Universidad Zaragoza
1542

Trabajo Fin de Grado

Sistema de información con aplicación de realidad aumentada para la manipulación de tableros de tareas basados en tarjetas

Autor

Guillermo Pérez García

Director

Rubén Béjar Hernández

Escuela de Ingeniería y Arquitectura

Junio de 2016

Este trabajo está dedicado a mis padres, que me apoyaron constantemente durante mis largos años de carrera, y a mis abuelos, que siempre tuvieron la ilusión de verme acabar la universidad y fallecieron la semana anterior a la entrega del proyecto.

Documento de autoría

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

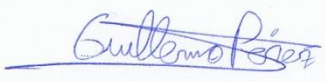
D./Dª. Guillermo Pérez García,

con nº de DNI 73.005.651-V en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) grado, (Título del Trabajo) Sistema de información con aplicación de realidad aumentada para la manipulación de tableros de tareas basados en tarjetas,

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 20 de junio de 2016



Fdo: Guillermo Pérez

Sistema de información con aplicación de realidad aumentada para la manipulación de tableros de tareas basados en tarjetas

Resumen

Este proyecto tiene como objetivo el desarrollo de un sistema de información para el uso de metodologías ágiles con tecnologías de realidad aumentada. De este modo se pretende conseguir un funcionamiento más cercano e intuitivo de las mismas.

Para ello se creará una aplicación móvil en la que se mostrará información del sistema superpuesta a la imagen real que capte la cámara.

Partiendo de unos prototipos iniciales con funcionalidades básicas, su desarrollo ha ido evolucionando a lo largo de dos iteraciones, con el fin de dotarlas de las funcionalidades necesarias y complementarias entre ellas para poder gestionar un sistema de estas características.

- Tabla de contenido

1 Introducción

El presente documento detalla el proceso de realización del Trabajo Fin de Grado titulado "Sistema de información con aplicación de realidad aumentada para la manipulación de tableros de tareas basados en tarjetas".

1.1 Contexto del proyecto

Este proyecto se ha realizado entre los meses de Febrero y Junio de 2016, período durante el cual formé parte del IAAA (Grupo de Sistemas de Información Avanzados) como colaborador. Todo el trabajo mostrado a lo largo de esta memoria ha sido elaborado íntegramente por mí, y he contado con la ayuda del profesor Rubén Béjar Hernández, que me ha guiado durante el proceso.

1.2 Contexto tecnológico

Desde la fase inicial de análisis se orientó este proyecto hacia un funcionamiento multiplataforma, basado en Web y en una aplicación móvil.

Para la parte de desarrollo web se emplearon tecnologías habituales como HTML, CSS y Javascript, que en los últimos años han aumentado su popularidad por la aparición de frameworks que aprovechan cada vez mejor su potencial.

La aplicación móvil está basada en Unity3D [4], un entorno de desarrollo en el que profundizaremos más adelante, y se ha empleado un plugin de realidad aumentada llamado Vuforia [1] para implementar la funcionalidad de reconocimiento de marcadores gráficos en cualquier soporte (pantallas digitales o impresos en papel).

En los distintos componentes que forman el 'backend' de este proyecto (servidor, consola de administración y base de datos) se han empleado tecnologías más conocidas como Java y MySQL. Para implementar una API basada en servicios web RESTful se empleó Jax-RS, junto con Grizzly [7] para crear el servidor y Glassfish [8] para el despliegue, JUnit para la realización de tests, y Gradle [6] para la gestión de dependencias.

1.3 Objetivo

El objetivo central de este proyecto ha sido desarrollar un sistema de información para gestionar metodologías ágiles, y que de este modo se ayude a descubrir la facilidad de uso de las mismas y concienciar sobre la gran utilidad que pueden tener en la actualidad.

1.4 Motivación

El punto de principal interés del proyecto a nivel personal era aprender a manejar tecnologías que son ajenas al plan de estudios del Grado en Ingeniería Informática, como Unity y C#.

Meses antes de iniciar este proyecto, y gracias a algunas asignaturas de esta titulación, descubrí la utilidad de herramientas para el control de versiones de código (git, sourcetree, etc), para la gestión y planificación de tareas en grupo (Trello) y para el desarrollo en grupo de documentación y diagramas (Google docs, Cacoo online). Gracias a mi experiencia más que satisfactoria con herramientas como las anteriores, decidí crear un proyecto relacionado con ellas.

1.5 Estructura del documento

Esta memoria de proyecto consta de seis apartados y cuatro anexos.

El primer capítulo explica el contexto personal y tecnológico del proyecto al lector. El segundo explica el problema que se quiere solucionar o mitigar, y de qué manera se ha abordado la solución elegida.

Los apartados tercero y cuarto explican detalles técnicos del proyecto, correspondientes a las fases de diseño e implementación. En ellos se explica la arquitectura final que presenta el sistema, las tecnologías y herramientas empleadas para el desarrollo y los problemas más importantes que se encontraron a lo largo del tiempo que duró este proyecto.

El apartado quinto está dedicado a la gestión del proyecto, y el último a la valoración personal y a explicar posibles mejoras que se podrían estudiar de cara al futuro.

Los anexos aportan información adicional a la memoria, y abarcan estos temas:

1. Anexo I - manual de instalación: explica cómo desplegar el servidor, los clientes y la base de datos.
2. Anexo II – manual de usuario: pasos a seguir para manejar correctamente los distintos clientes.
3. Anexo III – interfaz del servidor REST: describe la función de cada servicio que consumen los clientes, los datos que requiere y los datos que envía como respuesta.
4. Anexo IV – Casos de uso extendidos: describe detalladamente los casos de uso de los distintos clientes, las precondiciones, postcondiciones, los pasos que se deben seguir para realizar la función correctamente y los posibles casos de error.

2 Análisis del problema

2.1 Descripción

Las metodologías ágiles tienen como objetivo mejorar la gestión del control de tareas y la coordinación entre equipos de trabajo. Algunas metodologías de este tipo son Scrum, Kanban y XP (programación extrema).

Dentro de este marco existe el sistema Kanban, ideado por Toyota hace más de 30 años para manejar las necesidades de material en la cadena de producción. Esta técnica de gestión de las tareas es intuitiva y permite explorar de un solo vistazo el estado de varias tareas en un mismo momento, y analizar el progreso de las mismas. La palabra 'Kanban' en japonés significa 'tarjeta visual' y describe el pilar básico de esta técnica: tareas descritas de manera breve y concisa (generalmente se emplea el formato Post-it) colocadas en tableros compartidos, que se subdividen en las diferentes columnas, que representan las etapas o estados del proyecto.

Empresas con un alto nivel de cultura tecnológica ya emplean estos métodos en sus distintas variantes, pero algunas herramientas aparentan ser más complejas de lo que realmente son, y ello provoca que otras empresas decidan seguir con sus metodologías de trabajo actuales aunque no sean las mejores ni más eficientes.

Este Trabajo Fin de Grado pretende acercar las metodologías de tipo Kanban a empresas que hasta la fecha son reticentes a implantar este tipo de técnicas, creando un sistema de información que pueda ser empleado desde una vista web y desde una aplicación móvil con funcionalidad de realidad aumentada. De este modo se agiliza el uso de recursos compartidos, tanto físicos como virtuales, entre varios usuarios al mismo tiempo. Los tableros virtuales se pueden consultar normalmente desde la aplicación web y desde la app, que proporciona una vista global similar a la de la web, o emplear las funcionalidades de AR junto con la cámara del móvil para leer tableros físicos compartidos como pizarras, tablones de corcho o cualquier superficie sobre la que se puedan colocar los marcadores asociados a las distintas etapas del proyecto, como las paredes de una sala de reuniones.

2.1.1 Soluciones existentes

Microsoft Visual Studio permite registrar de manera completa y exhaustiva las tareas de un proyecto y asignarles distintas propiedades como su estado actual (activas, propuestas o cerradas), su descripción, fechas de inicio y fin, conteo de horas acumuladas, etc. Sin embargo no permite ver las tareas de una manera simple, rápida e intuitiva.

La herramienta gratuita Trello permite crear tareas que también pueden llegar a tener información muy completa, y las descripciones pueden ser muy detalladas llegando a contener listas de tareas con checkbox para indicar el progreso, o fechas de vencimiento, pero por defecto se crean tarjetas con la mínima información necesaria para poder entender rápidamente el estado de un proyecto.

2.2 Requisitos

Las etapas, 'stages' o columnas representan las distintas etapas del flujo de trabajo. Las tarjetas representan unidades de trabajo, una tarea a realizar por un equipo. Una tarjeta está contenida en una única columna, y se puede mover de una a otra columna en función del progreso de la realización de dicha tarea.

Requisitos funcionales de la vista web

RF01: los usuarios de la aplicación web podrán crear columnas.

RF02: los usuarios de la aplicación web podrán eliminar una columna con sus tarjetas.

RF03: los usuarios de la aplicación web podrán crear una tarjeta dentro de cualquier columna.

RF04: los usuarios de la aplicación web podrán eliminar una tarjeta.

RF05: los usuarios de la aplicación web podrán mover tarjetas de una a otra columna.

RF06: los usuarios de la aplicación web podrán acceder a la vista global de un tablero, sus etapas y sus tareas.

Requisitos funcionales de la app

RF07: el usuario de la aplicación móvil podrá acceder a la vista de un tablero, sus etapas y tareas.

RF08: el usuario de la aplicación móvil podrá ver la información completa de una tarea.

RF09: los usuarios de la aplicación móvil podrán eliminar tareas.

RF10: los usuarios de la aplicación móvil podrán ver las tareas de una columna/etapa leyendo el marcador correspondiente con la cámara del dispositivo.

Requisitos funcionales de la consola de administración

RF11: los usuarios de la consola de administrador podrán crear usuarios nuevos.

RF12: los usuarios de la consola de administrador podrán eliminar cualquier usuario existente.

RF13: los usuarios de la consola de administrador podrán modificar datos personales de cualquier usuario.

RF14: los usuarios de la consola de administrador podrán crear nuevos equipos de trabajo.

RF15: los usuarios de la consola de administrador podrán eliminar equipos de trabajo existentes.

RF16: los usuarios de la consola de administrador podrán dar permiso de acceso a un tablero a un equipo de trabajo.

RF17: los usuarios de la consola de administrador podrán revocar el permiso de acceso de un equipo de trabajo a un tablero.

RF18: los usuarios de la consola de administrador podrán crear tableros.

RF19: los usuarios de la consola de administrador podrán eliminar tableros.

RF20: los usuarios de la consola de administrador podrán crear etapas dentro de cada tablero.

RF21: los usuarios de la consola de administrador podrán eliminar etapas.

RF22: los usuarios de la consola de administrador podrán crear tareas dentro de cada etapa.

RF23: los usuarios de la consola de administrador podrán eliminar tareas.

Requisitos no funcionales

RNF1: la aplicación web debe poder utilizarse en navegadores web de PC y Mac.

RNF2: la aplicación móvil debe poderse utilizar en dispositivos Android.

RNF3: la consola de administrador debe poderse utilizar en cualquier computador a través de la máquina virtual Java.

2.3 Casos de uso

A continuación se muestra el diagrama con los casos de uso que satisfacen todos los requisitos funcionales anteriormente mencionados. También se indican los actores que pueden acceder a cada caso.

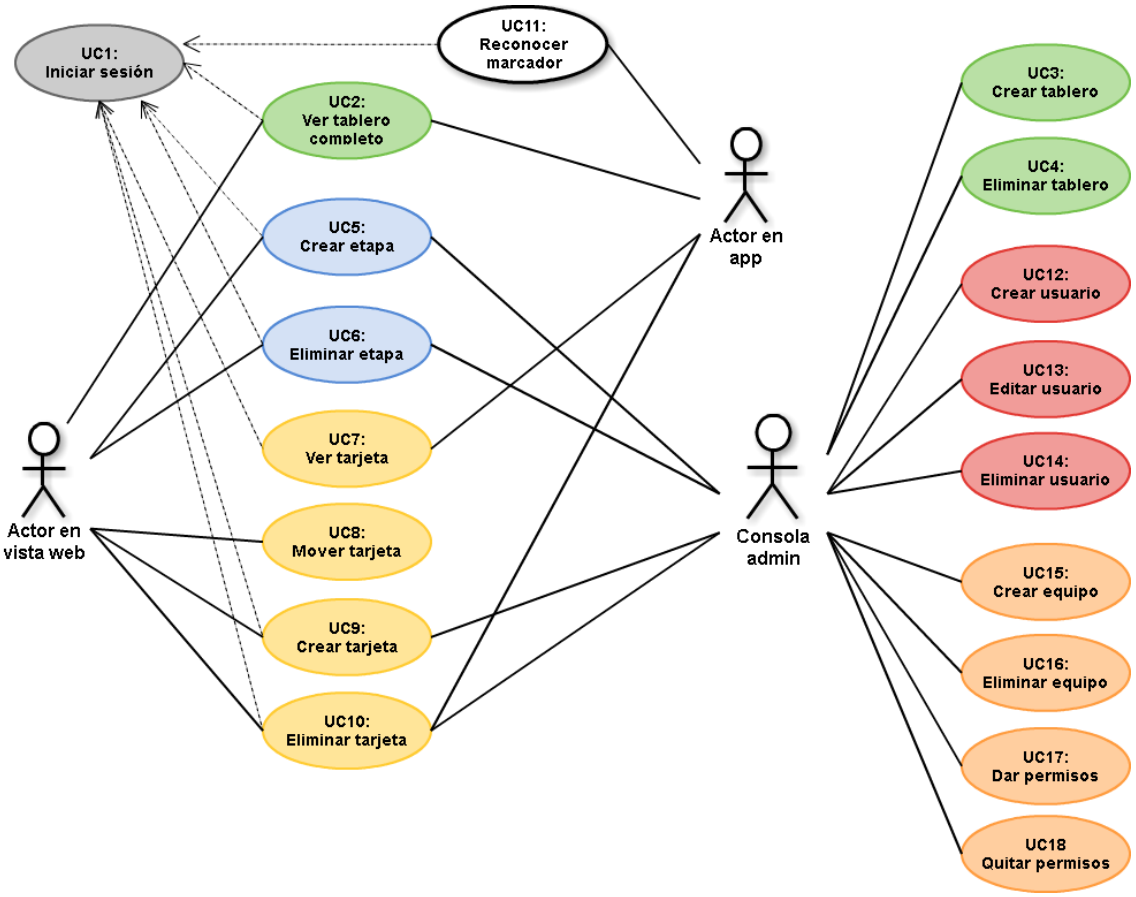


Figura 1: Diagrama de casos de uso

El siguiente mapa de navegación muestra la ejecución de la aplicación móvil con los distintos casos de uso a los que se puede acceder desde la app.

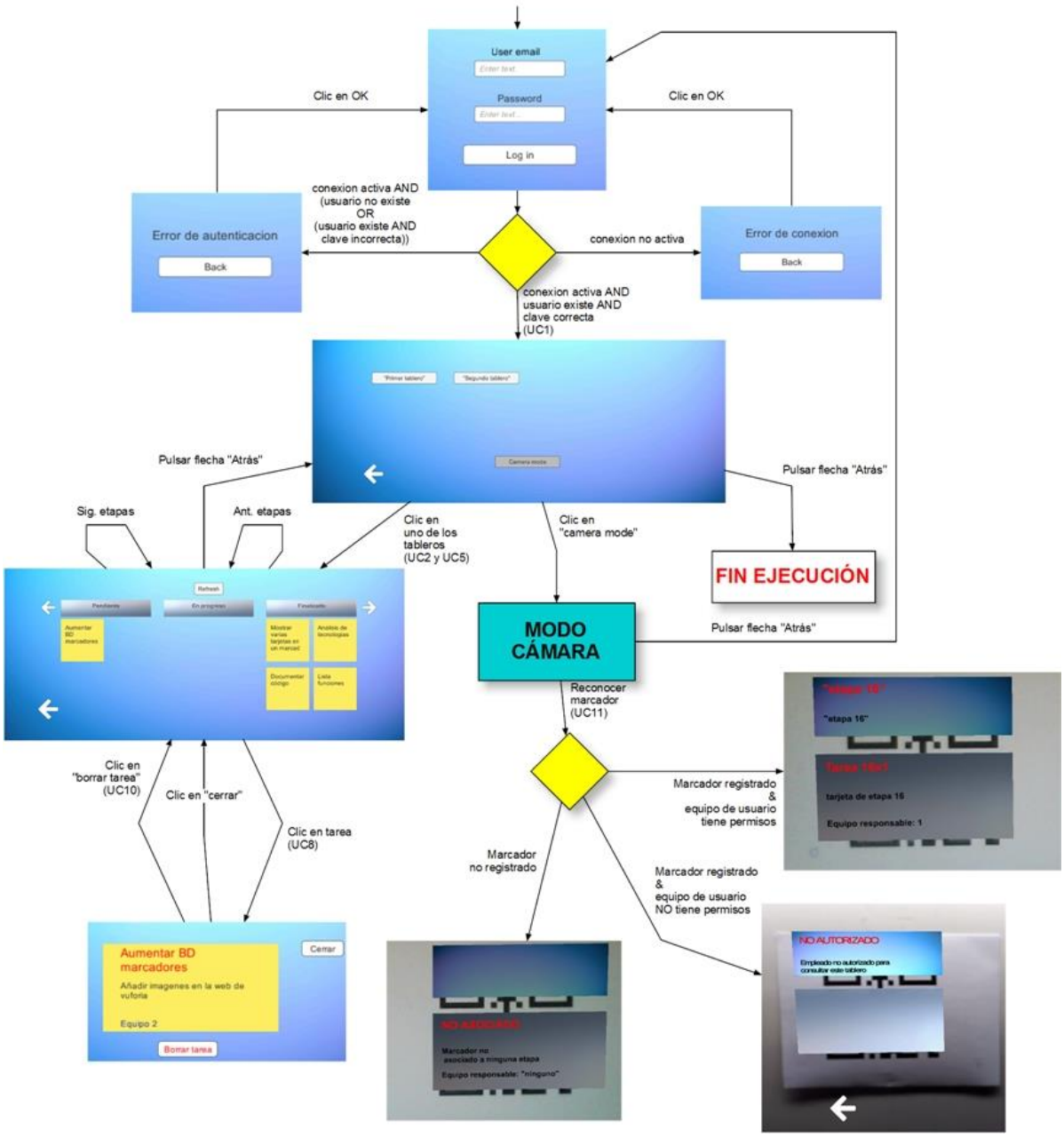


Figura 2: Mapa de navegación de la app móvil

La navegación en la vista web tiene la misma distribución. Sin embargo, el estilo gráfico es diferente al de la app y no se puede acceder a la función de reconocer marcadores (caso de uso 11).

