

GenLab



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)

Autorización de difusión y utilización

Dedicatorias

Agradecimientos

Índice

1. Índice de imágenes
2. Índice de diagramas
3. Resumen
4. Abstract
5. Introducción
6. Arquitectura de la aplicación
7. Materiales y métodos
8. Especificación de requisitos
9. Proceso de desarrollo
10. Manuales
 - 10.1. Manual de usuario
 - 10.2. Manuales del desarrollador
 - 10.2.1. Manual de extensión de la aplicación móvil
 - 10.2.2. Manual de extensión de la aplicación de administración
11. Apéndices
12. Resultados y discusión
13. Conclusiones
14. Bibliografía
15. Glosario

Índice de imágenes

Índice de diagramas

Resumen

El presente trabajo ha tratado de reunir en una sola aplicación diversas apps de genética pertenecientes a César Benito Jiménez (profesor de la Facultad de Biología de la Universidad Complutense de Madrid) ya situadas en el mercado electrónico y construidas mediante MIT App Inventor, de las cuales se ha tenido que recoger la lógica incluida en MIT App Inventor y traducirla a la parte del servidor. Dicha aplicación se ha construido ofreciendo una parte de servidor hecha en tecnologías web basadas en Java Spring y que permite gestionar los distintos contenidos que se muestran al usuario final. Además, también se ha realizado la parte de cliente que es la aplicación móvil en sí y que reúne las distintas aplicaciones para que el usuario pueda interactuar con ellas, dicha aplicación móvil se ha construido con distintas tecnologías web como HTML5, CSS3 y JavaScript, siendo todo encapsulado con Cordova para permitir que fuese una aplicación multiplataforma (Android e IOs) y así disponer la aplicación al mayor número de usuarios posibles.

Palabras clave : Aplicación, Cordova, Genética, Java, Móvil, Multiplataforma, Servidor, Spring, UCM, Web.

Abstract

This project ...

Keywords : App, Cordova, Genetics, Java, Mobile, Multi-platform, Server, Spring, UCM, Web.

Introducción

La importancia de las aplicaciones móviles es innegable en la actualidad y todavía más en el ámbito de la educación, es por ello que 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 impartida dentro de la facultad, he aquí el propósito de este trabajo, aunar todas ellas en una única aplicación, permitiendo la utilización de todas ellas a todos los usuarios que quieran disponer de la aplicación, además de incorporar una plataforma web mediante la cual los usuarios administradores puedan gestionar y mantener distintas secciones relacionadas con la aplicación. Asimismo, se ha realizado de tal forma que gracias a los manuales recogidos en este documento se pueda extender la aplicación con más subaplicaciones.

Por otro lado, 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 las distintas etapas del proceso, así como las tareas requeridas en cada una de esas etapas.

Este trabajo consta de varias partes, en primer lugar, se exponen los requisitos hallados durante la fase de especificación de la aplicación.

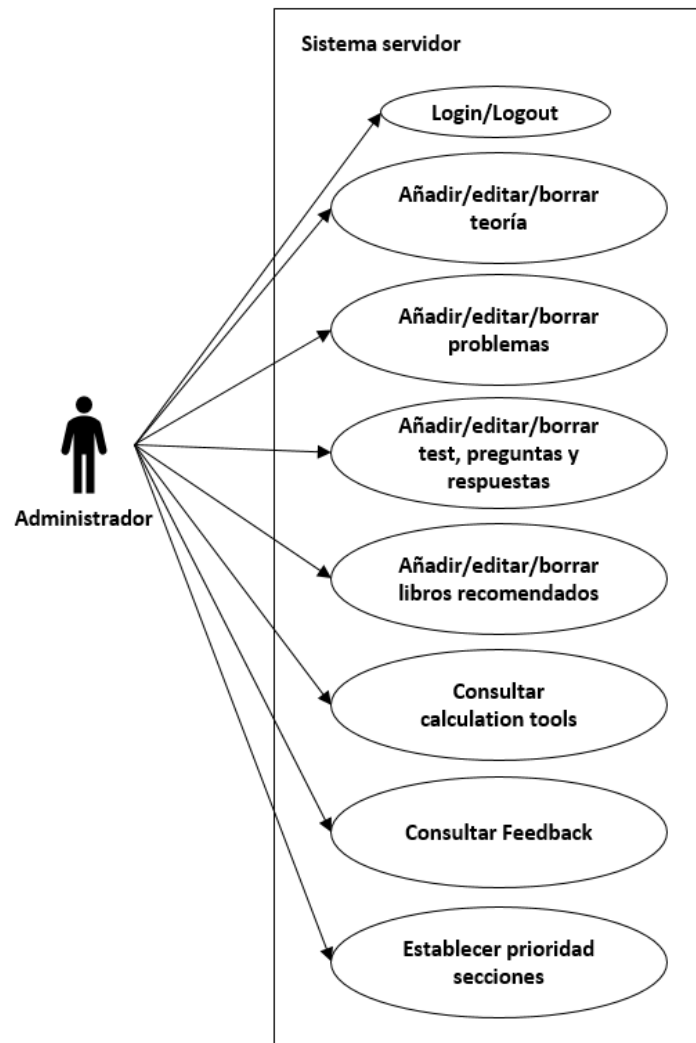
En segundo lugar, se presenta todo lo relacionado sobre el proceso de planificación y desarrollo del proyecto, donde se expone la metodología utilizada a la hora de realizar el proyecto.

