Calculation Tools, el proceso a seguir es distinto. Actualmente se dispone de un gran número de Calculation Tools (incluso más en un futuro). Cada una tiene sus propios elementos y estilos diferentes. Introducir todo el código de las distintas Calculation Tools dentro del archivo principal *index.html* haría que resulte en un fichero extremadamente grande y complicado de mantener y controlar. Para solucionar este problema se ha decidido cargar el contenido de cada Calculation Tool desde ficheros HTML externos que se insertan en el HTML principal.

Estos ficheros externos, cada uno correspondiente a una única Calculation Tools, se encuentran en la carpeta *ctools*, y a su vez, dentro de la carpeta cuyo nombre es el de la sección correspondiente.

Por ello, para incluir una nueva Calculation Tool en la aplicación, respetando la estructura existente en la misma, hay que seguir los siguientes pasos. A modo de ejemplo, vamos a suponer que queremos añadir la Calculation Tool “*Multiple Genes*” en la sección “*Polyhybrid*”.

- En primer lugar, es necesario crear un fichero HTML que tenga los elementos de la nueva Calculation Tool. El diseño de esta es completamente libre y se puede añadir todo lo necesario. Tan solo hay que tener en cuenta lo siguiente:
  - o Los estilos nuevos irán dentro del fichero *ctools.css* (o incluso, se pueden reutilizar algunos estilos de este fichero ya usados en otras Calculation Tools, por lo que conviene revisarlo antes de pensar en añadir algún estilo nuevo).
  - o Si se quiere añadir un botón que limpie inputs y resultados, deberá tener la clase `btn-clean`, y los elementos que quieran ser borrados deberán tener la clase `clean`, ya que ya existe una función en JavaScript que se encarga de recoger el evento de ese botón y de limpiar los elementos marcados con la clase `clean`.

- El botón que se encargue de calcular deberá tener un identificador descriptivo, ya que será usado posteriormente. Un buen identificador podría ser el resultado de unir el nombre de la sección con el nombre de la Calculation Tool (esto es lo que se usa actualmente).
- El nombre del fichero HTML deberá ser *nombreSeccion\_nombreCalculationTool.html*, y encontrarse en la carpeta de la sección correspondiente, para que la función existente pueda cargar sin problemas este fichero sin ninguna adicción de código (y para que el nombre sea más descriptivo a la hora de mantener las Calculation Tools). En el caso de ejemplo, este fichero se debe llamar *polyhybrid\_MultipleGenes.html*, y debe estar dentro de la carpeta *polyhybrid*.
- A continuación, es necesario añadir un elemento “de enlace” en el archivo *index.html* para que cuando se pulse en él se pueda proceder a la carga del fichero HTML externo. Hay una zona del fichero donde se encuentran todos estos elementos de enlace como se muestra en la imagen 30. Una vez añadido este nuevo elemento, es necesario poner en la etiqueta HTML un elemento *data* denominado *ctool*, con el nombre de la Calculation Tool (el mismo usado en el nombre del fichero HTML). En el caso de ejemplo, este elemento de enlace sería el siguiente, dentro de la lista de Calculation Tools de *polyhybrid*:

```
<div data-ctool="MultipleGenes">Multiples Genes</div>
```