2.4 Selección de tecnologías

2.4.1 Plataforma Unity3D

Se ha elegido esta plataforma por su compatibilidad con dispositivos Android e iOS, aunque la mayoría de las pruebas se vayan a realizar en Android y entorno de escritorio. Más detalles en [4].

2.4.2 Plugins de realidad aumentada

Se escogió el uso de Vuforia [1] por su extensa documentación existente, la gran ayuda que aporta el foro de dudas de usuarios (con un estilo similar a Stack Overflow y el foro oficial de Unity 3D) y la experiencia adquirida en el manejo de esta herramienta en los meses anteriores al comienzo del proyecto.

Se estudió al inicio del proyecto emplear el plugin ARToolkit [5] para este mismo propósito, pero el conjunto de Assets proporcionado por esta librería no era tan completo ni tan útil como el de Vuforia, su manejo para pruebas simples era mucho menos intuitivo, y la creación de marcadores era mucho más complicada: era necesario editar cada imagen manualmente para añadir un marco negro alrededor de cada marcador, el cual debía tener unas proporciones determinadas. Por el contrario, Vuforia admite cualquier imagen como "target" y, aunque haya que crear una base de datos a través de su portal web, el manejo de cada Gameobject en el entorno Unity es más sencillo.

2.4.3 Librería Cardboard / GoogleVR para uso de la app con gafas

Tras múltiples pruebas de integración del proyecto con los elementos proporcionados por Google en su librería GoogleVR (anteriormente denominada "Cardboard") para Android, iOS y Unity, se decidió no seguir adelante por esta vía. Los problemas de compatibilidad con el plugin Vuforia no tenían solución aparente, e impedían generar correctamente el archivo .apk para prueba en dispositivos móviles.

2.4.4 Códigos QR como marcadores de realidad aumentada

Se analizaron distintas fuentes de imágenes con licencia Creative Commons 0 para ser empleadas como marcadores, pero resultó difícil encontrar un conjunto amplio de las mismas que cumpliera las pautas recomendadas para que el reconocimiento de marcadores fuera fiable. Algunos de estos requisitos eran: no tener partes difuminadas, mostrar vértices bien definidos y tener colores bien diferenciados.

Finalmente se decidió usar las imágenes más simples que cumplen esos requisitos: códigos QR [9] de color blanco y negro, con gran definición (600x600), y formados por figuras geométricas básicas con vértices fácilmente distinguibles.

3 Diseño de la solución

3.1 Arquitectura del sistema

El sistema se diseñó con una arquitectura de tipo Cliente/Servidor en tres niveles: los distintos clientes, el servidor que procesa las peticiones de todos ellos, y la base de datos sobre la que se consultan y modifican los datos.

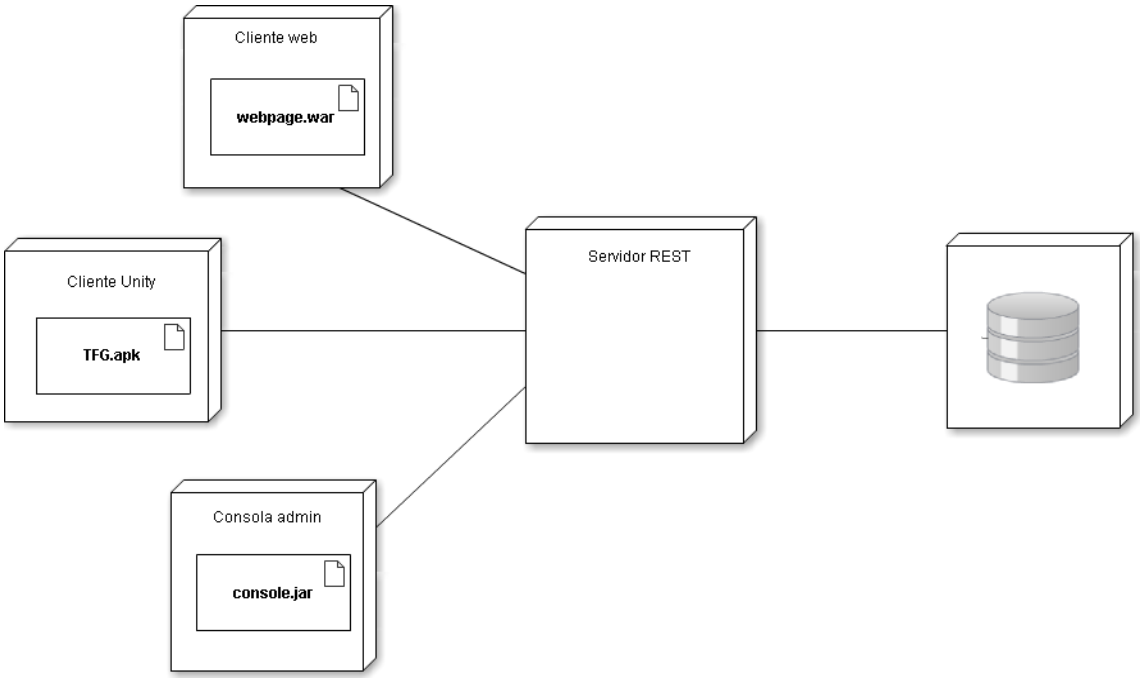


Figura 3: Diagrama de distribución del sistema

3.1.1 Cliente app

El elemento diferenciador de este proyecto se desarrolló en Unity3D para Android. Ofrece una visión simple y reducida de los tableros, etapas de trabajo y tareas almacenadas en el sistema, y permite reconocer marcadores impresos o mostrados en pantallas gracias al plugin de realidad aumentada Vuforia [1].

La app emplea los datos obtenidos al iniciar sesión para permitir o denegar el acceso a los distintos tableros almacenados. Si al leer un marcador autorizado (asociado a una columna/etapa que pertenece a un tablero autorizado), se muestra una cabecera con el título y descripción de la etapa, y sus tareas debajo. Al leer marcadores asociados a tableros no autorizados muestra un mensaje indicando que no se tiene permiso de acceso.

En el siguiente diagrama de paquetes se muestra la estructura de la aplicación Unity y las relaciones entre los distintos componentes que la forman.

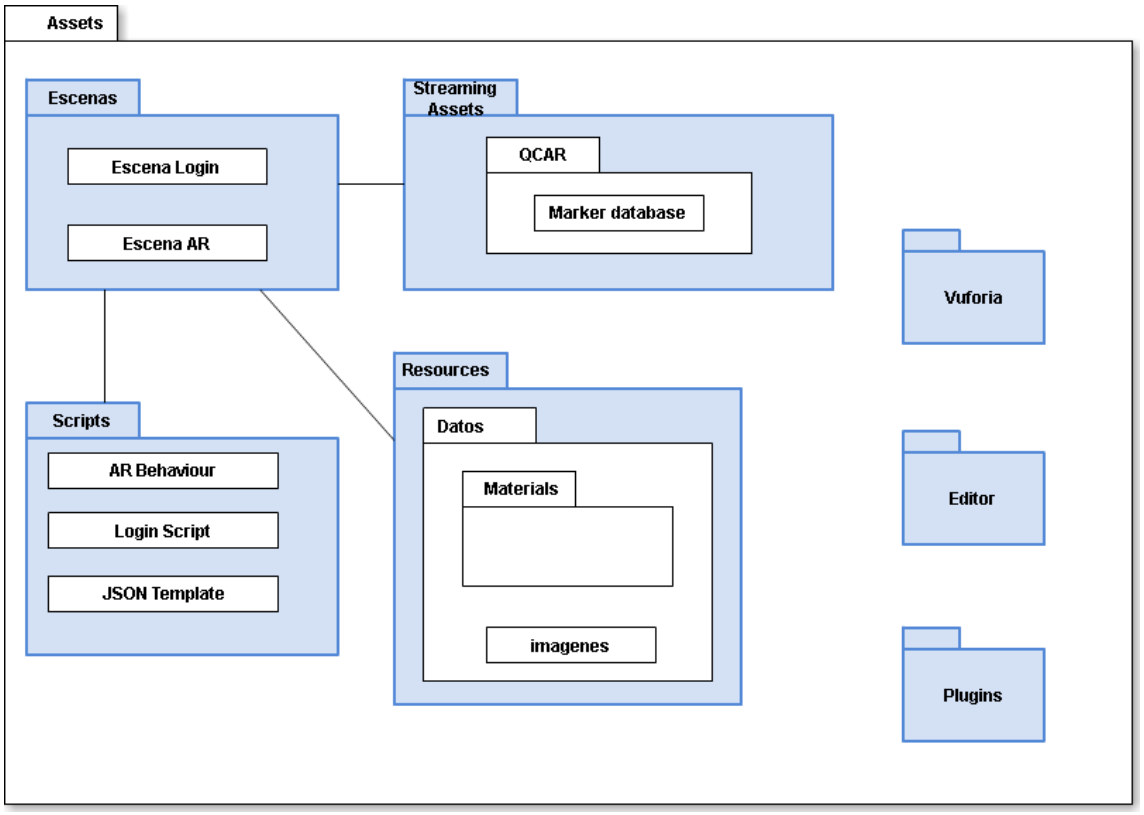


Figura 4: Diagrama de paquetes de Unity3D

3.1.2 Cliente web

Se desarrolló un cliente para navegador web con tecnología HTML, CSS y Javascript para ver de manera más detallada la información almacenada en el sistema. Permite crear etapas y tareas dentro de cada tablero, y mover tareas de una etapa a otra conforme vaya avanzando el trabajo que representan.

La estructura de columnas se diseñó pensando en mostrar la evolución del trabajo a lo largo del tiempo y el estado de cada tarea. No obstante, cada empresa lo puede adaptar a sus necesidades que más le convengan. Otra distribución válida consistiría en asignar una columna a las tareas de un único equipo o subgrupo de trabajo, pero de ese modo ya no se seguirían estrictamente las directrices que definen el estilo Kanban.

3.1.3 Consola de administración

En la parte final del proyecto se decidió crear una pequeña aplicación en Java y con interfaz de usuario Swing para permitir al administrador del sistema introducir, modificar o eliminar información de un modo más directo, sin tener que emplear la aplicación móvil ni el navegador web.

Mediante la vista web se permite crear y eliminar elementos básicos como tareas y etapas, pero hay otras tareas que requieren permisos más elevados. Cuando un nuevo empleado entra a trabajar en una empresa, las tareas de crear sus nuevas cuentas de usuario, su perfil en la intranet, etc. no las puede realizar cualquier empleado.

Mediante esta herramienta, el administrador podrá realizar operaciones de creación y eliminación sobre las cinco entidades que forman este sistema: tableros, etapas, tareas, usuarios

y equipos. También autorizar el acceso o revocarlo a uno o más equipos, y crear o eliminar empleados y equipos.

3.1.4 Servidor

El servidor fue el primer elemento del proyecto que se decidió desarrollar. Se eligió el lenguaje Java por la experiencia adquirida a lo largo de la carrera, ya que en la mayor parte de las asignaturas era el lenguaje más utilizado. En el pasado también se emplearon las librerías Jax-RS, que permiten crear clientes y servidores REST, y el framework Grizzly [7] para crear el servidor y Oracle GlassFish [8] para el despliegue del servidor. Se emplearon adicionalmente JUnit para pruebas y Gradle [6] para manejo de dependencias.

Las primeras versiones del servidor fueron muy básicas e implementaban las funciones CRUD básicas: leer, crear, actualizar y eliminar información de las entidades que forman el sistema. Paralelamente, se crearon los tests de JUnit correspondientes para asegurar que el código creado cumplía su propósito.

A lo largo de las semanas se fue perfeccionando y se añadieron funcionalidades necesarias pero de menor prioridad, como el inicio de sesión y la creación de usuarios. Estas dos funcionalidades se crearon recordando uno de los conceptos más repetidos en la carrera: las contraseñas nunca deben guardarse a plena vista ni en una base de datos sin encriptación, de modo que se empleó un método básico de encriptación de claves aprendido en la asignatura Seguridad Informática.

Todas las rutas (endpoints) de la API se describen en el anexo III.

El siguiente diagrama de paquetes muestra la estructura de paquetes y ficheros del servidor, y las relaciones entre ellos a alto nivel:

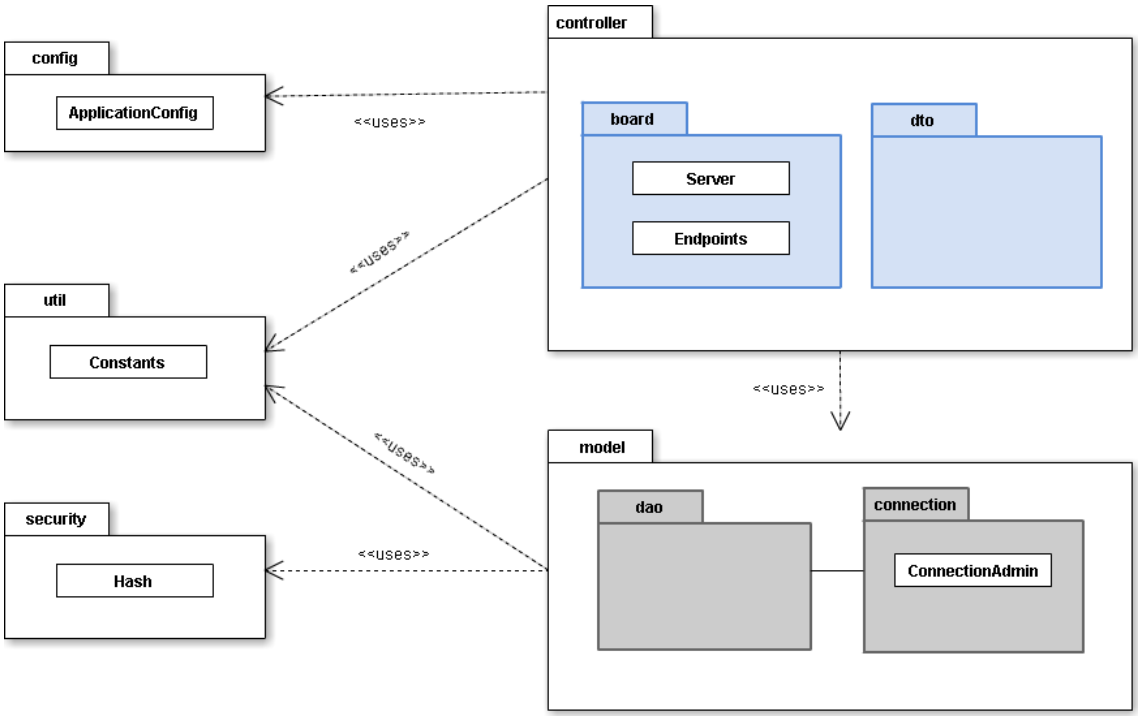


Figura 5: Diagrama de paquetes Java

En el diseño arquitectural del sistema se ha empleado el estilo Modelo – Vista – Controlador.