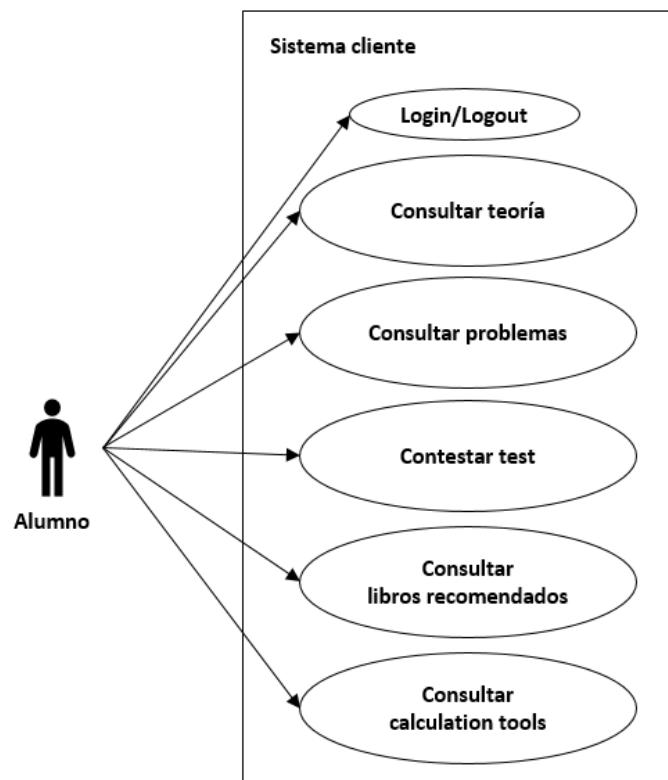
Seguidamente se incluye el conjunto de manuales necesarios para entender el correcto manejo y funcionamiento de la aplicación, así como otros manuales para permitir y facilitar la extensión tanto de la plataforma web como de la aplicación móvil.

Finalmente, se muestran los resultados obtenidos a partir de la realización del trabajo y las conclusiones asociadas a estos resultados, además de otras secciones como la bibliografía consultada o el glosario de términos.

Introduction

Arquitectura de la aplicación





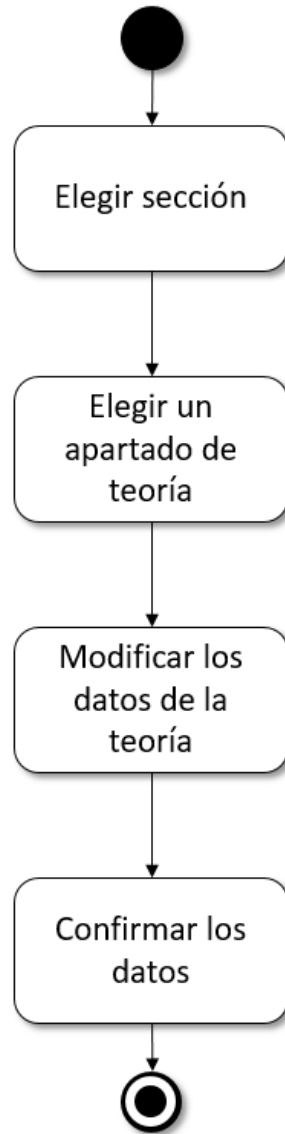


Diagrama de actividad: Editar teoría

Materiales y métodos

Especificación de requisitos

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.

Elegir sección: Muestra las distintas subsecciones disponibles dentro de cada sección. Estas subsecciones 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 Feedback: Muestra como feedback las distintas respuestas a los tests que han ido contestando los usuarios de la aplicación móvil.

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 ver 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.

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 en sí, tanto la parte servidor como la parte cliente, 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 y los motivos por los cuales se ha usado esa alternativa.

DESARROLLO DE LA APLICACIÓN DE ADMINISTRACIÓN (SERVIDOR)

Una vez pensada la arquitectura, el lenguaje y framework a utilizar para desarrollar la aplicación web del lado servidor, 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 [referencia bibliográfica al elemento nav que hemos usado?](#) el layout de la página que consideramos prioritario ya que es algo visible en todas nuestras páginas. Este layout consistía en una barra de navegación diseñada de manera responsive para evitar una mala visualización en navegadores móviles y un botón para cerrar sesión

(inicialmente deshabilitado hasta el momento en el que habilitamos la seguridad con usuario y contraseña). Podemos observar la versión final del layout en la imagen **X.X**.

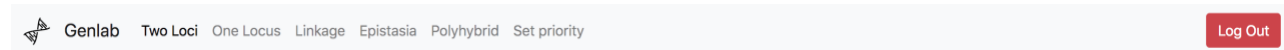


Imagen **X.X**. Layout de la aplicación.

Una vez teníamos listo nuestra aplicación en su forma mínima pasamos a diseñar el esquema de los datos, que podemos representar en dos diagramas. Uno de los diagramas, el mostrado en la imagen **X.X** es el modelo de dominio, en el que se representan todas las entidades que participan en la aplicación. El otro diagrama 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, que podemos ver en la imagen **X.X**.