Una vez realizado lo anterior, la nueva Calculation Tool ya se verá cargada y mostrará su contenido cuando se seleccione el elemento correspondiente.

```
<!-- CTTOOLS -->
<div id="ctoolsView">
  <div id="one-locus-ctools">
    <div class="ctools-list">
      <div data-ctool="Testcross">Testcross Aa x aa</div>
      <div data-ctool="F2Dominance">F2 Dominance Aa x Aa</div>
      <div data-ctool="F2Codominance">F2 Codominance A1A2 x A1A2</div>
      <div data-ctool="Codominance3">Codominance 3 Alleles A1A2 x A1A3</div>
      <div data-ctool="Codominance4">Codominance 4 Alleles A1A2 x A3A4</div>
    </div>
  </div>
  <div id="two-independent-loci-ctools">
    <div class="ctools-list">
      <div data-ctool="Testcross">TestCross Dominance AaBb x aabb</div>
      <div data-ctool="F2Dominance">F2 Dominance AaBb x AaBb</div>
      <div data-ctool="F2Codominance">F2 Codominance A1A2B1B2 x A1A2B1B2</div>
      <div data-ctool="F2Codom2">F2 Codom(2)-Dom A1A2Bb x A1A2Bb</div>
      <div data-ctool="F2Codom4">F2 Codom(4)-Dom A1A2Bb x A3A4Bb</div>
      <div data-ctool="F2TestcrossDom">F2-TestCrossDom AaBb x Aabb</div>
      <div data-ctool="F2TestcrossDom2_1">F2-TestCross2-Dom A1A2Bb x A1A2bb</div>
      <div data-ctool="F2TestcrossDom2_2">F2-TestCross2-Dom A1A2Bb x A3A4bb</div>
    </div>
  </div>
</div>
```

Imagen 30. Lista de Calculation Tools en el archivo *index.html*

- Por último, es necesario crear una petición AJAX en el fichero *ctools.js* que se encargue de llamar a la API del servidor con los datos pertinentes al seleccionar el botón de calcular de la nueva Calculation Tools, y que se encargue de mostrar en los elementos correspondientes los resultados. La petición debe tener las siguientes propiedades:
  - La petición debe ser de tipo POST
  - La url debe ser <http://ingenias.fdi.ucm.es:60070/api/v1/calctool?CTid=XX>, siendo XX el identificador de la Calculation Tool asignado por el servidor (ver manual de desarrollador de la aplicación de administración para más información en este mismo documento).
  - En la petición debe incluirse una cabecera u opción de tipo CORS para que el servidor acepte la petición.
  - El tipo de los datos (`contentType`) que se envíen (en caso de mandarse) será `application/json`
  - Los datos (en caso de mandarse) se mandan en formato JSON.

Para ver cómo realizar los cálculos y mandarlos a la aplicación móvil desde el servidor, revisa el manual del desarrollador de la aplicación del servidor (la aplicación de administración) en este mismo documento.

En la imagen 31 se muestra un ejemplo de petición AJAX para la Calculation Tool *Testcross* de la sección *One Locus*.

```
$.ajax({
  type: "POST",
  url: "http://ingenias.fdi.ucm.es:60070/api/v1/calctool?Ctid=10",
  beforeSend: function(request) {
    request.setRequestHeader("Access-Control-Allow-Origin", "*");
  },
  contentType: "application/json",
  data: JSON.stringify({
    "A": alleles_A,
    "a": alleles_a
  }),
  success: function(data, textStatus, jqXHR) {
    if (!data.cleanInputs) {
      $("#total-Testcross").text(alleles_A + alleles_a);

      $("#Expected-A-Testcross").text(data.expectedValues.expA.toFixed(1));
      $("#Expected-a-Testcross").text(data.expectedValues.expA.toFixed(1));

      $("#value-Testcross").text(data.resultValues.chi.toFixed(2));
      $("#agree-Testcross").text(data.agree.chi);
      if (data.result) {
        $("#result-message-Testcross").text(data.result);
      }
      if (data.feedbackMessage) {
        alert(data.feedbackMessage);
      }
    } else {
      alert(data.feedbackMessage);
    }
  },
  error: function(jqXHR, textStatus, errorThrown) {
    alert("Internet connection must be available to get and show the results");
  }
});
```

Imagen 31. Petición AJAX de una Calculation Tool.

Para ejecutar la aplicación, es necesario descargar e instalar el archivo de instalación correspondiente según su sistema operativo (en Android, por ejemplo, el archivo .apk), y a continuación, iniciarla en su dispositivo.

## Manual de extensión de la aplicación de administración

A continuación, se procederá a explicar el procedimiento a seguir para la extensión de la parte servidor de la aplicación. Las herramientas utilizadas para su implementación están recogidas en la sección Herramientas de esta memoria. La arquitectura de la aplicación está recogida en la sección Arquitectura de la aplicación servidor de esta memoria.