- Los distintos tipos de cliente contienen el código necesario para enviar peticiones al servidor y visualizar la información recibida.
- El servidor gestiona del mismo modo las peticiones de los tres distintos clientes. En su desarrollo se ha empleado el patrón Data Access Object para separar las distintas acciones en cinco clases, una por cada entidad que forma el modelo de base de datos. Cada una de esas clases contiene el código para crear, modificar o eliminar únicamente la entidad que le corresponde. Además, la clase ConnectionAdmin proporciona la información de acceso a la base de datos para que los distintos objetos DAO puedan consultar o enviar datos.
- La base de datos se puede ubicar en la misma máquina que el servidor, o puede emplearse un gestor de bases de datos situado en una máquina externa. El diseño de las distintas tablas se explica en el siguiente apartado.

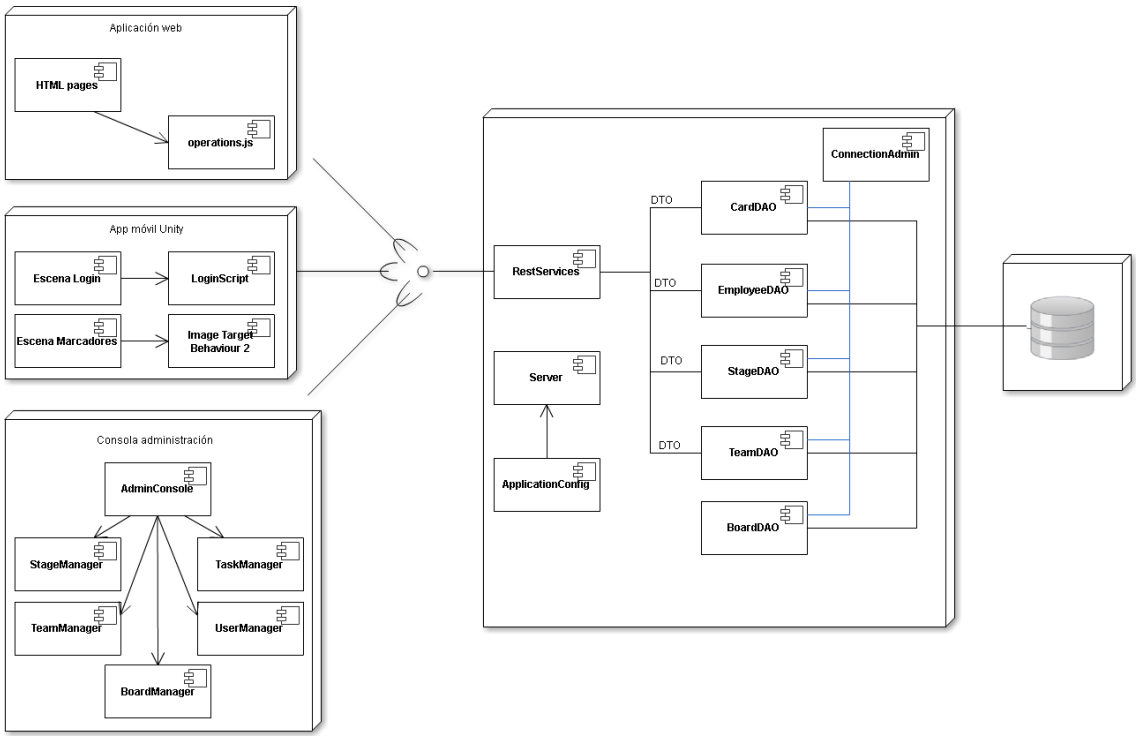


Figura 6: Diagrama de despliegue / Componentes y conectores

3.2 Modelo de datos

El modelo de datos se diseñó en la fase inicial del proyecto, previamente al comienzo del desarrollo del servidor, para poder crear a partir de dicho modelo todos los endpoints necesarios de la API REST.

Para poder iniciar sesión se almacenan las contraseñas encriptadas mediante la clase Hash de Java. Los permisos de cada usuario van ligados al equipo al que pertenece (uno y sólo uno).

Todos los empleados de un equipo tienen los mismos permisos, y no se distinguen rangos ni categorías.

Este modelo se diseñó con los campos habituales que se suelen almacenar de cada entidad, y se podría ampliar en función de las necesidades que surjan. Por ejemplo, almacenar en cada empleado su categoría salarial, su fecha de entrada en la empresa, o una referencia a otro empleado: su supervisor.

Las distintas entidades del sistema están representadas en tablas individuales de la base de datos. Todas constan al menos de un identificador numérico autoincremental, que se asigna en el momento de la inserción por el propio gestor de BD, y un campo “name”. Algunas entidades incluyen el campo “descripción”, para dar más detalle de la información que representan, y las columnas (etapas) y tareas además incluyen un campo numérico llamado “position” para ordenarlas en la vista web y la aplicación.

Las contraseñas de cada usuario son campos de texto alfanumérico que se encriptan previamente a la inserción, de modo que aunque se consulte el valor de dicho campo no se obtendrá la contraseña del usuario. El método de comprobación de contraseñas en el momento de iniciar sesión vuelve a encriptar la introducida por el usuario, la compara con la almacenada y devuelve cierto o falso en función de si coinciden o no.

La tabla “permission” representa los permisos de acceso de cada equipo a los distintos tableros almacenados en el sistema. Si hay un registro que permita a un equipo acceder a un tablero, cualquier usuario perteneciente a ese equipo podrá acceder a ese tablero desde los distintos clientes (web y app).

En el esquema se aprecia la jerarquía existente entre tableros, etapas y tarjetas (tareas). No es posible crear una etapa en un tablero que no exista, ni una tarea en una columna (etapa) inexistente. Del mismo modo, a la hora de borrar cualquiera de las tres entidades, se ha implementado un borrado en cascada para eliminar toda la información deseada con una sola llamada.

Las entidades se detallan a continuación en el diagrama E/R exportado directamente del modelo desarrollado con MySQL Workbench.

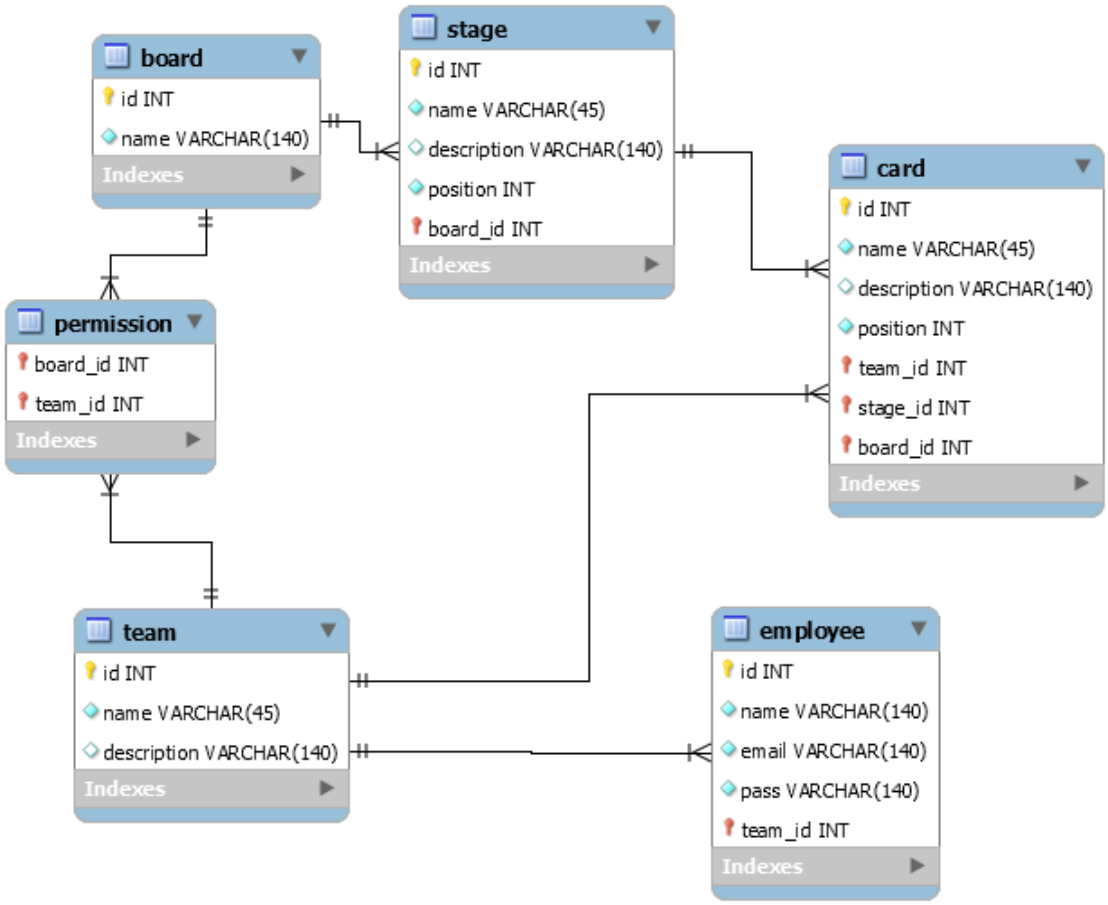


Figura 7: Diagrama del modelo de datos de la BD

4 Implementación

El proyecto está compuesto por tres clientes de distintos tipos, un servidor que gestiona las peticiones de los clientes, y una base de datos en la que se almacena toda la información del sistema.

4.1 Tecnologías empleadas

A continuación se describen las distintas tecnologías que se han empleado en el desarrollo de este proyecto.

4.1.1 Unity3D

Unity [4] es un motor de videojuegos con plataforma de desarrollo propia que permite crear juegos para dispositivos móviles, ordenadores y videoconsolas, y está disponible para Microsoft Windows, OS X y Linux. También permite crear aplicaciones para integrarlas con Google Cardboard, Samsung GearVR, SteamVR, Oculus Rift y Microsoft Hololens.

Esta herramienta ofrece un completo entorno para desarrollar contenido en tres dimensiones, con posibilidades de ampliación mediante la tienda de Assets: plugins, modelos 3D, sonidos, proyectos completos, extensiones, texturas, etc. El foro de la comunidad es muy activo y ayuda a solucionar la gran mayoría de problemas concretos o dudas que suelen aparecer durante la elaboración de proyectos de este tipo.

En cuanto al manejo de scripts, Unity soporta los lenguajes Javascript y C#, e incluye una versión propia de MonoDevelop para crear y editar scripts desde la versión 3.0.

4.1.2 C#

Es un lenguaje orientado a objetos, basado en .NET y desarrollado por Microsoft. Mantiene la sintaxis habitual de C y es muy similar a Java.

Para crear scripts asociados a las escenas y objetos de Unity se programa en este lenguaje mediante el editor de texto MonoDevelop (incluido en el entorno de desarrollo Unity3D). No dispone de funciones como compilación de código y ejecución, ya que estas funciones se realizan en el editor de Unity. Se compila y se muestran los errores (si los hay) cada vez que se guardan cambios en cualquier fichero, y se ejecuta desde el reproductor integrado en el editor cada vez que el usuario pulsa el botón “Play” en la parte superior central. También se recompila el código y muestran los errores existentes al crear un build para el sistema operativo deseado (.apk en este caso para Android).

4.1.3 Grizzly

Project Grizzly [7] es un entorno de desarrollo para aplicaciones Java que permite crear servidores web escalables y robustos. Mediante distintos módulos permite utilizar WebSockets, servicios web con Jax-WS y manejo de eventos con baja latencia gracias a Comet.

Se ha empleado junto con Gradle [6] para manejar las dependencias de la aplicación y JUnit para la realización de pruebas en cada endpoint existente.

4.1.4 Tecnologías web habituales

Las páginas necesarias para visualizar el contenido en formato web se han desarrollado en HTML con hojas de estilos CSS y funciones adicionales en Javascript.

4.1.5 MySQL

El modelo final de base de datos se ejecuta sobre MySQL Workbench, una herramienta potente que permite manejar desde pequeñas bases de datos locales para aplicaciones caseras hasta conexiones a máquinas remotas con esquemas y tablas de mayor tamaño.

4.2 Pruebas

Los métodos creados para probar las distintas rutas del servidor se encuentran en la clase BoardServiceTest, y se agruparon por método (GET, POST, PUT, DELETE) y por entidad involucrada.

En los métodos GET se comprobaba que al enviar un identificador numérico existente en la base de datos, se obtenía la entidad correspondiente (tablero, etapa, tarea, empleado o equipo) en formato JSON y con el código HTTP 200 apropiado. Para el caso contrario, al enviar un identificador que no correspondía a ninguna entidad, se comprobaba que la respuesta recibida contenía el código de error 404 NOT FOUND.

Ejemplo de método para testear la ruta /board/team/:

```
@Test
public void getTeam() throws IOException {

  Client client = ClientBuilder.newClient();
  Response response = client.target(address + ":" + port +
"/board/team/1").request(MediaType.APPLICATION_JSON).get();
  assertEquals(200, response.getStatus());
  assertEquals(MediaType.APPLICATION_JSON_TYPE,
response.getMediaType());
  Team leido = response.readEntity(Team.class);
  System.out.println("Equipo leido: " + leido.getID() + ", " +
leido.getName());
}
```

Para testear los métodos POST se creaban entidades con datos de prueba (los campos correspondientes a cada tabla) y se enviaban en formato JSON al servidor. Si el código HTTP de la respuesta era 201, la inserción había sido correcta. Si por el contrario había sucedido algún error, el código devuelto era 400 (BAD REQUEST).

Ejemplo de código dedicado a probar peticiones POST en /board/card/:

```
@Test
public void createCard() throws IOException {

   Client client = ClientBuilder.newClient();
   Card c = new Card();
   c.setName("Lista funciones");
   c.setDescription("Definir los modos de E/S");
   c.setPosition(1);    c.setStageID(1);
   c.setTeamID(1);      c.setBoardID(1);
   Response response = client.target(address + ":" + port +
"/board/card").request(MediaType.APPLICATION_JSON).post(
Entity.entity(c, MediaType.APPLICATION_JSON));
   assertEquals(201, response.getStatus());
   assertEquals(MediaType.APPLICATION_JSON_TYPE,
response.getMediaType());
```

```
    Card leido = response.readEntity(Card.class);
    System.out.println("Tarjeta leida: " + leido.getID() + ", " +
leido.getName());
}
```

Para la actualización de datos en el servidor se empleaba el método PUT. En este caso se enviaba información de manera muy similar a los métodos POST, pero en este caso también se incluye el ID de la entidad que se quiere modificar. De esta manera se cambia el valor actual de todos los campos por los valores de la entidad enviada en la petición. Se modifican todos los campos menos el identificador, que se utilizaba para la búsqueda. El código que indica que la modificación se ha realizado correctamente es 200 OK, y el código de error es 400 BAD REQUEST.

Ejemplo de petición PUT en la ruta /board/employee/:

```
@Test
public void updateEmployee() throws IOException {

    Employee e = new Employee();
    e.setID(10);
    e.setName("Prueba");
    e.setEmail("Prueba");
    e.setPass("nuevapass");
    e.setTeamID(2);

    Client client = ClientBuilder.newClient();
    Response response = client.target(address + ":" + port +
"/board/employee/10").request(MediaType.APPLICATION_JSON).put(
Entity.entity(e, MediaType.APPLICATION_JSON));
    assertEquals(200, response.getStatus());
    assertEquals(MediaType.APPLICATION_JSON_TYPE,
response.getMediaType());
    Employee upd = response.readEntity(Employee.class);
    assertEquals(upd.getName(), e.getName());
    assertEquals(upd.getID(), e.getID());
}
```

La eliminación de datos en el servidor se realiza mediante el envío de peticiones DELETE a las mismas rutas ya definidas. Se emplea únicamente el identificador numérico de la entidad, que se envía como parámetro en la propia ruta.

El código que indica que la petición se ha atendido correctamente es 204 NO CONTENT, y si no se ha podido realizar la eliminación se devuelve el código 404 NOT FOUND.

```
@Test
public void deleteStage() throws IOException {

    Client client = ClientBuilder.newClient();
    Response response = client.target(address + ":" + port +
"/board/stage/11").request().delete();
    assertEquals(204, response.getStatus());
}
```

4.3 Herramientas de desarrollo

Las herramientas empleadas fueron:

- Dispositivos móviles para pruebas: Sony Xperia M2 y BQ Aquaris E5s
- Unity: Entorno de desarrollo Unity 5.2.0 x86
- Java: IntelliJ IDEA Community Edition
- Base de datos: MySQL Workbench
- Elaboración de la memoria: Microsoft Office 2013
- Elaboración de diagramas: Cacoo y EDGE Diagrammer
- Gestión de tareas pendientes: Trello

4.4 Problemas y soluciones

4.4.1 Migración de Vuforia 4.2 a 5.5

Cuando empezó la implementación de la aplicación en Unity, a mediados de marzo, se utilizó la versión 4.2.9 de Vuforia para reconocimiento de marcadores en una fase inicial de pruebas. Semanas más tarde, cuando se quiso modificar el proyecto para que la aplicación se pudiera usar en modo normal o con gafas (Cardboard, específicas, etc.), se descubrió que sólo las versiones 5.0 y posteriores soportaban los distintos tipos de visión estereoscópica: Optical See-through y Video See-through.