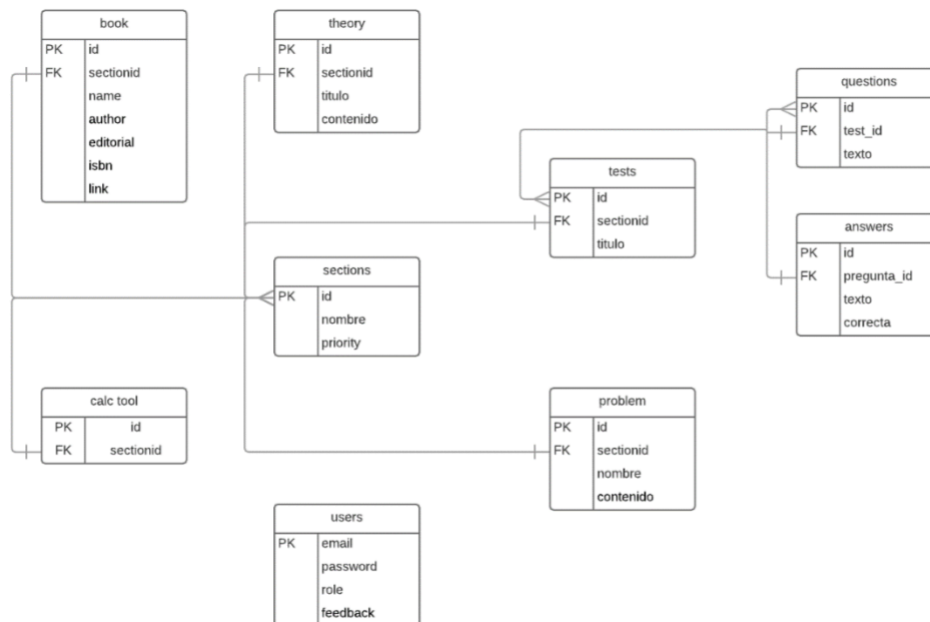


Imagen **X.X**. Modelo de dominio.

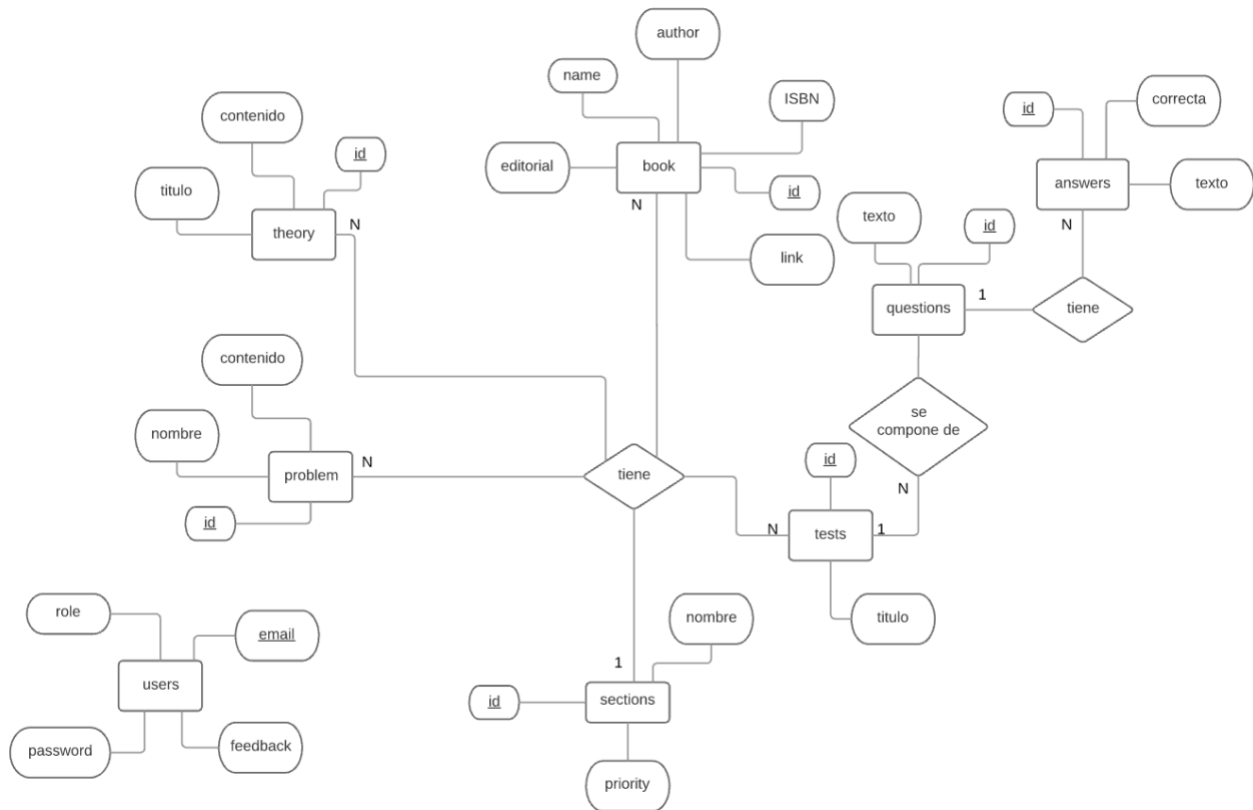


Imagen **X.X**. Diagrama Entidad-Relación de la Base de Datos

En la aplicación Java se crearon los objetos entidad equivalentes que, mediante anotaciones JPA, permitían encapsular los datos de la base de datos. En el caso excepcional de las Calculation Tools, en lugar de almacenarlas en la base de datos, están almacenadas en la propia aplicación (Estando su nombre e id en un enumerado y las operaciones que realizan en el propio servicio de aplicación).