## Prerrequisitos para el desarrollo

Para comenzar a desarrollar este proyecto, debemos instalar en nuestro ordenador una versión de Java igual o superior a la 1.8 (<https://www.java.com/es/download/>) y Maven (referencia a <https://maven.apache.org/>). Una vez instalado Maven, mediante la consola, nos dirigiremos a la carpeta donde esté nuestro proyecto (donde deben estar los archivos *mvnw* y *pom*) y ejecutaremos el comando *mvn clean install* para instalar todas las dependencias necesarias.

Por último, debemos instalar Lombok (<https://projectlombok.org/download>), una librería de Java que permite simplificar al máximo los objetos que contienen los datos de nuestra aplicación.

## Añadiendo una nueva sección a nuestra aplicación

En el momento de la publicación de este proyecto, las secciones existentes son las descritas en la sección de esta memoria llamada Especificación de requisitos. Para añadir una nueva sección en nuestra aplicación sólo se requiere añadir la sección en el fichero *application.yml* encontrado en el directorio *src/main/resources* y añadir el nombre de la sección, su número identificador y su prioridad en la tabla *sections* de la base de datos. Por último, hará falta añadir la sección con su *id* en el la enumerado llamado *SectionsMapping* encontrado en el paquete *Models*. No hará falta añadir más información acerca de la sección y ya existirá en la aplicación y será visible para los alumnos (aunque vacía por el momento).

## Añadiendo un nuevo módulo a nuestra aplicación

En el momento de la publicación de este proyecto, los módulos existentes son los descritos en la sección de esta memoria Especificación de requisitos. Para añadir un nuevo módulo y mantener la modularidad y arquitectura de nuestra aplicación deberemos seguir el siguiente procedimiento:

1. Crearemos, en el paquete *controllers* una nueva clase que mapee todos los endpoints necesarios para gestionar ese módulo. Para mantener el estilo del código, la propia clase mapeara las URLs que empiecen por “/<nombre-modulo>”, por ejemplo, en el módulo *tests*, la clase *TestsController* mapeará todas las URLs que empiecen por “/tests” y cada método interno mapeará una URL que haga referencia a las acciones que realiza dicho método.
2. Crearemos, en el paquete *models*, los objetos necesarios para este nuevo módulo. Estos objetos podrán ser entidades que se volcarán a la base de datos u objetos que utilice la vista. También se podría implementar el patrón Transfer si fuera necesario, aunque nosotros no hayamos hecho uso de este. Para la creación de estos objetos de manera simple, recomendamos el uso de las anotaciones proporcionadas por la herramienta “lombok” que se han utilizado en el resto de los objetos.
3. En el paquete *services*, añadiremos un paquete con el nombre “<nombre-modulo>Service” que contendrá la interfaz y la implementación o implementaciones necesarias de esa interfaz.
4. Añadiremos en el paquete *repositories* una nueva interfaz con el nombre “<nombre-modulo>Repository” que extenderá la clase *CrudRepository* parametrizada con el tipo de datos que manejará y el tipo de dato que sea su *id*.
5. Crearemos una tabla en la BD que represente al nuevo módulo a la que accederemos mediante el repositorio creado en el paso anterior.

6. Por último, deberemos crear la interfaz gráfica de este nuevo módulo (en caso de ser accesible gráficamente y no formar solo parte de la API) creando las plantillas necesarias en la carpeta `resources`.

## Añadiendo una nueva Calculation Tool a nuestra aplicación

Para añadir una nueva Calculation Tool y mantener la modularidad y arquitectura de nuestra aplicación seguiremos los siguientes pasos:

1. En el paquete `Models` añadiremos la Calculation Tool deseada indicando su nombre, id de sección e id de la propia Calculation Tool.
2. En el paquete `Services/ctservice`, en la sección correspondiente añadiremos, en la implementación deseada de la interfaz, un método que reciba los parámetros de cálculo de entrada y devuelva un objeto del tipo `CTResult` (así como todos los métodos auxiliares necesarios) realizando en su interior todos los cálculos pertinentes.
3. En el paquete `Controllers`, en la clase `CalculationToolsController` añadiremos, en el método `getCalcResult` la llamada al método de cálculo creado en el paso anterior. Lo añadiremos teniendo en cuenta que el `switch` más externo indica el id de sección y el `switch` interno indica el id de la Calculation Tool dentro de esa sección.