Al seguir los pasos especificados por Vuforia en su guía de migración entre versiones [3] la cámara dejó de funcionar. No fue posible solucionar este problema recuperando versiones anteriores, ni eliminando los archivos (denominados "Assets") de Vuforia y volviendo a importarlos desde cero en el mismo proyecto, de modo que tras varias semanas sin ningún resultado se decidió crear un proyecto Unity3D nuevo y exportar los elementos ya creados que no tenían relación con Vuforia (scripts, imágenes, escenas, base de datos de marcadores, etc.). Una vez creado el nuevo proyecto, se importó la versión 5.5.9 de Vuforia y se comprobó que la cámara AR y los marcadores volvían a funcionar correctamente.

En este proceso aparentemente fácil de solucionar se invirtieron 3-4 semanas. Se pudo seguir desarrollando la parte web y la aplicación, pero las pruebas sólo se pudieron realizar con el reproductor integrado en el editor de Unity y realizando las peticiones REST al servidor en local, de modo que no se pudo determinar si el servicio funcionaba con IPs externas hasta que se arregló el problema explicado.

4.4.2 Manejo de "bases de datos" de marcadores Vuforia

La herramienta Vuforia requiere crear una "base de datos" local con el conjunto de marcadores que se deseen emplear en una aplicación. El término "base de datos" no se ajusta al que se maneja comúnmente en informática, ya que se trata de un archivo .unitypackage que las contiene en un formato propio de Vuforia para ser importado posteriormente en Unity como un "custom package".

Si se desea modificar una base de datos de marcadores es necesario borrar manualmente todos los archivos existentes de la versión anterior, editar los archivos XML de configuración propios de Unity en los que se hace mención a la base de datos, descargar la nueva versión de la BD e importarla como la primera vez. Si existen en el proyecto dos versiones de la misma base de datos al mismo tiempo, es posible que no funcionen los marcadores de ninguna de las versiones.

Este proceso se tuvo que repetir varias veces a lo largo del proyecto, ya que inicialmente se utilizaron a modo de prueba varias imágenes de un conocido videojuego que tenían copyright, y posteriormente se decidieron usar códigos QR [9]. Cada vez que se quiso ampliar el número de marcadores fue necesario borrar la anterior versión e importar la nueva.

Este formato propio de Vuforia también impidió el uso de conjuntos de marcadores mucho más amplios, ya que debe especificarse un conjunto finito de imágenes. Con una base de datos remota en la que se almacenaran imágenes y una referencia asociada, se podría ampliar el sistema en gran medida.

4.4.3 Problemas de conexión cliente-servidor

Las pruebas de conexión con IP externa sólo se pudieron hacer durante las horas de desarrollo realizadas en el edificio Ada Byron, dado que cada equipo ya tiene una dirección externa asociada y las redes domésticas en muchos casos no permiten el acceso externo aunque se abran ciertos puertos manualmente. En este caso se usó la dirección http://lanuza.cps.unizar.es, asignada a uno de los equipos del laboratorio 2.10.

4.4.4 Depuración en Unity (plataforma móvil)

La depuración de errores en aplicaciones móviles Android añadió un punto de complejidad, ya que sólo se puede acceder a los registros (logs) activando el modo depuración en el dispositivo y utilizando Android Studio. En muchos casos fue más sencillo crear archivos de texto manualmente y escribir las trazas de ejecución o fallo manualmente en los puntos del código que nos interesen.

5 Gestión del proyecto

A continuación se proporciona la documentación que refleja cómo se ha organizado el proyecto a lo largo del tiempo. También se especifican todas las herramientas y configuraciones empleadas para desarrollar el proyecto.

5.1 Planificación

La estimación de horas realizada al inicio del proyecto se repartió en una fase inicial, dos iteraciones de desarrollo, una fase de pruebas para comprobar el sistema una vez finalizado, y una fase final para realización de diagramas y memoria.

El cronograma descrito en la propuesta de proyecto constaba de:

- Fase previa
  - Definición de requisitos y estudio de tecnologías necesarias.
  - Selección del plugin de AR.
  - Diseño preliminar.
- Primera iteración
  - Prototipo de aplicación Unity3D que integre funcionalidad AR.
  - Modelo de datos, API web, cliente web sencillo.
- Segunda iteración
  - App definitiva Unity3D que integre funcionalidad AR.
- Documentación técnica y memoria del proyecto.

En la segunda iteración, además de lo inicialmente planificado, se completó el cliente web, se añadieron más funcionalidades al servidor y se desarrolló la consola de administrador.

5.2 Control de esfuerzos

Desde el principio del proyecto se llevó un registro de horas de trabajo invertidas en cada parte del trabajo. Para ello se utilizó una hoja de cálculo Excel, en la que se dividió el trabajo en seis grandes apartados:

- **Base de datos**: 7h
- **Servidor**: 50h
- **Cliente web**: 95h (~25% del total)
- **Cliente app móvil**: 140h (~35% del total)
- **Consola de administración**: 12h
- **Memoria**: 70h
  - **Total estimado**: 390h

También existen categorías con un menor número de horas dedicadas, como el análisis de tecnologías o las reuniones de proyecto.

Para una mejor legibilidad, se han agrupado las tareas del mismo tipo por colores. Las tareas verdes corresponden a la aplicación Unity, las moradas al desarrollo del servidor y las amarillas al cliente web.

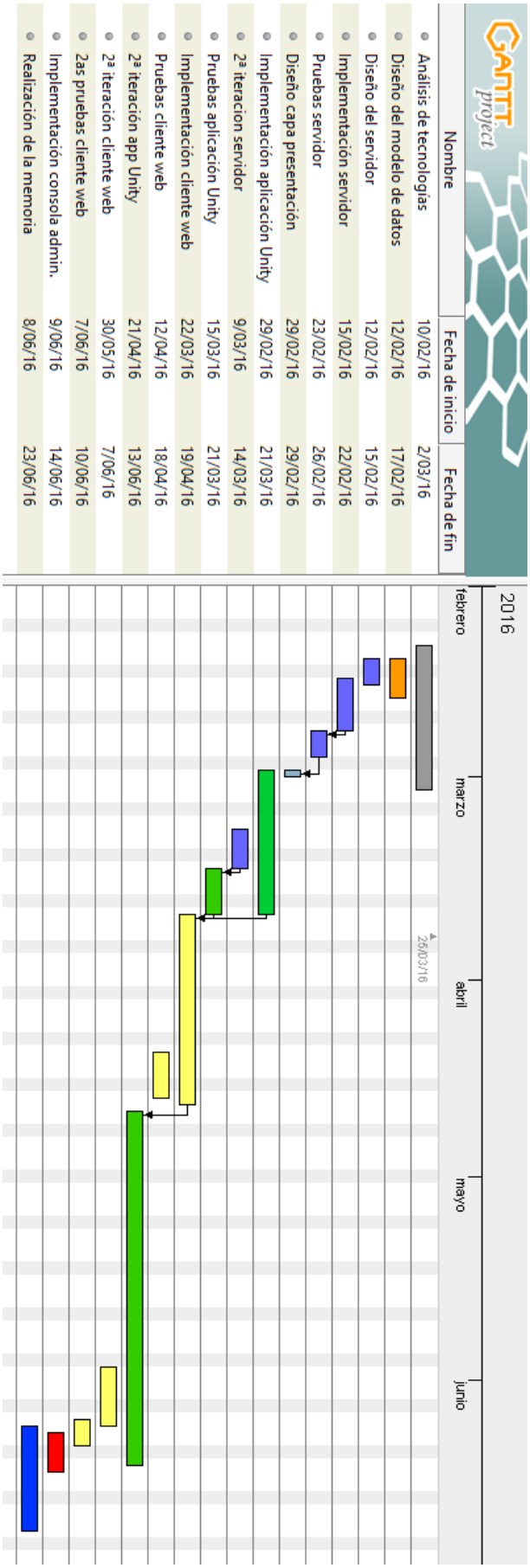


Figura 8: Diagrama de Gantt

El servidor y el cliente móvil se realizaron en distintas iteraciones.

- El servidor inicial se desarrolló con las funciones básicas necesarias para poder probar los clientes conforme se implementaban. Posteriormente se añadieron funciones más elaboradas como el login de usuario, la inserción y la eliminación de etapas o tareas desde los distintos clientes, y la comprobación de permisos de usuario.
- El cliente móvil se creó inicialmente con el modo cámara de AR para reconocer marcadores, y posteriormente se añadió la interfaz gráfica para mostrar tableros completos sin emplear la cámara.

5.3 Gestión de configuraciones

Los documentos, diagramas, paquetes de Unity y Vuforia, scripts de base de datos y demás archivos del proyecto se almacenaron en Google Drive.

El código del servidor y de la aplicación web se almacenó en un repositorio de BitBucket [11], y el código de la app para Android en otro repositorio distinto de ese mismo servicio de control de versiones [10]. La herramienta de escritorio empleada para subir los cambios al repositorio fue SourceTree.

No fue necesario crear ramas en ningún repositorio porque el proyecto se desarrolló en solitario, pero sí hubo que crear un repositorio nuevo y empezar casi desde cero a primeros de mayo, puesto que por la migración de Vuforia anteriormente mencionada varios componentes de Unity y Vuforia dejaron de funcionar. Al copiar el código de Unity a un proyecto nuevo y limpio, e importar la versión 5.5 de Vuforia desde cero, el proyecto volvió a funcionar.

- **Unity3D** 5.2.0f3 x86
- **Vuforia** 5.5.9 para Unity
- **IntelliJ** IDEA Community Edition 14.1.4
- **MySQL Workbench** 6.3 x64
- **SourceTree** v1.8.3.0 gestionando repositorios en BitBucket
- **EDGE Diagrammer** v6.24.2046
- **Gantt Project** 2.7.2

5.4 Gestión de riesgos

- Descripción: pérdida de documentos o código.
- Probabilidad de ocurrencia: baja.
- Posible impacto: alto.
- Posibles efectos: pérdida total o parcial de código creado y documentación escrita.
- Cómo evitarlo: usando herramientas de control de versiones (Sourcetree/Bitbucket) y almacenamiento en la nube (Google Drive).

- Descripción: problemas de integración entre componentes de Unity.
- Probabilidad de ocurrencia: media.
- Posible impacto: alto.

- Posibles efectos: fallo de uno o más componentes, o incluso fallo global de la aplicación.
- Cómo evitarlo: desarrollo de un prototipo previo para probar si los componentes son compatibles.

- Descripción: actualización de componentes por parte del desarrollador.
- Probabilidad de ocurrencia: baja.
- Posible impacto: baja.
- Posibles efectos: fallo de componentes, cambio en la estructura de ficheros.
- Cómo evitarlo: emplear versiones estables (Long-Time Support) ó emplear última versión y actualizar tras períodos largos de tiempo, sólo bajo estricta necesidad.

6 Conclusiones

6.1 Valoración personal

Este trabajo me ha permitido descubrir la gran responsabilidad que se tiene al desarrollar un proyecto de estas dimensiones con total autonomía. Una única persona debe hacer las labores de análisis, programación y gestión de proyecto, y todo ello requiere grandes conocimientos técnicos y a la vez una gran capacidad de planificación.

Los trabajos de mediano o gran tamaño realizados a lo largo de la carrera tenían un objetivo bastante marcado, y con niveles de exigencia y complejidad variables. Se tenía cierta libertad a la hora de escoger las tecnologías y herramientas, y la parte de gestión normalmente no era tan relevante. La diferencia entre esos trabajos y este proyecto ha servido para aproximarme a la realidad que se experimenta en la vida laboral, especialmente si se deciden llevar a cabo proyectos por cuenta propia.

A nivel técnico, he aprendido a utilizar por mi cuenta tecnologías muy diferentes a las que se enseñan a lo largo de este grado, como Unity3D, que a su vez me ha permitido experimentar con la realidad aumentada.

6.2 Líneas de expansión futuras

El principal componente que se podría ampliar de cara al futuro sería el modo de visión en la GUI de tableros y en la cámara de realidad aumentada. Se intentó integrar el plugin GoogleVR (anteriormente Cardboard) para poder emplear el móvil con unas gafas de realidad virtual. Esto fue imposible a corto plazo por los problemas de integración de Vuforia y GoogleVR, puesto que ambas librerías tienen un Manifest propio y cualquier modificación de dicho archivo XML provoca que el .apk no pueda construirse.

Otra mejora de gran importancia implicaría utilizar una base de datos remota para almacenar marcadores gráficos. De este modo la aplicación no estaría limitada al uso de 25 marcadores, y se podría insertar nuevos, eliminar existentes o reasignar manualmente a una etapa/columna deseada.

Un tercer cambio de gran importancia técnica y también estética sería el uso de AngularJS en el cliente web. Mejoraría la calidad del código Javascript y también la usabilidad y apariencia de los distintos menús.

El soporte multi-lenguaje para los textos en pantalla y mensajes de error también se valora positivamente para desarrollos futuros.

Bibliografía

1. Vuforia: https://developer.vuforia.com/
2. Integración de Vuforia y GoogleVR: https://developer.vuforia.com/library/articles/Solution/Integrating-Cardboard-to-the-ARVR-Sample
3. Migración de proyectos Unity a Vuforia 5.5: https://developer.vuforia.com/library/articles/Solution/Migrating-Unity-Projects-to-Vuforia-5-5
4. Unity3D: http://unity3d.com/
5. ARToolkit: http://artoolkit.org/
6. Gradle: http://gradle.org/
7. Project Grizzly: https://grizzly.java.net/
8. GlassFish Application Server: https://glassfish.java.net/
9. Creación de códigos QR: http://www.qrcode.es/es/generador-qr-code/
10. Repositorio de aplicación móvil: https://bitbucket.org/guillepg/tfgapp5
11. Repositorio del servidor y portal web: https://bitbucket.org/guillepg/tfg_server

Índice de figuras

Anexo I: manual de instalación

Servidor

Para configurar y ejecutar el servidor REST implementado, es necesario importar el código disponible en [11]. Se puede clonar el repositorio entero o descargar todo el código en un archivo comprimido .ZIP.

El proyecto ha sido desarrollado con IntelliJ IDEA y es posible que la estructura de paquetes y archivos de configuración no funcionen correctamente con otros entornos de desarrollo como Netbeans o Eclipse.

La clase controller.Board.Server se debe ejecutar para que el servidor arranque. No es necesario introducir ningún parámetro adicional.

Para acceder a la API será necesario consultar previamente la IP interna que tenga el equipo en la red, o la pública si se puede acceder desde el exterior a la red. En el caso de redes caseras esto no siempre es posible porque el acceso está muy restringido desde el exterior por el propio operador, pero en dominios como el de la Universidad de Zaragoza sí que es posible. Este servidor se probó correctamente empleando la conexión del equipo http://lanuza.cps.unizar.es, ubicado en el laboratorio 2.10, y con clientes conectados a la red inalámbrica y a la red de datos 4G de Movistar.

Cliente app

La aplicación desarrollada en Unity es compatible con Android 4.2 y posteriores, y ha sido probada en dispositivos con Android 4.4 y 5.1. Para ejecutarla es necesario instalar el archivo .apk presente en el repositorio [10].

Tras la instalación se debe crear manualmente en Android/data/ el fichero com.guillepg.TFGApp5/files/IP.txt con la IP y el puerto donde esté ubicado el servidor, por ejemplo: http://lanuza.cps.unizar.es en la línea 1, y 8084 en la línea 2).

Consola de administrador

Se puede ejecutar la consola de administración desde la misma máquina que haga las funciones de servidor, o también puede exportarse a un archivo .jar para su ejecución desde máquinas distintas.

Para iniciar la aplicación debe ejecutarse la clase console.AdminConsole. Es necesario que el sistema admita ventanas emergentes de Java Swing, puesto que la entrada de datos y la navegación por los distintos menús se realiza mediante una interfaz gráfica de este tipo.

Base de datos

Es necesario disponer de un gestor de bases de datos como MySQL Workbench para poder crear y manejar las tablas necesarias. Entre los archivos del repositorio [11] se puede encontrar un modelo exportado de base de datos (archivo .mwb), el archivo tfg_schema.sql que contiene el código SQL necesario para crear las distintas tablas del esquema TFGDB, y el archivo tfg_insert.sql que contiene una serie de datos de prueba para poblar las tablas y poder realizar pruebas.