Una vez diseñado el modelo de nuestros datos creamos los repositorios que extienden la clase *CrudRepository* parametrizada con la entidad que manejará y el tipo de dato que sea su id, proporcionando una serie de métodos con las operaciones básicas sobre la base de datos (e.g. *save*, *delete*, *findAll*, etc...) pudiendo crear más métodos nosotros como por ejemplo *findBySectionId*, al que pasaremos el section id y equivaldrá a una query del estilo:

```
select * from <table> where sectionid = <parámetro>
```

Después, desarrollamos los servicios, formados por pares de una interfaz y una clase que la implementa. Estos servicios cumplirán la función de conectar el controlador con los repositorios necesarios haciendo las operaciones necesarias para implementar las reglas de negocio. En el caso de las Calculation Tools aquí se realizan todas las operaciones necesarias, ya que no hay repositorios.

Además de todo lo anteriormente explicado, que ha sido desarrollado en Java, también hemos usado Javascript en la aplicación del servidor para ayudar a desarrollar:

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.

A estas alturas del desarrollo, **presentamos la aplicación a nuestro tutor (esto debería decirlo?)** para tener su opinión y conocer posibles avances a realizar sobre ella o modificaciones. Acordamos que debíamos realizar una iteración para añadir a la aplicación dos funcionalidades nuevas:

1. La obtención de feedback 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 implementa usando Javascript para hacerlo de una manera fácil (i.e.: con elementos arrastrables ordenándolos por prioridad de manera descendente)

Por último, añadimos la capa de seguridad usando Spring Security que redirigirá todos los intentos de acceso a cualquier endpoint al endpoint “/login”. Esta capa de seguridad no afectará a la API que usará la aplicación móvil, que desarrollamos más tarde.

Nuestra API exponía ciertos servicios al exterior para que desde la aplicación se pudiera acceder (pero no editar) al contenido de la aplicación.

DESARROLLO DE LA APLICACIÓN MÓVIL (CLIENTE)

Una vez estuvo pensada la arquitectura y estructura de la parte del servidor y aproximadamente por la mitad de su codificación, se comenzó a desarrollar la aplicación móvil que iba a servir de cliente para la aplicación. En primer lugar, nos reunimos con el profesor de la facultad de Biología César **apellido** para explicarle cómo iba el desarrollo de la aplicación en la parte del servidor y que nos

diera algunas pautas para proceder a desarrollar la aplicación móvil. Además, usamos como referencia las aplicaciones existentes para diseñar y establecer los parámetros con los que tenía que contar la aplicación móvil. Estas aplicaciones se encuentran disponibles en Google Play Store en el siguiente enlace [enlace](#).

Una vez tuvimos claro lo que tenía que tener la nueva aplicación, procedimos a su desarrollo. El primer problema con el que nos encontramos fue la multitud de dispositivos y sistemas operativos que cuentan los dispositivos móviles, por lo que al diseñar esta nueva aplicación teníamos que elegir una alternativa sencilla para poder desarrollar al mismo tiempo una aplicación que pudiera ser válida para cualquier sistema operativo de móvil (o al menos, que cubriese la gran mayoría de sistemas operativos más usados por los móviles) y, además, debido a las múltiples dimensiones de estos dispositivos, realizarlo de manera que se viera adecuadamente en todas las pantallas. Es por ello que decidimos emplear tecnologías web para llevar a cabo la codificación de la aplicación, usando los lenguajes HTML, CSS y JavaScript para ello, apoyado por Apache Cordova. Esta decisión fue tomada en base a lo explicado anteriormente, ya que 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, por lo que también arreglamos otro problema que era la comunicación de la aplicación con el servidor. Es por ello que, además de la aplicación de administración, se ha desarrollado una API en el servidor para que atienda estas peticiones. **Para mayor información sobre este apartado se explica en la parte de la memoria de la aplicación de administración (poner hipervínculo que lleve ahí).**

Una vez elegido el lenguaje y la plataforma, se procedió a su codificación al mismo tiempo que la API dedicada en el servidor. En primer lugar, se diseñó la página principal de la aplicación, donde iban a estar los distintos apartados con los que tenía que contar la aplicación, siendo estos apartados los problemas, los tests, la teoría, los libros recomendados y las calculation tools. Además, convertimos las cinco aplicaciones que había anteriormente en una sola, y cada aplicación anterior es ahora una sección dentro de la aplicación móvil, cada una con sus apartados y datos específicos. La vista principal se muestra en la imagen **X.X**

Imagen de la vista principal

El siguiente problema que surgió fue la forma de realizar la navegación entre las distintas secciones de la aplicación. Inicialmente, pensamos en tener distintas hojas HTML y comunicarlas mediante enlaces, pero nos dimos cuenta que al tener que cargar los estilos y los nuevos elementos cada vez

que el usuario pulsase una sección, no se conseguía la rapidez suficiente para que el usuario pudiera navegar cómoda y rápidamente entre secciones, por lo que se decidió utilizar el modelo SPA, contando con una única hoja HTML que se carga al iniciar la aplicación junto a sus estilos y los elementos de la misma se iban mostrando u ocultando en función de las acciones del usuario. De esta manera se conseguía que el cambio entre secciones y apartados fuese inmediato, mejorando la experiencia del usuario.