## Añadiendo un nuevo usuario administrador

A fecha de entrega de este proyecto el único usuario existente en la base de datos con el rol de administrador es el recogido en el fichero adjunto llamado *aplicación-servidor.txt*. Para añadir nuevos usuarios con dicho rol a la base de datos deberemos acceder a esta y crear una nueva entrada en la tabla `users` e insertar:

- En el campo `email`, el identificador del usuario.
- En el campo `role`, el rol, que en este caso será ADMIN
- En el campo `password`, la contraseña encriptada según el algoritmo BCrypt. Para calcular de manera sencilla el hash a partir de una cadena de texto recomendamos el uso de <https://bcrypt-generator.com/>

Queda como mejora futura la posibilidad de añadir usuarios a través de la interfaz de la aplicación web, lo que por el momento no ha sido implementado ya que se ha centrado el desarrollo en otras funcionalidades consideradas más importantes.



## Reparto del trabajo

En este apartado se explica la participación de los miembros del equipo en el desarrollo de la aplicación en su totalidad.

Cabe destacar que todos los aspectos de diseño, las decisiones y las conclusiones han sido debatidas y tomadas en conjunto. La estructura del grupo ha seguido un esquema descentralizado.

En cuanto a la distribución del trabajo a realizar sobre la aplicación entre los distintos miembros del grupo se ha realizado de la siguiente manera (Siempre teniendo en cuenta que no ha sido una distribución estricta, y que en caso de necesitar más apoyo en una parte nos hemos podido centrar los 3 integrantes en ella):

- Pablo Arranz Ropero se ha encargado mayormente de desarrollar la aplicación del lado servidor (la aplicación de administración) y de la administración de los sistemas (es decir, Alojamiento y ejecución de la aplicación en el servidor).
- Juan Alberto Camino Sáez se ha encargado mayormente del desarrollo de la aplicación del lado cliente (la aplicación móvil), así como del desarrollo de ciertas plantillas del lado del servidor.
- Carlos López Martínez se ha encargado mayormente de desarrollar junto con Pablo la aplicación del lado servidor (la aplicación de administración), aunque también ha servido de apoyo a Juan Alberto en el desarrollo de la aplicación móvil.

Esta memoria ha sido desarrollada entre los tres integrantes del grupo de la manera más distribuida posible, siempre centrándose cada uno en las secciones que más conciernen al área de la aplicación que han desarrollado.

## Conclusiones y trabajo futuro

Inicialmente, César Benito Jiménez (Profesor de la Facultad de Biología) nos presentó su problema: necesitaba una herramienta para presentar a sus alumnos una cierta cantidad de materia y ejercicios. Hasta el momento él había desarrollado unas aplicaciones móviles que no veía factibles a largo plazo por su peso en memoria y la dificultad de actualización de los contenidos.

A partir de ahí decidimos crear GenLab, una aplicación que permite gestionar todos estos contenidos mediante una aplicación web accesible desde cualquier navegador, de manera que sin apenas conocimientos informáticos la gestión sea lo más sencilla posible. También creamos una aplicación móvil para que todos sus alumnos puedan acceder a estos contenidos y el profesor César Benito Jiménez pueda obtener información de qué tal están haciendo sus alumnos los ejercicios mediante un panel bastante visual.

Las dos aplicaciones obtenidas se han realizado teniendo en cuenta la posible extensión y mantenimiento de ellas, debido a esto, en este documento se recogen las instrucciones necesarias para ello (véase *Manual del desarrollador*).

En resumen, se han obtenido las dos aplicaciones planteadas al introducir esta memoria, cumpliendo así las expectativas creadas durante la planificación del proyecto.