Anexo II: manual de usuario

Vista de página web

Pantalla de inicio de sesión

En la primera pantalla que aparece al acceder a la aplicación web, debemos introducir los datos de inicio de sesión.

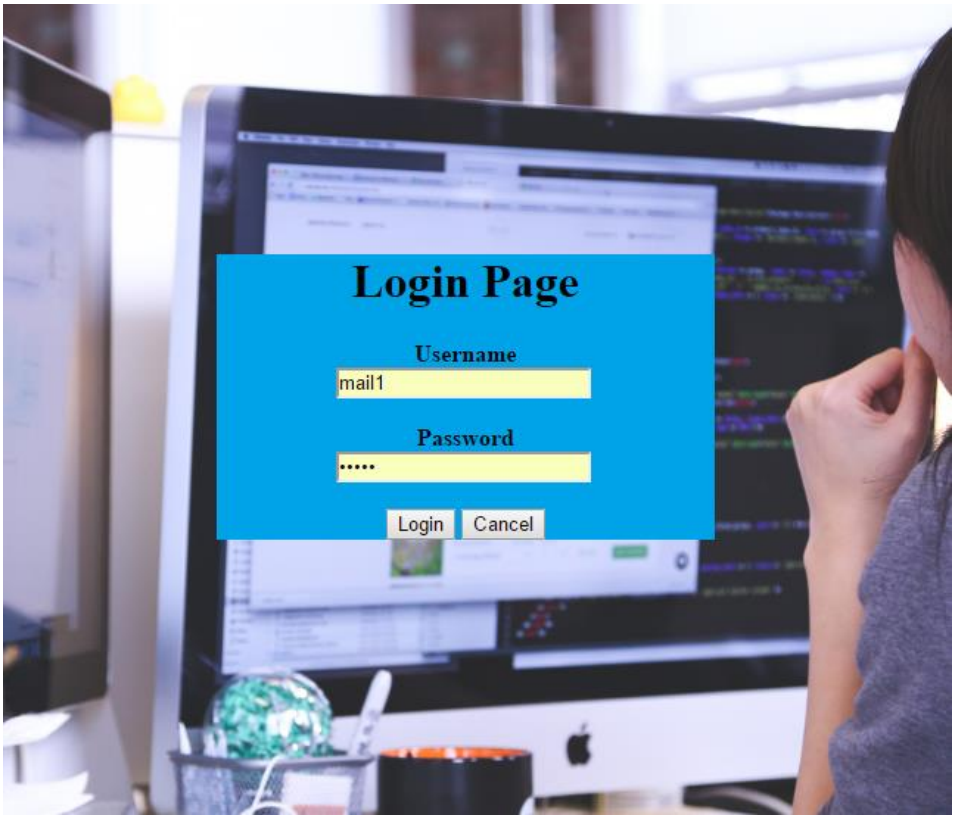


Figura 9: página de inicio de sesión

Si la combinación de usuario y contraseña no son correctos o el usuario no existe, se indicará en un mensaje de error en una ventana emergente del navegador.

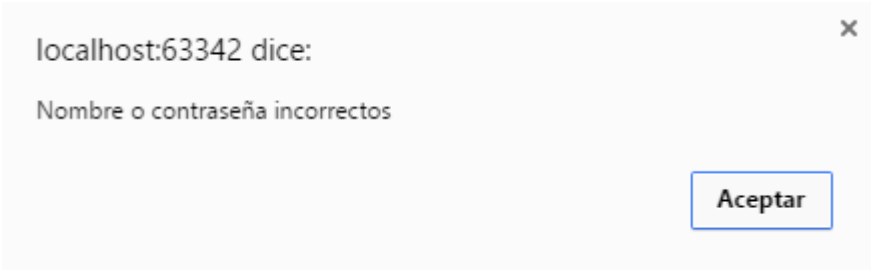


Figura 10: error en los datos de acceso

Si los datos de sesión son válidos se accederá a la segunda pantalla para seleccionar uno de los tableros que tengamos autorizados.

Pantalla de selección de tablero

En esta pantalla aparecen los tableros a los que nuestro equipo tiene acceso.

Figura 11: pantalla de selección de tablero

Si el equipo al que pertenece el usuario no tiene acceso autorizado a ningún tablero, se indicará en un mensaje de alerta y se le redireccionará a la pantalla de login para iniciar sesión de nuevo. En este caso se deberá contactar con el administrador para que le conceda acceso a uno o más tableros.

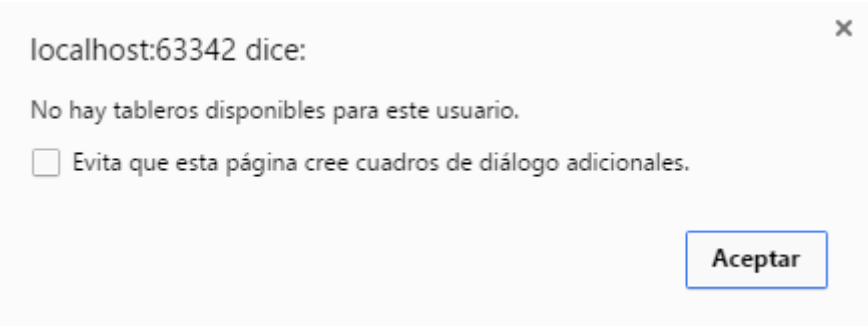


Figura 12: mensaje de error - no existen tableros

Una vez se haga clic sobre uno de los tableros, se navegará a la siguiente página donde aparecerán las etapas de dicho tablero.

En caso de acceder directamente a la URL de esta página sin pasar previamente por la de inicio de sesión, se informará de que no se ha iniciado sesión con un mensaje de alerta y se redireccionará a la primera pantalla.

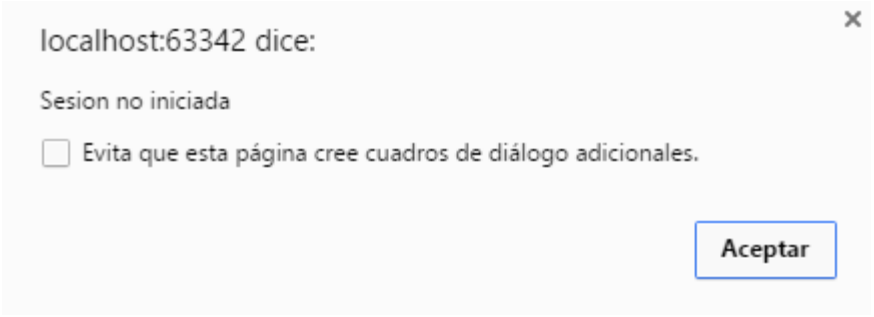


Figura 13: mensaje de error – sesión no iniciada

Vista global de un tablero

A continuación se ven todas las etapas asociadas al tablero seleccionado, y dentro de cada etapa las tareas contenidas en ella.

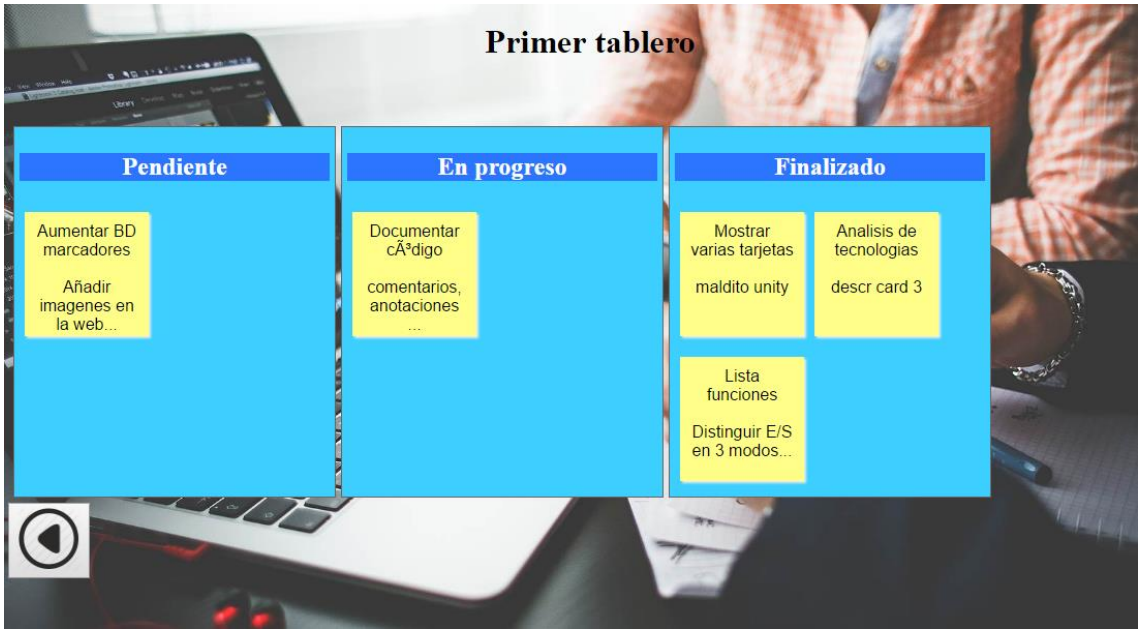


Figura 14: pantalla de vista global de tablero

Cada tarea individual se puede arrastrar y soltar de una columna a otra para indicar que su estado ha cambiado.

Al hacer clic con el botón secundario sobre un hueco de una etapa o sobre una tarea, se accede a dos menús contextuales distintos con funciones de creación y eliminación.

Menú contextual sobre una etapa

Al clicar en un espacio libre de una etapa se permite crear una nueva columna en el tablero, eliminar la seleccionada o crear una tarea en ese tablero.

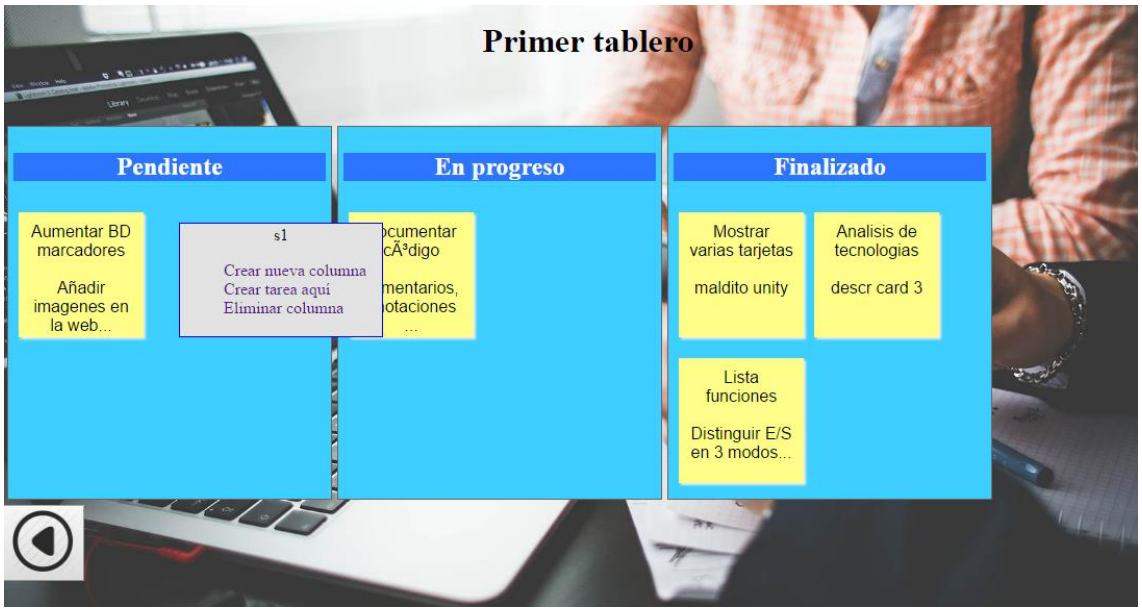


Figura 15: menú contextual sobre una etapa/columna

Si se hace clic en “crear nueva columna” se accede al formulario de creación de columna (figura 17), y al clicar en “crear tarea aquí” se accede al de creación de tarea (figura 18). Al hacer clic en “eliminar columna”, se elimina la columna (etapa) seleccionada directamente.

Menú contextual sobre una tarea

Al clicar sobre una tarea aparece este menú contextual que permite eliminarla directamente, sin pedir confirmación previa.

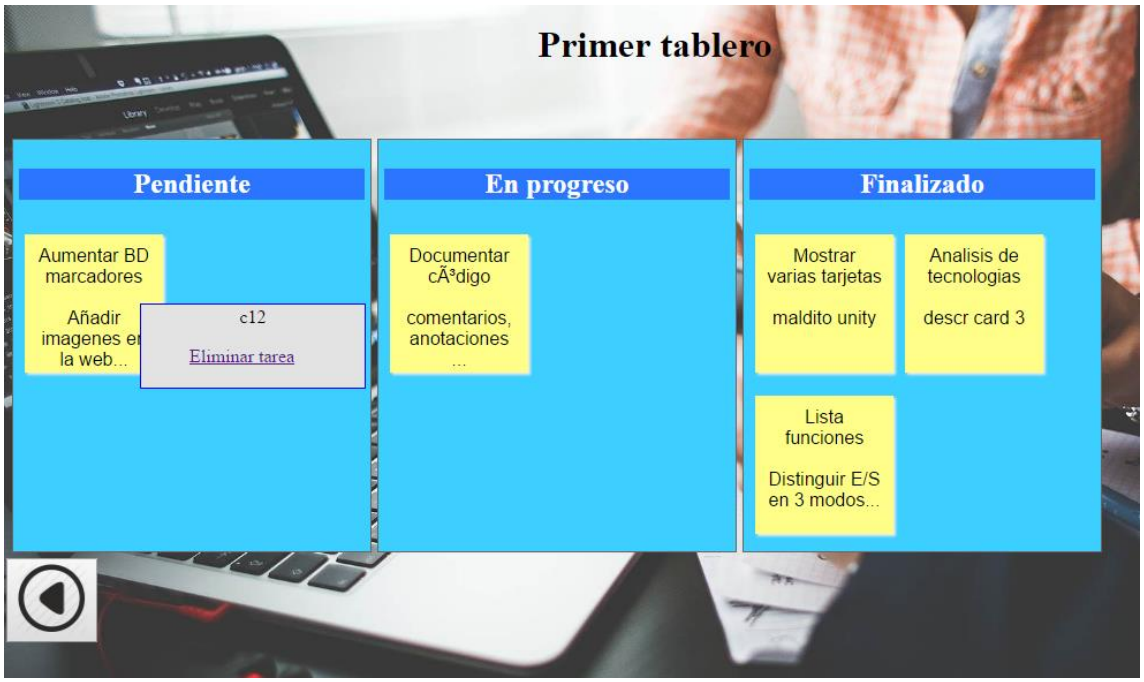


Figura 16: menú contextual sobre una tarea

Formulario de creación de etapa/columna

En esta página se pueden introducir los dos datos esenciales de cada etapa: nombre y descripción. El ID y la posición relativa se asignan internamente.

Figura 17: formulario de creación de columna/etapa

Cuando se introduzcan los datos y se haga clic en “Crear”, se enviará la petición al servidor y se redirigirá el navegador hacia la vista del tablero donde se habrá creado la nueva etapa. Si ha sucedido algún error, aparecerá la vista sin ninguna modificación.

Formulario de creación de tarea

Los campos de texto funcionan del mismo modo que en la pantalla anterior (creación de etapas), pero se debe introducir una etapa válida en “Estado de la tarea” seleccionando uno de los valores de los botones circulares inferiores.

Figura 18: formulario de creación de una tarea

Tras clicar en “Crear”, el comportamiento es análogo a la creación de una etapa.

Aplicación para Android

La aplicación móvil permite consultar la información de los tableros y tareas almacenadas en el sistema. Puede ver las tareas de cada tablero con una perspectiva amplia, en una vista que muestra 12 tareas al mismo tiempo (4 por etapa, 3 etapas a la vez), o reconocer marcadores con la cámara del dispositivo para acceder directamente a la información sin tener que navegar hasta una etapa de un tablero en concreto.

Pantalla de inicio de sesión

Figura 19: pantalla de inicio de sesión en la APP

En esta primera pantalla debemos introducir el correo electrónico y la contraseña personal. Si es correcta avanzaremos a la siguiente pantalla (figura 21). Para crear un usuario o modificar la contraseña de uno existente es necesario ejecutar la consola de administración.

Si por el contrario la combinación de correo y contraseña no es correcta, mostrará un mensaje de error y un botón para volver a la misma pantalla y reintroducir los datos (figura 20.1).