Posteriormente, y una vez terminada la API por el lado del servidor, se procedió a realizar y probar las peticiones AJAX que devolvían los datos de las distintas secciones y apartados. En este apartado tuvimos complicaciones al principio ya que por temas de seguridad no nos dejaba comunicarnos con la API del servidor desde aplicaciones externas, y para solucionarlo tuvimos que añadir un header específico en la cabecera de la URL para que permitiese este flujo de comunicación y de esta forma obtener los datos. Una vez solucionado este problema la comunicación con el servidor ya se realizaba correctamente y se mostraba en la aplicación los datos correspondientes a distintas secciones.

Una vez terminado lo anterior, se procedió a desarrollar el apartado de las calculation tools, donde nos surgió otro problema importante, el cual era que había en total treinta y tres calculation tools disponibles (incluso podría haber más en un futuro), y cada una tiene sus distintos elementos y estilos. Introducir todos estos elementos en el único HTML existente haría de este un archivo demasiado grande y muy difícil de mantener, por lo que había que buscar una manera de cargar estas calculation tools de otros archivos para facilitar los cambios y la mantenibilidad del sistema. Este fue uno de los problemas que más trabajo nos ha dado, ya que intentamos distintas soluciones para cargar ficheros HTML externos en otro y la mayor parte de ellos no funcionaban o no daban el resultado esperado. Finalmente, utilizamos una función de la librería JQuery que te permitía cargar y añadir elementos externos dentro de una etiqueta de HTML, por lo que el problema en principio estaba solucionado y ya cargaba las calculation tools perfectamente. Sin embargo, aunque parecía que todo funcionaba bien, resulta que al cargar varias veces las calculation tools el proceso de carga pasaba de ser inmediato a durar varios segundos e incluso minutos al cargar un determinado número de calculation tools en la misma ejecución. Finalmente, descubrimos que se debía a que en estos HTML externos cargábamos de nuevo los scripts y las hojas de estilos ya cargadas anteriormente, por lo que esto hacía que cada vez fuese más lento a medida que se iban cargando más documentos externos. Tras eliminar estas cargas en los ficheros externos, la aplicación ya funcionaba perfectamente y los tiempos de carga de las calculation tools son uniformes durante toda la ejecución.

La codificación de todas las Calculation tools y de toda la lógica detrás de las mismas y de las peticiones AJAX que pedían los resultados al servidor (no pusimos las fórmulas de cálculo en el cliente para reducir el consumo y el espacio que ocupa la aplicación en el dispositivo móvil, aspecto fundamental al tratarse de una aplicación orientada al móvil) fue sin duda lo que más tiempo nos ha

llevado de toda la aplicación, debido a su gran cantidad y complejidad. Una vez finalizadas, se llevaron a cabo una serie de pruebas para comprobar que los resultados eran los esperados y que se mostraban correctamente, y tras solucionar algunos errores derivados de estas pruebas, se terminó este apartado en la aplicación.

Por último, tuvimos que añadir una identificación de usuarios y un bloqueo de las secciones existentes, y que estas se desbloquearan cuando todos los tests de la sección anterior hubiesen sido completados.

Manuales

Manual de usuario

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

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 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. En esta pantalla además se nos muestra al igual que durante toda la navegación, un menú desplegable para seleccionar una aplicación (One Locus, Two Loci, Polyhybrid, Linkage o Epistasia) en concreto, dentro de la cual están las mismas secciones nombradas anteriormente.

[imagen main view]

[imagen main view con 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 lista CT]

Al pinchar sobre una de ellas se nos mostrará una pantalla con la herramienta de cálculo, dentro de la cual estarán 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 de una CT antes y después de darle a calcular]

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 de problemas]

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 listado 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 de un test con alguna pregunta respondida]

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.

[imagen de ejemplo de teoría]

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 libros recomendados]

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

A parte de lo tratado anteriormente, dentro del menú desplegable encontramos un botón de logout para cerrar la sesión de la app y un botón para volver atrás entre acciones.

[imagen conjunta logout y botón de atrás]

Manual del desarrollador

Manual de extensión de la aplicación móvil

En este apartado se procede a explicar todos los detalles de programación y la estructura que se ha seguido para programar la aplicación móvil, así como las instrucciones necesarias para llevar a cabo una ampliación o extensión de la aplicación.

Para llevar a cabo esta aplicación, se ha utilizado Apache Cordova. Esto es debido a que 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).