Estas aplicaciones han permitido mejorar varios aspectos:

1. El uso de espacio en los dispositivos móviles ha pasado a ser de tan solo 6.41 MB, cuando anteriormente la suma de todas las aplicaciones ocupaba un total de 58.89 MB.
2. La gestión por parte del profesorado es mucho más sencilla. Se ha pasado de necesitar desarrollar parte de la aplicación mediante “AppInventor” en caso de querer extender, por ejemplo, la teoría, a simplemente acceder al editor de texto de la aplicación web servidor.
3. La obtención de información de los resultados del alumnado en los diferentes tests por parte del profesorado, permitiendo así enfocar las clases o ejercicios a ciertos temas dependiendo de las necesidades de los alumnos.

La organización y planificación del proyecto está recogida en la sección *Metodologías* de esta memoria.

Tras los resultados y los conocimientos obtenidos durante el desarrollo del proyecto obtenemos varias conclusiones:

1. Un proyecto full-stack donde se ha tenido que implementar una parte servidor, una parte cliente y la interacción entre ellas incorporando además una base de datos para mantener la consistencia de la información, requiere de una buena planificación, así como del reparto y división de las tareas para conseguir los objetivos marcados.
2. Para que una aplicación ocupe mayor volumen de mercado es conveniente que dicha aplicación sea compatible con el mayor número de dispositivos posibles, ya sea en relación con el sistema operativo utilizado (multiplataforma) como con el tamaño de dicho dispositivo (diseño responsive).

### **Trabajo futuro**

Como trabajos pendientes se encuentran los siguientes:

- Realizar experiencias de usuario con usuarios reales (alumnos de la Facultad de Biología interesados en el uso de la aplicación) permitiendo así obtener opiniones reales para mejorar o incorporar distintos aspectos en la aplicación, dichas experiencias de usuario deberían realizarse bajo condiciones de no inmersión por parte de los desarrolladores o interesados de la aplicación, dejando a los usuarios de pruebas a su libre albedrío, dichos usuarios deberían rellenar una encuesta predefinida con distintos aspectos a valorar durante el manejo de la aplicación. Estas experiencias, en definitiva, permitirían mejorar la usabilidad de la aplicación.
- Añadir usuarios administradores desde la interfaz de usuario, en lugar de directamente sobre la base de datos.
- Incrementar el número de secciones (aplicaciones) según las necesidades de la Facultad de Biología, esto se podría realizar gracias al manual definido en esta memoria en la sección *Manual de extensión de la aplicación de administración*.
- Solucionar posibles fallos (bugs) que pudiese contener la aplicación, esto podría solucionarse con una etapa de testeo antes de pasar la aplicación a producción, o en versiones siguientes si así lo deseara el propietario de la aplicación.
- La sección de feedback en la parte del servidor podría modificarse en forma de gráfica para que el usuario administrador que quiera verlo (profesor) pueda comprender los resultados de una forma más visual.
- Mantenimiento general de la aplicación, incorporando cambios ya sea por fallos o por posibles mejoras que se pudiesen incorporar en la aplicación.

## Conclusions and future work

Initially, César Benito Jiménez (Professor at the School of Biology) presented us with his problem: He needed a tool to introduce his students to a certain number of subjects and exercises. Until now, he had developed mobile applications that he did not see as feasible in the long term due to the space in storage needed and the difficulty of updating the contents.

From there we decided to create GenLab, an application that allows to manage all these contents through a web application accessible from any browser, so that without hardly any computer knowledge, the management is as simple as possible. We also created a mobile application so that all the students can access these contents and César can obtain information about how their students are doing with a very visual panel.

Both applications have been made taking into account the possible extension and maintenance of them, because of this, this document contains the necessary instructions for it. (See Manual del desarrollador).

In short, the two applications proposed in the introduction of this report have been obtained, thus fulfilling the expectations created during the planning of the project.

These applications have led to improvements in several areas:

1. The amount of used storage on mobile devices has decreased to just 6.41 MB, compared to a total of 58.89 MB for all previous applications.
2. Teacher management is much simpler. It has gone from needing to develop part of the application using "AppInventor" in case of wanting to extend, for example, the theory, to simply accessing the text editor of the new web application.
3. Obtaining information on the results of the students in the different tests by the teaching staff, thus allowing them to focus the classes or exercises on certain areas depending on the needs of the students.