Figura 20.1: mensaje de error en los datos de sesión

Si al enviar los datos de sesión no hay respuesta del servidor, nos indicará otro mensaje de error indicando que no hay conexión (figura 20.2).

Figura 20.2: mensaje de error de conexión

Pantalla de selección de tablero y modo de vista

En esta pantalla el usuario acceder a la vista general de cada tablero al que tenga acceso (figura 22) o pasar al modo cámara para reconocer marcadores (figura 24).

La flecha inferior izquierda cierra la aplicación, y al reabrirla se debe iniciar sesión de nuevo.

Figura 21: pantalla de selección de tablero en la APP

Pantalla de vista de tablero

En la figura 22 se muestran las tareas y etapas asociadas al tablero seleccionado en la anterior pantalla.

Para actualizar la vista en caso de haber realizado cambios desde la consola de administración o desde la vista web, basta con pulsar el botón “refresh” de la parte central superior y se recargará toda la vista del tablero mostrado.

Si se desea ver el siguiente grupo de etapas se debe pulsar la flecha superior derecha. Esta acción eliminará de la pantalla las tres mostradas anteriormente y mostrará la información de las siguientes. En caso de que no hubiera más etapas, la pulsación en el botón no tendrá ningún

efecto. En caso de que haya más etapas pero sean menos de tres, se mantendrá la estructura de 3 columnas pero sólo se verá la información de las existentes en sus respectivos espacios.

Si se desea ver anteriores etapas, el funcionamiento del botón superior izquierdo es análogo al anterior. Si se hace clic en este botón desde la primera página no se verá ningún cambio.

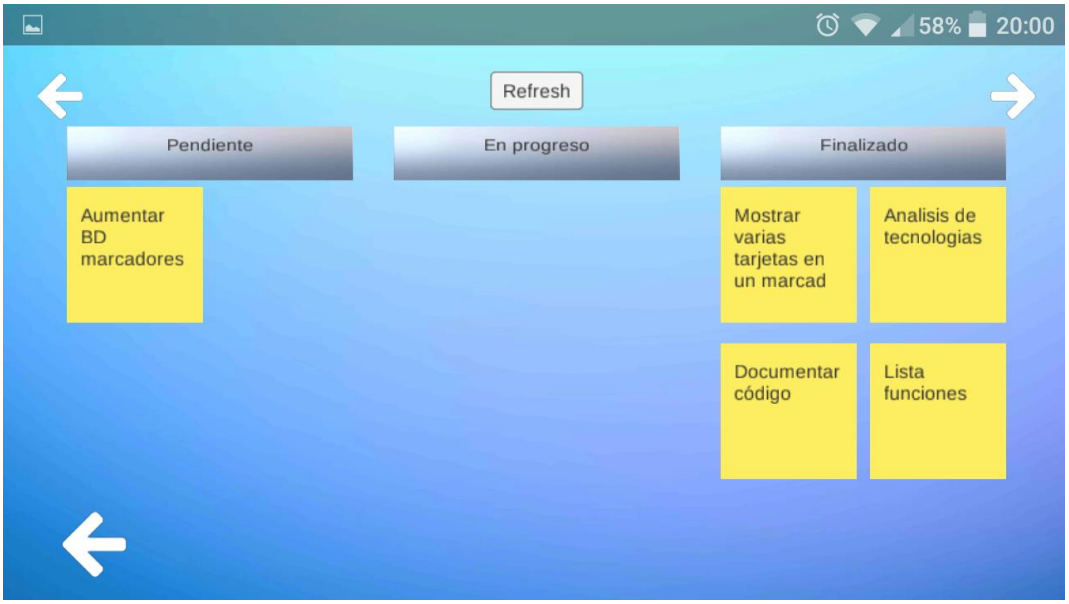


Figura 22: vista de tareas y etapas asociadas al tablero seleccionado.

Pantalla de detalle de una tarea

Para poder ver más detalles de una tarea, al hacer clic sobre ella se accede a otra pantalla en la que se ve la descripción de la tarea (no solamente el título) y el equipo responsable de llevarla a cabo (figura 23).

Figura 23: vista en detalle de una tarea.

Al hacer clic en el botón “cerrar” se vuelve a la vista de tablero. Al clicar sobre “borrar tarea” se envía una petición al servidor para eliminar dicha tarea, se cierra la vista actual (detalle de una tarea), se habilita la vista global del tablero y se actualiza para mostrar el último cambio realizado.

Modo cámara con realidad aumentada (AR)

Al colocar un marcador en el campo de visión de la cámara se carga el marcador asociado, se manda una consulta al servidor con el identificador del marcador y se muestra la información de la etapa asociada a ese marcador.

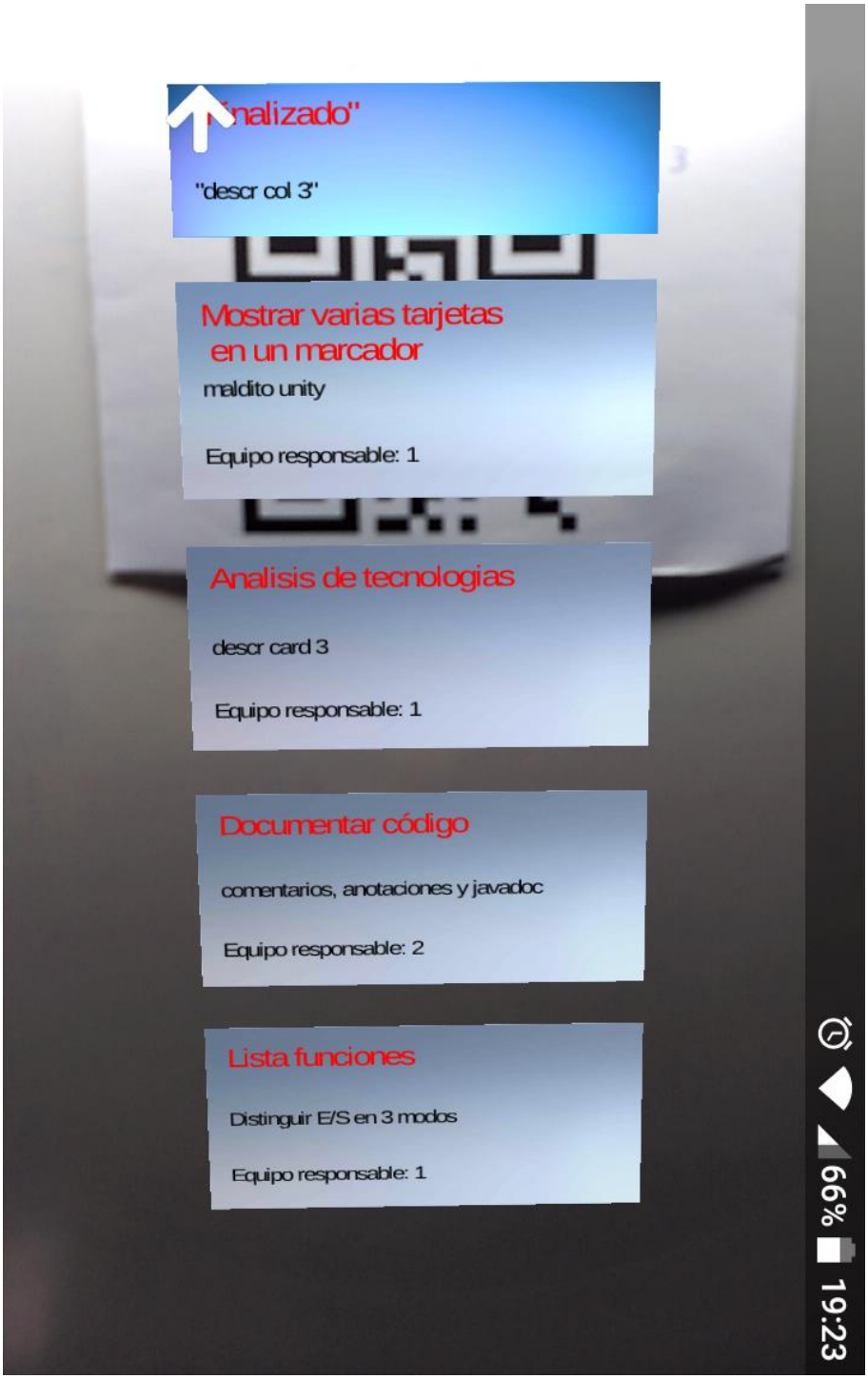


Figura 24: lectura de un marcador gráfico

La cámara del dispositivo puede reconocer varios marcadores al mismo tiempo, pero por cuestiones de rendimiento se ha limitado a 3. El desarrollador recomienda como máximo 5.

Si uno de los marcadores reconocidos pertenece a un tablero al que el usuario no tiene acceso, se muestran los mismos gráficos con un mensaje informando de la falta de permisos.

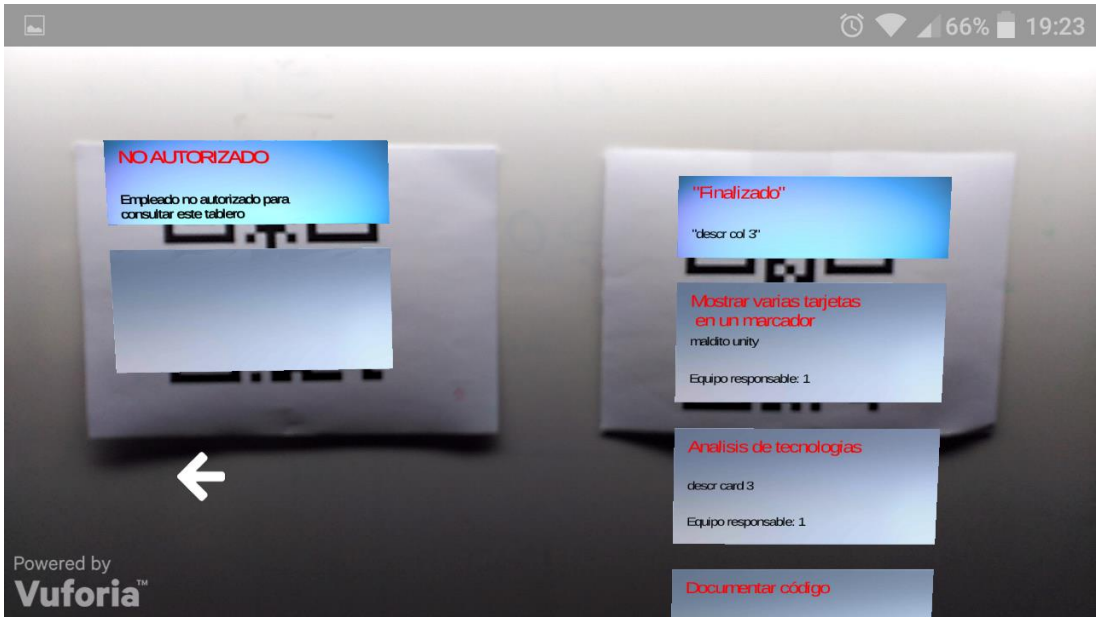


Figura 25: mensaje de error de empleado no autorizado, y lectura de un marcador autorizado.

Si uno de los marcadores, por el contrario, no está en uso porque su identificador asociado no corresponde al de ninguna columna (etapa) de la base de datos, mostrará un mensaje indicando que no hay datos que mostrar para ese marcador.

Figura 26: mensaje de error – marcador no asociado

Al clicar en la flecha de la zona inferior izquierda, se cierra el modo cámara y se vuelve a la escena inicial en la que será necesario introducir los datos de sesión de nuevo.

Consola de administrador

La consola de administrador permite realizar funciones de creación, modificación y eliminación sobre los datos presentes en el sistema.

Menú principal

En la pantalla inicial se elige sobre qué entidad se desea realizar un cambio. Después de cada operación se volverá a mostrar este pequeño menú para indicar si se desea realizar otra operación o finalizar la ejecución.

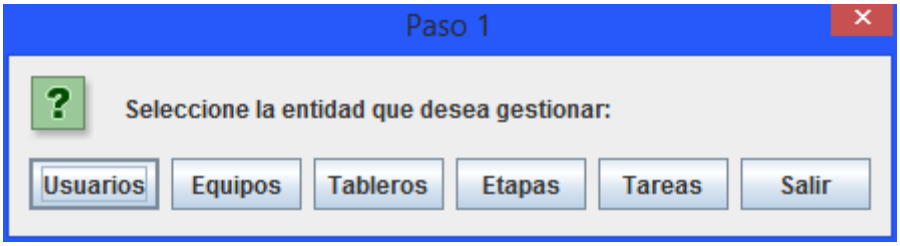


Figura 27: menú principal de la consola de administrador

Gestión de usuarios

Al clicar en el botón “usuarios” se puede elegir entre crear un nuevo usuario, y eliminar o modificar los datos de uno existente.

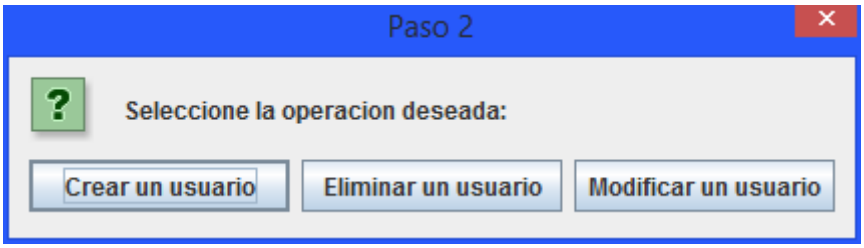
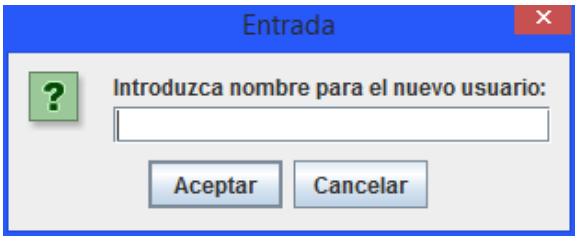
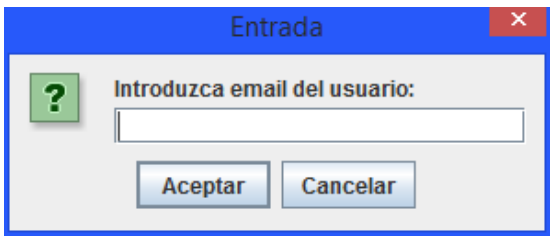


Figura 28: menú de gestión de usuarios

Al elegir la opción de “Crear”, se muestra una serie de pantallas en la que se deben introducir los datos del usuario uno a uno. En caso de “modificar un usuario” la secuencia de datos a introducir es la misma, y además en primer lugar se pide el ID del usuario que se quiere modificar.





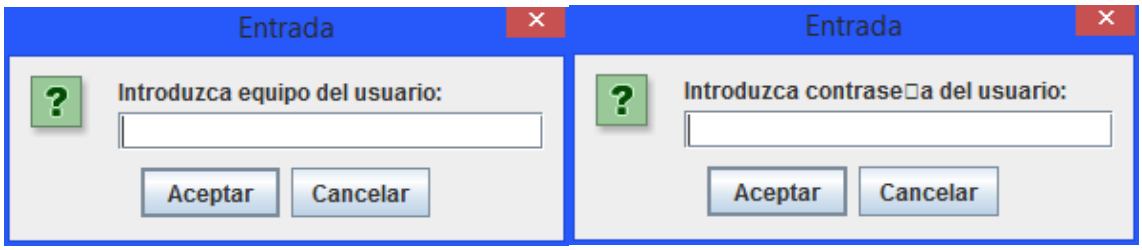


Figura 29-32: inserción de datos de nuevo usuario

Si la inserción o modificación ha tenido éxito, se mostrará un mensaje indicándolo.

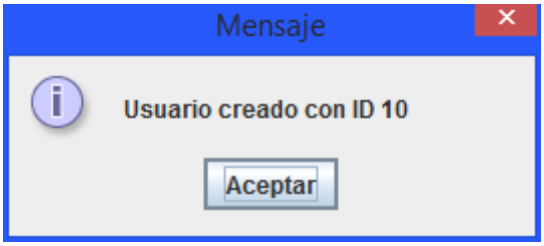


Figura 33: mensaje de éxito en la creación de usuario

De lo contrario se mostrará un mensaje mostrando que hubo un error insertando los datos, o que no se pudo conectar con el servidor.