INSTALACIÓN DE PROGRAMAS NECESARIOS

Para instalar Apache Cordova en el ordenador, hay que seguir los siguientes pasos:

- Instalar Node.js, ya que esta aplicación se basa en su uso. Se puede descargar a través del siguiente enlace <https://nodejs.org/es/>


- Una vez instalado, es necesario tener un cliente de Git, ya que, aunque no es obligatorio su uso, el programa descarga algunos paquetes de esta plataforma al crear un nuevo proyecto. Puedes descargarlo a través del siguiente enlace <https://git-scm.com/downloads>
- Por último, para descargar el Cordova, hay que hacerlo mediante el instalador de paquetes de Node.js npm. Se realiza mediante línea de comandos, dependiendo del sistema operativo que use el ordenador. Por ejemplo, en el caso de Windows, el comando sería el siguiente, en la consola: `npm install -g cordova`

Tras esto, ya estará instalado Apache Cordova en el ordenador, y ya podremos ejecutar los comandos para crear los ejecutables de la aplicación para el dispositivo móvil.

Para hacer la depuración y la codificación más sencilla, también será necesario algún editor de texto especializado en tecnologías web (como Visual Studio Code) y un navegador actualizado para poder probar la aplicación (como Google Chrome). Esto, aunque muy recomendable, no es obligatorio.

ESTRUCTURA Y ARQUITECTURA DE LA APLICACIÓN

En este apartado se procede a explicar la estructura que se ha llevado a cabo para desarrollar la aplicación.

Como se ha citado anteriormente, para desarrollar la aplicación se ha usado Apache Cordova, por lo que la estructura inicial es la de un proyecto Cordova. En la imagen  se muestra la estructura del proyecto. Sin embargo, la carpeta donde se encuentra el código de la aplicación se encuentra en la carpeta www, el resto lo autogenera Cordova al crear el proyecto y al hacer el build de la aplicación.

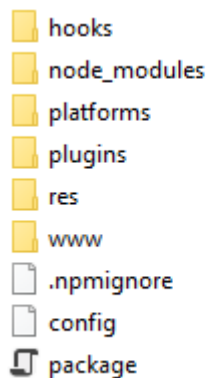



Imagen . Estructura del proyecto Cordova.

Dentro de la carpeta `www`, se encuentra el código HTML, CSS y JavaScript con el que se ha diseñado la aplicación, y cuya estructura de carpetas, representada en la imagen **X.X**, se procede a explicar a continuación:

- Por un lado, tenemos un único fichero HTML llamado *index.html*, el cual contiene todos los elementos de la aplicación. Solo hay un único archivo debido a que se ha decidido usar el modelo SPA (Single Page Application), explicado posteriormente en la arquitectura.
- Por otro lado, tenemos dos carpetas. Una de ellas, *resources*, contiene todos los recursos que usa la aplicación (dentro de *css* se encuentran las hojas de estilo propias, dentro de *bootstrap* las hojas de estilo de Bootstrap descargadas de su página web, dentro de *img* las imágenes y, por último, dentro de *js* tanto los scripts propios como scripts de JQuery, que también se ha utilizado para manejar distintos aspectos en la aplicación). La otra carpeta, llamada *ctools*, contiene, a su vez, otras carpetas correspondientes a cada sección de la aplicación, y, en su interior, ficheros HTML donde se encuentran definidas las calculation tools de cada aplicación en concreto (se explican más detalles sobre esto en la extensión de la aplicación).

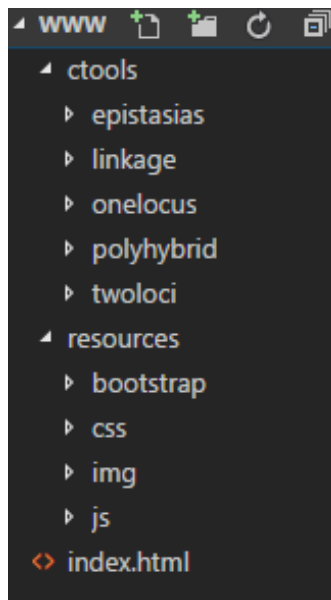


Imagen **X.X**. Estructura de la carpeta `www` que contiene el código de la app

En cuanto a la arquitectura, se basa principalmente en una programación orientada a eventos, usando el modelo SPA para que de esta manera la carga de la página y de sus recursos se realice una única vez al iniciar la aplicación, reduciendo 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.

Asimismo, para recoger los datos se ha usado una API creada en el lado del servidor, donde a través de peticiones AJAX el servidor manda los datos requeridos por la aplicación (como los problemas, los tests, el cálculo de las calculation tools...), para así reducir el consumo de memoria en la aplicación móvil y tener la capacidad de modificar estos datos sin tener que descargar de nuevo la aplicación.

EXTENSIÓN 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, sin embargo, se desea añadir una nueva calculation tool, ya que es necesario seguir una serie de pasos adicionales para facilitar la mantenibilidad y la modularización de las mismas.

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.

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 problema y sin ningún tipo de limitación, respetando la estructura de diseño proporcionada para permitir que nuevos desarrolladores puedan modificar la aplicación.