The organization and planning of the project is included in the section called Metodologías of this report.

After the results and the knowledge obtained during the development of the project we obtain several conclusions:

1. A full-stack project where a server part, a client part and the interaction between them had been implemented also incorporating a database to maintain the consistency of the

information, requires good planning, as well as the distribution and division of tasks to achieve the targets.

2. In order to cover a larger market volume, it is advisable that the application is compatible with as many devices as possible, either in relation to the operating system used (multi-platform) or to the size of the device (responsive design).

## **Future work**

Pending jobs:

- Carrying out user experiences with real users (students of the School of Biology interested in the use of the application) allowing us to obtain real opinions to improve or incorporate different aspects in the application. Such user experiences should be carried out under conditions of non-immersion by the developers or interested parties of the application, leaving the test users to their own free will, these users should fill out a predefined survey with different aspects to be evaluated during the handling of the application. These experiences, in short, would improve the usability of the application.
- Adding administrator users from the user interface, rather than directly from the database.
- Increasing the number of sections (applications) according to the needs of the School of Biology, this could be done thanks to the manual defined in this report in the section called *Manual de extensión de la aplicación de administración*.
- Fixing possible bugs that the application might contain. This could be solved with a testing stage before the application goes into production phase, or in later versions if the application owner wishes so.
- The feedback section on the server side could be modified as a graph so that the administrator user who wants to see it (teacher) can understand the results in a more visual way.
- General maintenance of the application, incorporating changes due to failures or possible improvements that could be incorporated in the application.

# Resultados de aprendizaje

Gracias a este proyecto hemos adquirido conocimientos en:

1. Desarrollo web con Java Spring Framework.
2. Desarrollo móvil con tecnologías web y Apache Cordova.
3. Administración de sistemas y despliegue de aplicaciones en un servidor ajeno a nosotros.
4. Creación y administración de una base de datos relacional MySQL.
5. Aplicación de metodologías ágiles en un grupo y proyecto real, padeciendo los posibles contratiempos que se pueden producir.
6. Extracción de requisitos en un problema real y comunicación con el cliente que nos encargó una solución para dicho problema.
7. Planificación de un proyecto con determinados requisitos (tanto funcionales como de fechas o de artefactos entregables mínimos)

# Bibliografía

- [1] Librería Java Thymeleaf (<https://www.thymeleaf.org/documentation.html>)
- [2] Librería Java Lombok (<http://jnb.ociweb.com/jnb/jnbJan2010.html>)
- [3] Guías Java Spring (<https://spring.io/guides>)
- [4] Documentación Apache Cordova (<https://cordova.apache.org/docs/en/latest/>)
- [5] Librería Javascript MathJax (<https://docs.mathjax.org/en/latest/start.html>)
- [6] Barra de navegación Bootstrap (<https://getbootstrap.com/docs/4.0/components/navbar/>)
- [7] Aplicaciones originales desarrolladas por César Benito Jiménez (<http://biologicas.ucm.es/genetica-android>)
- [8] Core J2EE Patterns: Best Practices and Design Strategies (Dan Malks, Deepak Alur, John Crupi)
- [9] Design Patterns: Elements of Reusable Object-Oriented Software (Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides)
- [10] Pro Spring 5: An In-Depth Guide to the Spring Framework and Its Tools (Iuliana Cosmina, Rob Harrop, Chris Schaefer, Clarence Ho)
- [11] Essential Scrum: A Practical Guide to the Most Popular Agile Process (Kenneth S. Rubin)

# Glosario

API: Application Programming Interface.

Calculation Tool: Herramienta de la aplicación con la que podemos realizar cálculos relacionados con la genética.

CORS: Cross Origin Resource Sharing.

IDE: Integrated Development Environment.

ISBN: International Standard Book Number.

JDK: Java Development Kit.

MIT: Massachusetts Institute of Technology.

MVC: Modelo Vista Controlador.

REST: Representational State Transfer.

UCM: Universidad Complutense de Madrid.

WYSIWYG: What You See Is What You Get.