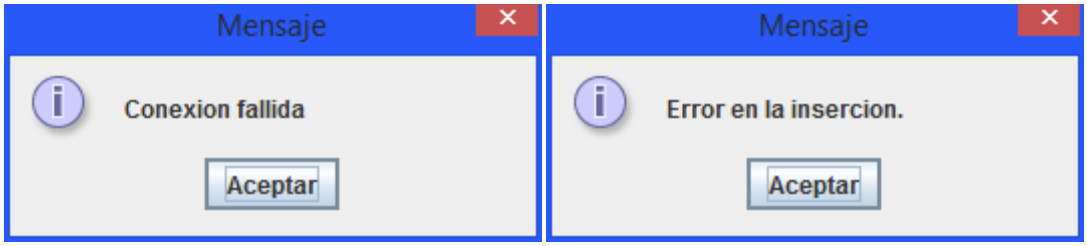


Figura 34 y 35: mensajes de error en la inserción de usuario

Al indicar que se desea eliminar un usuario solamente se pide el identificador de dicho usuario, y el siguiente mensaje mostrará si la operación ha tenido éxito o si ocurrió algún error.

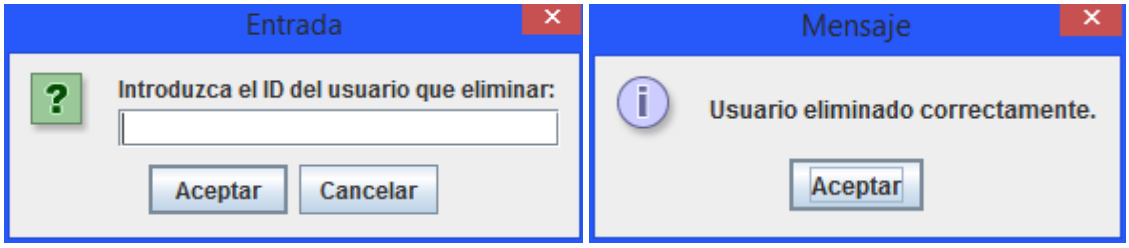


Figura 36 y 37: inserción de identificador del usuario a eliminar, y mensaje de éxito

Gestión de equipos de trabajo

Clicando en el segundo botón del menú principal se muestra el submenú que permite gestionar los equipos de trabajo y sus permisos asociados.

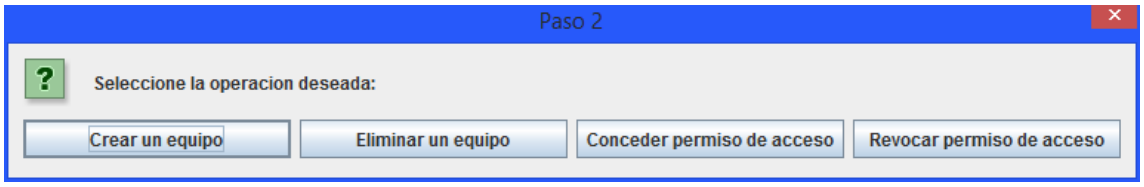


Figura 38: menú de gestión de equipos

La primera opción, "crear equipo", es análoga a "crear usuario" anteriormente explicada. Se deben introducir menos datos, puesto que la información personal de un usuario es más completa y concreta que la de un grupo de empleados.

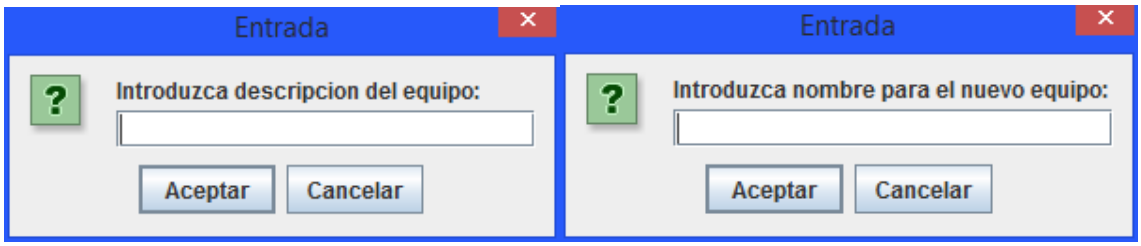


Figura 39 y 40: inserción de datos del nuevo equipo

Si los datos se insertan correctamente tras hacer clic en "Aceptar", se mostrará lo siguiente:

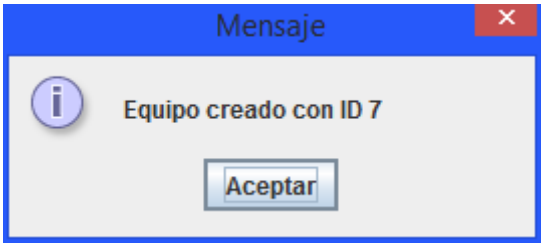
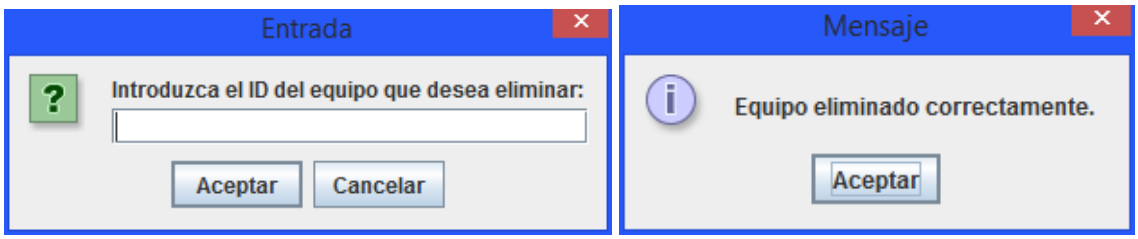


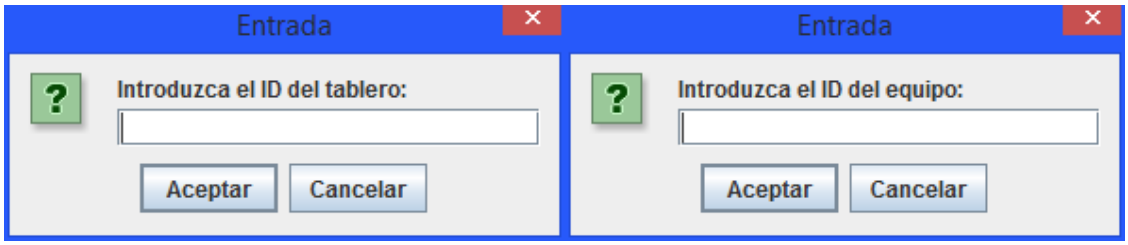
Figura 41: mensaje de éxito al crear equipo

La función "eliminar equipo" funciona de manera similar a "eliminar usuario". Introduciendo un ID de un equipo existente y haciendo clic en "Aceptar" se eliminará dicho equipo. De lo contrario se mostrarán los mensajes "error al eliminar equipo" o "error de conexión".



Figuras 42 y 43: inserción de ID para eliminación de equipo

Para conceder o denegar el acceso de un equipo a un determinado tablero se debe seleccionar la correspondiente función de entre las dos últimas.



Figuras 44 y 45: inserción de datos para eliminación de permiso de acceso

Si la creación del permiso de acceso ha sido satisfactoria se mostrará un mensaje indicándolo.

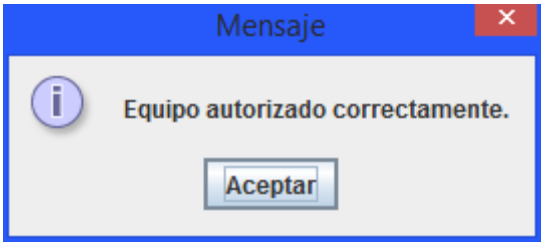


Figura 46: mensaje de éxito al autorizar equipo

Si se ha eliminado el permiso entre ese equipo y ese tablero se indicará que ya no está autorizado.

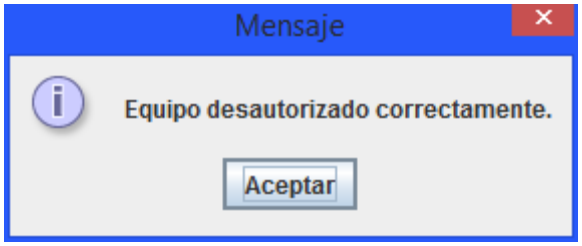


Figura 47: mensaje de éxito al desautorizar equipo

Gestión de tableros y etapas

Las dos primeras funciones del submenú de tableros y las únicas dos funciones del submenú de etapas son equivalentes a las explicadas anteriormente, que se encargaban de crear o eliminar usuarios y equipos.

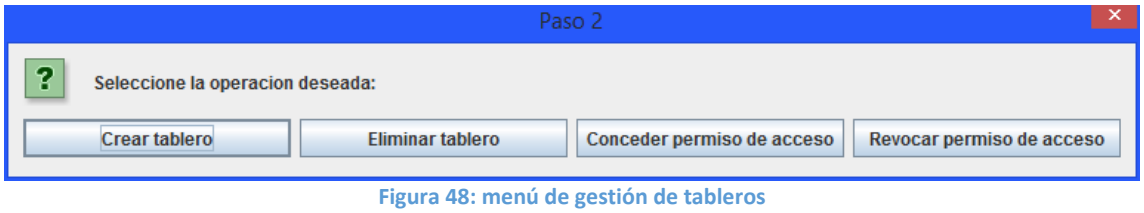


Figura 48: menú de gestión de tableros

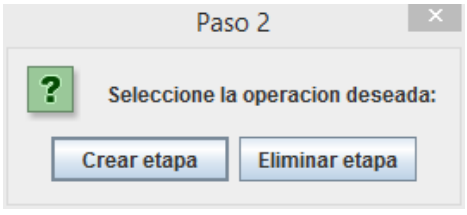
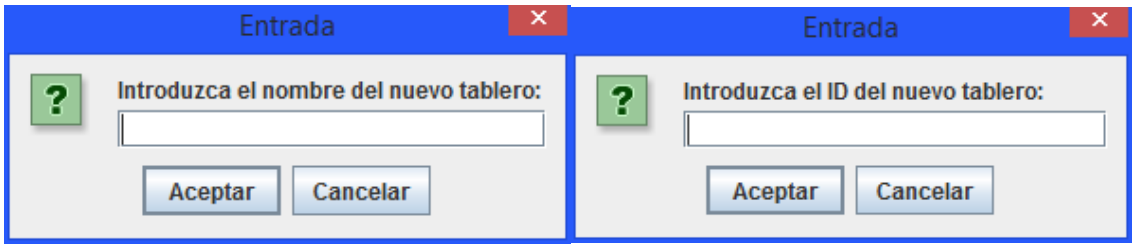


Figura 49: menú de gestión de etapas

La creación de tableros funciona del mismo modo que las operaciones similares anteriores.



Figuras 50 y 51: inserción de datos para la inserción de nuevo tablero

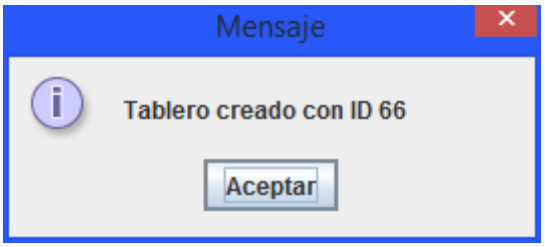
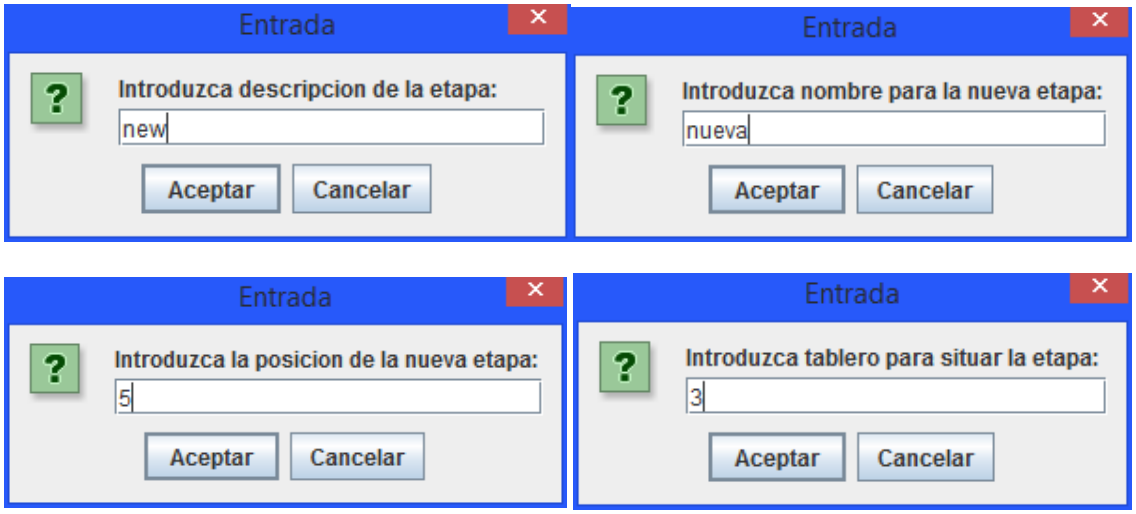


Figura 52: mensaje de éxito al crear nuevo tablero

La creación de etapas, además, requiere que se introduzca el ID del tablero que contendrá la nueva columna, y la posición de esta con respecto a las ya existentes. Se mostrarán de izquierda a derecha de menor a mayor valor del campo "posición". En caso de existir dos etapas/columnas con la misma posición, el ID menor tendrá prioridad.



Figuras 53, 54, 55 y 56: inserción de datos para la creación de una nueva etapa/columna

En caso de que los datos anteriores sean válidos, se mostrará un mensaje de éxito. Si los campos de texto exceden la longitud máxima, o se introduce un ID que no se corresponde a ningún tablero, se mostrará un mensaje de error.

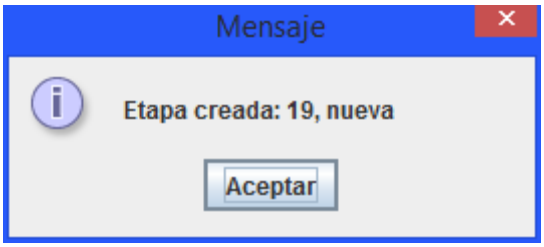
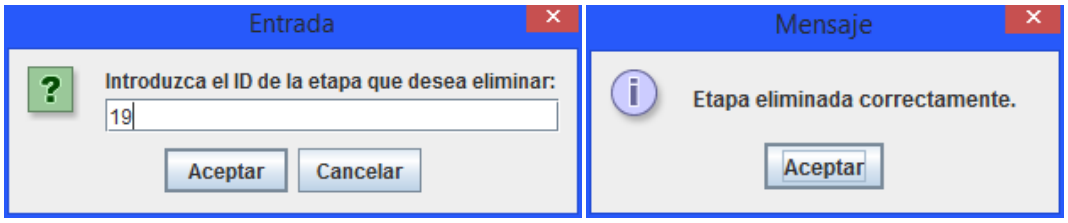
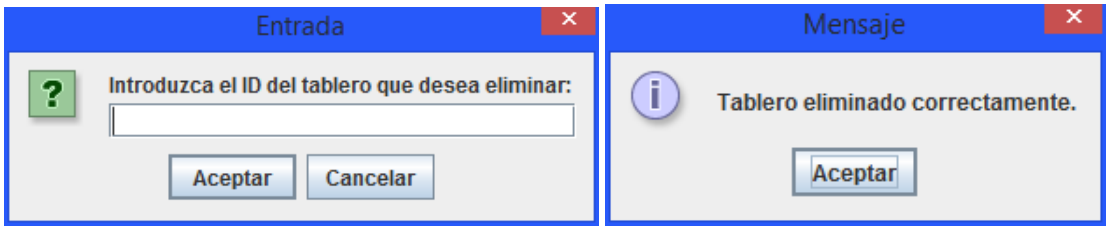


Figura 57: mensaje de éxito al crear nueva etapa

Borrar una etapa o un tablero también funcionan del mismo modo que un usuario o un equipo:



Figuras 58 y 59: inserción de ID para eliminación de etapa, y mensaje de éxito



Figuras 60 y 61: inserción de ID para eliminación de tablero, y mensaje de éxito

En el caso de estas dos entidades, ya que existen dependencias entre ellas y también con la entidad “Tarjeta”, no se podrá borrar un tablero que contenga una o más etapas, ni una etapa que contenga una o más tarjetas.

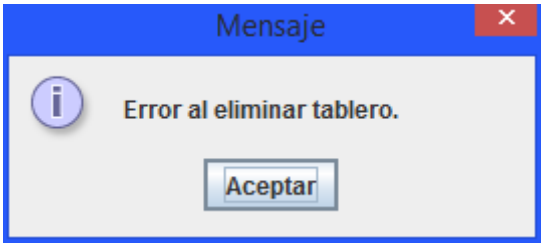


Figura 62: mensaje de error durante la eliminación

Gestión de tareas

El menú de gestión de tareas permite crear tareas con toda su información asociada, o eliminar tareas ya existentes.