En el caso de las calculation tools, el proceso a seguir es distinto, ya que al usar un modelo SPA y, debido a las múltiples calculation tools de las que se dispone actualmente o incluso de las que se puede disponer en un futuro, y, además, teniendo cada una 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, por lo que se ha decidido cargar su contenido desde ficheros HTML externos que se insertan en el HTML principal (de esta manera se evita la carga de recursos adicionales, insertando solo el código HTML de las calculation tools, ya que las hojas de estilo se han cargado anteriormente).

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 la misma es completamente libre y se puede añadir todo lo necesario. Tan solo hay que tener en cuenta lo siguiente:
 - 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).
 - 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 id 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 **X.X**. 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.

```
<!-- CTOOLS -->
<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 **X.X**. 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 incluir lo siguiente:
 - 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 cabecera de la petición (en el *beforeSend* de la petición AJAX) debe colocarse un elemento **Como se llama a esto????** 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 **X.X** 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 **X.X** Petición AJAX de una calculation tool.

EJECUCIÓN DE LA APLICACIÓN

Para ejecutar la aplicación, se puede hacer de dos maneras: una de ellas es descargando la aplicación en un dispositivo móvil o en un emulador mediante los comandos de Apache Cordova *cordova build OS* y *cordova run OS* (siendo OS el sistema operativo deseado), o bien ejecutando el archivo *index.html* en un navegador para que de esta forma muestre su contenido.

Si estas usando un navegador, es posible que no cargue las calculation tools ya que algunos navegadores bloquean la carga de ficheros externos por seguridad. En Google Chrome, para deshabilitar esto, es necesario ejecutar el siguiente comando:

`.\chrome --allow-file-access-from-files path`

Siendo path la ruta donde se encuentra el fichero *index.html* en el sistema de ficheros. Al navegar por la aplicación tras ejecutar este comando, las calculation tools cargarán sin ningún problema.

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. Los detalles de su implementación (véase lenguajes de implementación, frameworks y patrones de diseño utilizados) están recogidos en la sección [<link y referencia>](#).

Para comenzar a desarrollar este proyecto, debemos instalar en nuestro ordenador una versión de Java igual o superior a la 1.8 ([referencia a 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, una librería de Java que permite simplificar al máximo los objetos que contienen los datos de nuestra aplicación. ([referencia a https://projectlombok.org/download](#))

Seguidamente, importaremos el proyecto al IDE del que estemos haciendo uso.

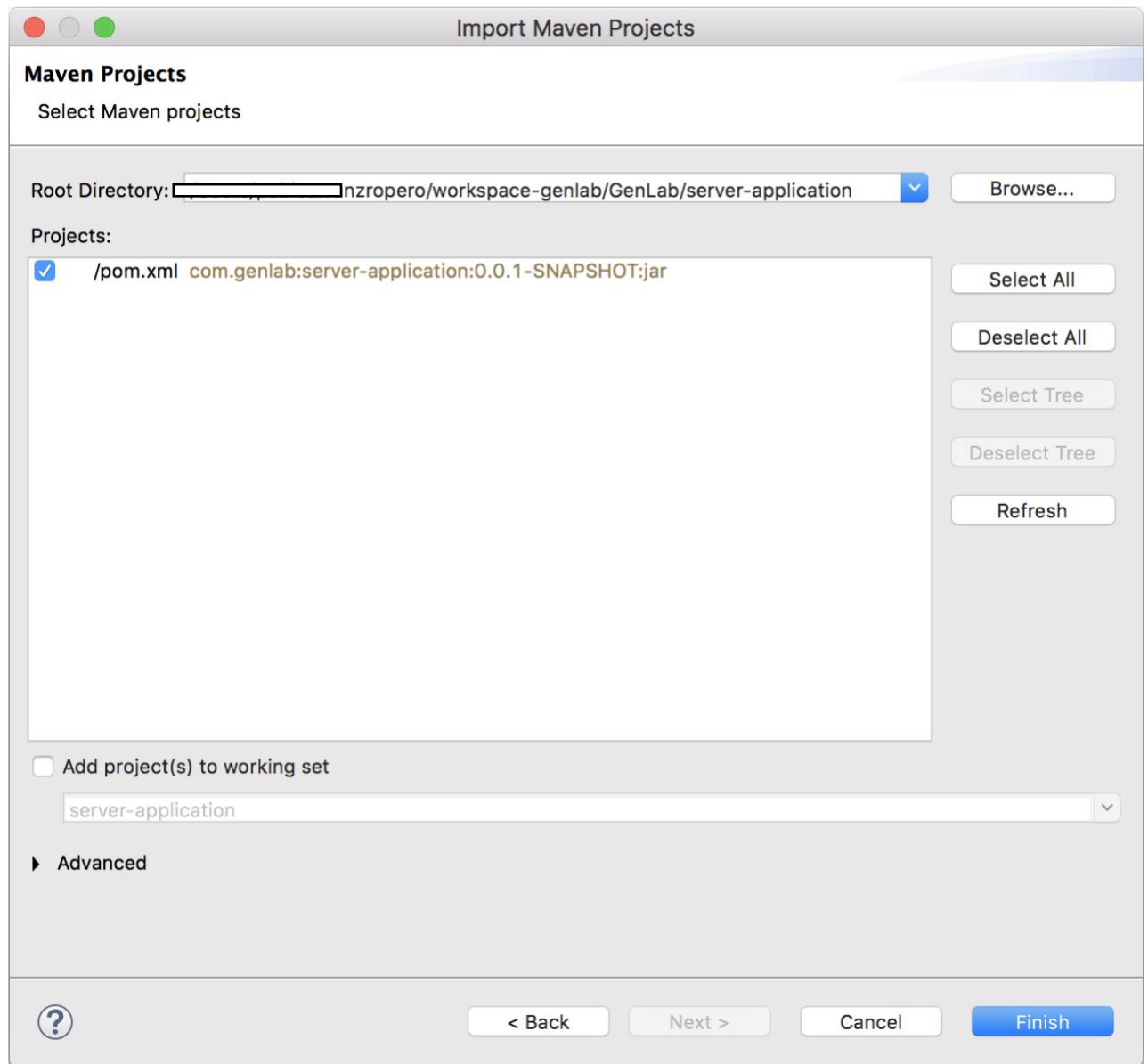


Ilustración x.x: Importación de proyecto a STS

El proyecto está desarrollado siguiendo el patrón MVC y una arquitectura multicapa (referencia a donde expliquemos patrones usados) por lo que encontraremos el proyecto distribuido, a grandes rasgos, de la siguiente manera:

- **Controllers:** Contiene todos los manejadores de rutas de nuestra aplicación web y de la API REST. Realiza la parte de Controlador, como su nombre indica, en el patrón MVC y forma parte de la capa de presentación.
- **Models:** Contiene todas las entidades y objetos de datos de la aplicación. Realiza la parte de Modelo en el patrón MVC y forma parte de la capa de datos y de la capa de vista.

- **Repositories:** Contiene los repositorios de la aplicación, es decir, son las clases que accederán a la base de datos cuando sea necesario. Forma parte del Modelo en el patrón MVC y 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 del Modelo en el patrón MVC y de la capa de negocio en nuestra arquitectura.
- **Config, Interceptor y Utils:** Estos paquetes contienen utilidades necesarias para la configuración o para la intercepción de peticiones HTTP.
- **Resources:** Contiene las plantillas creadas en HTML, utilizando Thymeleaf (referencia). También contiene los archivos CSS y JavaScript necesarios para la construcción de la vista. Forma parte de la Vista en el patrón MVC y de la capa de presentación en nuestra arquitectura.

Añadiendo una nueva sección a nuestra aplicación

En el momento de la publicación de este proyecto, las secciones existentes son las descritas en (sección y referencia). Para añadir una nueva sección en nuestra aplicación será tan sencillo como 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 (sección y referencia). 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 mapeará 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 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

Al publicar nuestra aplicación las Calculation tools de las que dispone la misma son las enumeradas en **(sección)**. 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 (comosellame) ¿cuya contraseña queda en conocimiento de (quien sea)? Para añadir nuevos usuarios con dicho rol a la base de datos. Debemos 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/>

Apéndices

Resultados y discusión

En este apartado se muestran los distintos resultados obtenidos como parte del desarrollo del proyecto, dichos resultados en esencia son dos aplicaciones, una aplicación web creada a partir de Java Spring junto con Thymeleaf y otras tecnologías web como HTML o CSS, y que funciona como gestor de la parte del servidor y donde la cual el usuario administrador pueda gestionar el contenido de las distintas secciones que quiera mostrar al usuario final de la aplicación móvil, además podrá ver feedback de los resultados recogidos de los distintos usuarios que usen la aplicación cliente.

[imagen de la aplicación servidor]

La segunda aplicación es la aplicación móvil, creada a través de tecnologías web (HTML, Javascript y CSS) y encapsulada con Apache Cordova para permitir que sea multiplataforma (permitiendo así llegar tanto a clientes de Android como de IOS), y que es la encargada de la parte cliente dentro de la relación cliente-servidor establecida durante la planificación del proyecto. Dicha aplicación se ocupa de mostrar la distinta información relevante para el correcto entendimiento de la materia por parte del usuario final, todo ello mediante la total interacción del usuario con la aplicación.

[imagen de la aplicación móvil]

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.

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

También forma parte de los resultados obtenidos la base de datos MySQL, necesaria para mantener la consistencia del contenido de las distintas subsecciones de la aplicación además de la información de los usuarios de la aplicación.

Otros resultados obtenidos son los artefactos creados a lo largo del desarrollo del proyecto, estos son la especificación de requisitos y el mockup interactivo de la aplicación realizado con MyBalsamiq.

Conclusiones

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 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).
3. ...

Trabajo para el futuro

A parte de lo realizado durante el proyecto 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 feedback 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.

- Incrementar el número de secciones (aplicaciones) según las necesidades de la Facultad de Biología, esto se podría realizar gracias al seguimiento de las instrucciones recogidas en este documento para ello. (Véase x.y)
- Solucionar posibles fallos (bugs) que pudiese contener la aplicación, esto podría solucionarse con una etapa de testeo antes de subir 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.

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>)

Glosario, siglas y acrónimos

API : Application Programming Interface.

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.

Facultad de Informática – Universidad Complutense de Madrid

UCM: Universidad Complutense de Madrid.

WYSIWYG: What You See Is What You Get.