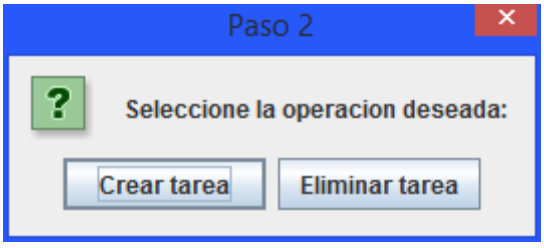
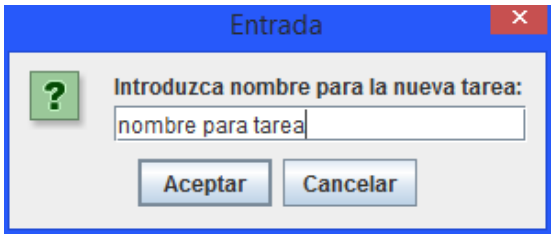
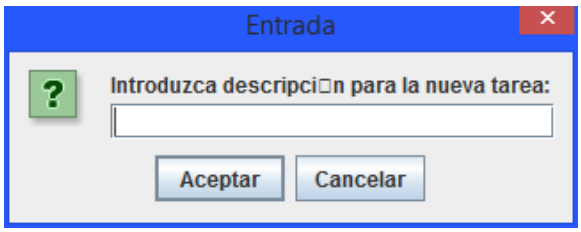
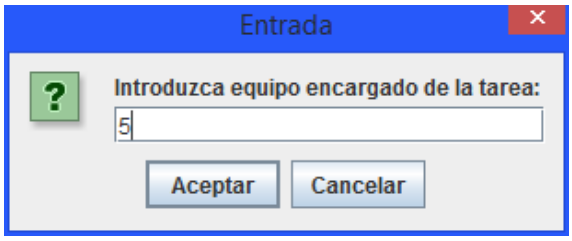
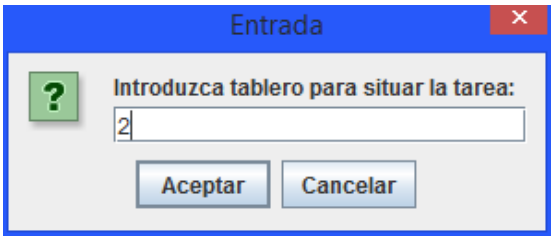


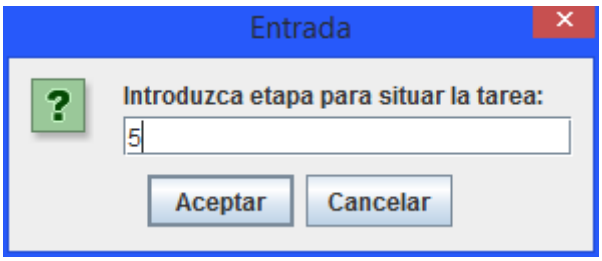
Figura 63: menú de gestión de tareas











Figuras 64-68: inserción de datos para creación de nueva tarea

Si los datos anteriores son válidos, se insertará la nueva tarea y mostrará un mensaje de éxito.

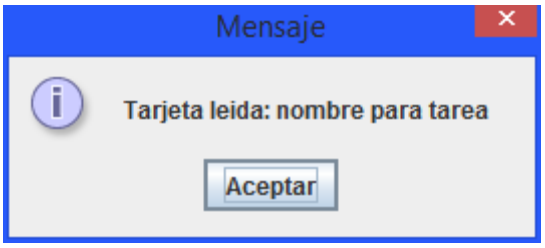


Figura 69: mensaje de éxito en la creación de tarea

Si en los campos que deben leer un número (tablero, etapa y equipo) se introduce un texto, nos avisará del fallo con el mensaje de error correspondiente:

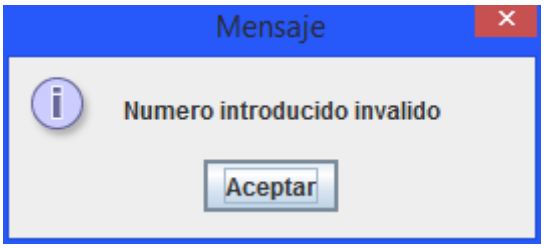
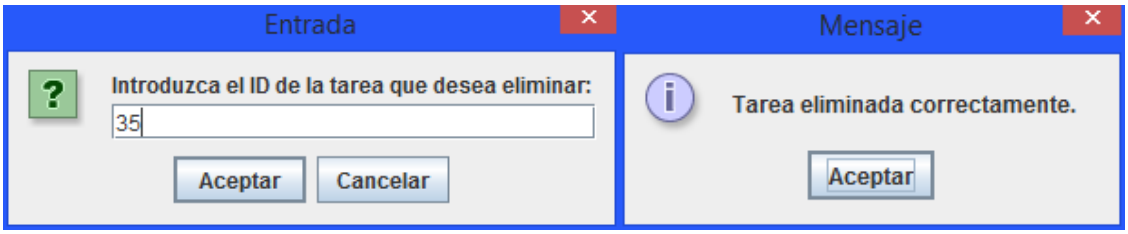


Figura 70: mensaje de error en el formato del número introducido

El borrado de tareas necesita que se introduzca tan sólo el ID de la tarea:



Figuras 71 y 72: inserción de ID para eliminación de tarea, y mensaje de éxito

Anexo III: interfaz del servidor REST

La clase RestServices es la encargada de gestionar las peticiones entrantes a través de la ruta raíz /board/.

Métodos de tipo GET

http://direccionIP:puerto/board/{id} consume un identificador numérico en la URL de la petición y devuelve en formato JSON toda la información del tablero con el ID anterior.

http://direccionIP:puerto/board/card/{id} devuelve la información de la tarjeta con identificador 'id' de la petición.

http://direccionIP:puerto/board/checkpermissions/{id} devuelve la lista de tableros a los que tiene acceso el equipo 'id' en formato JSON. La petición POST board/checkemployee/ es el primer paso que forma el inicio de sesión, y /board/checkpermissions/ es el segundo paso.

http://direccionIP:puerto/board/employee/{id} obtiene la información del empleado con identificador 'id' y lo devuelve en formato JSON para ser procesado en el cliente.

http://direccionIP:puerto/board/stage/{id} obtiene la información de una etapa (columna) y todas las tareas que contiene, y la devuelve al cliente en texto JSON.

http://direccionIP:puerto/board/stageinfo/{name} obtiene únicamente la información principal de una etapa, realizando la búsqueda por nombre en lugar de su identificador. No obtiene las tareas asociadas.

http://direccionIP:puerto/board/team/{id} devuelve en formato JSON al cliente la información de un equipo.

http://direccionIP:puerto/board/createpermission/{teamID}/{boardID} autoriza al equipo con identificador 'teamID' a acceder al tablero con identificador 'boardID'.

http://direccionIP:puerto/board/deletepermission/{teamID}/{boardID} revoca el acceso del equipo 'teamID' al tablero 'boardID'.

Métodos de tipo POST

Los métodos POST consumen texto en formato JSON que contiene la información de la entidad que se quiere crear en el servidor, y devuelven un código 201 CREATED si la inserción tuvo éxito o 400 BAD REQUEST en caso de error.

http://direccionIP:puerto/board/board/ crea un tablero nuevo si la información proporcionada es válida y completa.

http://direccionIP:puerto/board/card/ crea una tarea nueva si la información es válida, es decir, si la etapa y tablero introducidos son correctos.

http://direccionIP:puerto/board/stage/ crea una etapa nueva si los datos introducidos son correctos, es decir, si el tablero introducido existe y se tiene permiso de acceso.

http://direccionIP:puerto/board/checkemployee/ envía la información proporcionada por el usuario en el formulario de inicio de sesión, y comprueba si es correcta. Posteriormente se comprueban los permisos de acceso de su equipo, como se ha indicado anteriormente en /board/checkpermissions/.

http://direccionIP:puerto/board/team/ crea un nuevo equipo en el sistema con la información proporcionada.

Métodos de tipo PUT

Los siguientes métodos consumen las respectivas entidades en formato JSON, y devuelven esas mismas entidades con un código de estado 200 OK si la actualización tuvo éxito, o 400 BAD REQUEST en caso de que hubiera sucedido algún error.

http://direccionIP:puerto/board/card/{id} actualiza toda la información de una tarjeta, salvo su ID.

http://direccionIP:puerto/board/employee/{id} actualiza todos los datos de un empleado menos su identificador único.

http://direccionIP:puerto/board/stage/{id} actualiza la información asociada a una etapa.

http://direccionIP:puerto/board/team/{id} actualiza la información de un equipo.

Métodos de tipo DELETE

Los siguientes métodos consumen un ID en formato numérico enviado en la URL de la petición, y generan un texto JSON que se envía al cliente con el estado 404 NOT FOUND en caso de error, o 204 NO CONTENT en caso de que la eliminación en la base de datos haya tenido éxito.

http://direccionIP:puerto/board/{id} elimina el tablero con el identificador ID, si existe, y todas las etapas y tareas que tenga asociadas.

http://direccionIP:puerto/board/card/{id} elimina la tarea con el ID proporcionado.

http://direccionIP:puerto/board/employee/{id} elimina de la base de datos el perfil del empleado con identificador ID.

http://direccionIP:puerto/board/stage/{id} elimina la etapa con identificador ID y sus tareas asociadas.

http://direccionIP:puerto/board/team/{id} elimina el equipo con identificador ID si no existen empleados relacionados con dicho equipo.

Métodos de tipo OPTIONS

Métodos creados para responder a las peticiones que en algunas ocasiones envían los navegadores web. En todos los casos devuelven un estado 200 OK en formato JSON para que el navegador pueda enviar las siguientes peticiones sin interrumpir su ejecución.

http://direccionIP:puerto/board/{id}

http://direccionIP:puerto/board/

http://direccionIP:puerto/board/card/{id}

http://direccionIP:puerto/board/card/

http://direccionIP:puerto/board/employee/{id}

http://direccionIP:puerto/board/employee/

http://direccionIP:puerto/board/stage/{id}

http://direccionIP:puerto/board/stage/

http://direccionIP:puerto/board/team/{id}

http://direccionIP:puerto/board/team/

http://direccionIP:puerto/board/stageinfo/{name}

http://direccionIP:puerto/board/checkemployee/

http://direccionIP:puerto/board/createpermission/{teamID}/{boardID}

http://direccionIP:puerto/board/deletepermission/{teamID}/{boardID}

Anexo IV: casos de uso extendidos

ID	1
Nombre	Iniciar sesión
Descripción	Validar datos de sesión y conceder acceso al sistema.
Actor principal	Actor en app, actor en web
Precondición	El actor ha accedido a la web o ha ejecutado la app móvil.
Postcondición	El usuario tiene acceso al sistema y puede ver o modificar la información
Escenario de éxito	1. El actor introduce su correo o nombre identificador y su contraseña. 2. El actor hace clic en “iniciar sesión”. 3. Se muestra la pantalla de selección de tablero, con los tableros a los que tiene acceso el actor.
Extensiones	3.b: si no se puede conectar con el servidor, se muestra el mensaje indicando “error de conexión”. 3.c: si los datos de inicio de sesión no son correctos, se muestra un mensaje indicando “error de autenticación”.

ID	2
Nombre	Ver tablero
Descripción	Acceder a la información de un tablero.
Actor principal	Actor en app.
Precondición	El actor ha iniciado sesión.
Postcondición	Se muestran las etapas y tareas de un tablero.
Escenario de éxito	1. El usuario inicia sesión. 2. Se muestra una pantalla con un botón por cada tablero al que el actor tiene permiso de acceso. 3. El actor clica en uno de los botones. 4. Se muestra la vista global del tablero, con sus etapas y tareas.
Extensiones	2.b: si el usuario no puede acceder a ningún tablero, se muestra un mensaje diciendo que no tiene permisos de acceso. 4.b: si el tablero accedido no contiene etapas, se muestra un mensaje indicando que está vacío y es necesario crear etapas desde la consola o la vista web.

ID	3
Nombre	Crear tablero
Descripción	Crea un nuevo tablero en el sistema
Actor principal	Actor en consola de administrador
Precondición	El actor ha iniciado la consola de administración
Postcondición	Se crea un nuevo tablero vacío en el sistema
Escenario de éxito	1. El usuario selecciona “tableros” en el menú inicial. 2. El usuario clica en “crear tablero” en el menú secundario. 3. El usuario introduce el ID y el nombre del nuevo tablero, y pulsa “aceptar”. 4. Se crea el nuevo tablero en el sistema. 5. Se vuelve a mostrar el menú inicial.
Extensiones	4.b: si el formato del ID es incorrecto (texto en lugar de número), se muestra un mensaje indicando error en el formato. 4.c: si tiene lugar algún fallo durante la inserción de datos, se muestra un mensaje indicando que no se ha insertado el tablero.

ID	4
Nombre	Eliminar tablero
Descripción	Elimina un tablero, junto con sus etapas y tareas.
Actor principal	Actor en consola de administrador
Precondición	El actor ha iniciado la consola de administración
Postcondición	El tablero seleccionado ya no existe en el sistema
Escenario de éxito	1. El usuario clica en “tableros” en el menú principal. 2. El usuario clica en “eliminar tablero” en el segundo menú. 3. El usuario introduce el ID del tablero a eliminar y hace clic en “aceptar” 4. Se elimina el tablero y se muestra un mensaje de éxito. 5. Tras clicar en “aceptar” se vuelve al menú principal.
Extensiones	4.b: si el tablero no existe, se muestra un mensaje de error. 4.c: si el dato introducido no tiene formato de número, se muestra un mensaje de error indicando que se ha escrito algo no válido.

ID	5
Nombre	Crear etapa
Descripción	Crea una nueva etapa (columna) en el tablero.
Actor principal	Actor en web.
Precondición	El actor ha iniciado sesión.
Postcondición	Se muestra la nueva etapa junto a las ya existentes.
Escenario de éxito	1. El usuario selecciona un tablero y accede a su vista. 2. El usuario hace clic secundario sobre cualquier etapa. 3. El usuario rellena los datos del formulario de creación y pulsa “crear etapa”. 4. Se crea la nueva etapa en el sistema. 5. Se vuelve a la vista de tablero, con la nueva etapa presente.
Extensiones	4.b: si alguno de los campos está vacío, se muestra el error y se impide continuar la ejecución.

ID	6
Nombre	Eliminar etapa
Descripción	Se elimina una etapa y las tareas que contiene
Actor principal	Actor en consola de administrador
Precondición	El actor ha iniciado la consola de administración
Postcondición	La etapa indicada ya no existe
Escenario de éxito	1. El usuario clica en “etapas” en el menú principal. 2. El usuario clica en “eliminar etapa” en el segundo menú. 3. El usuario introduce el ID de la etapa a eliminar y hace clic en “aceptar” 4. Se elimina la etapa y se muestra un mensaje de éxito. 5. Tras clicar en “aceptar” se vuelve al menú principal.
Extensiones	4.b: si la etapa no existe, se muestra un mensaje de error. 4.c: si el ID introducido no tiene formato de número, se muestra un mensaje de error indicando que se ha escrito algo no válido.

ID	7
Nombre	Ver tarjeta
Descripción	El usuario puede ver el nombre, descripción y equipo encargado de una tarea.
Actor principal	Actor en app
Precondición	El usuario ha iniciado sesión
Postcondición	Se muestra la información de una tarea en pantalla
Escenario de éxito	1. El usuario clica en un tablero en la pantalla de selección de tableros. 2. El usuario clica en una tarea entre todas las mostradas. 3. Aparece en pantalla la información detallada de la tarea. 4. Clicando en “cerrar” el usuario puede volver a la pantalla anterior

ID	8
Nombre	Mover tarjeta
Descripción	Se arrastra una tarea de una etapa a otra, y se actualiza su información al soltarla.
Actor principal	Actor en vista web
Precondición	El usuario ha iniciado sesión.
Postcondición	La tarea está asociada a otra etapa distinta a la original.
Escenario de éxito	1. El usuario selecciona un tablero en la pantalla inicial de selección. 2. El usuario hace clic y mantiene el botón pulsado sobre una tarea. 3. El usuario mueve la tarea de la etapa original a otra cualquiera, y suelta el botón izquierdo del ratón. 4. La tarea desde ese momento pertenece a la etapa destino.
Extensiones	Si se arrastra la tarea hasta la misma etapa original, no se realiza ningún cambio en el sistema.

ID	9
Nombre	Crear tarjeta
Descripción	Crea una tarjeta nueva en el sistema
Actor principal	Actor en consola de administrador
Precondición	El actor ha iniciado la consola de administración
Postcondición	Existe una tarjeta nueva en la base de datos
Escenario de éxito	1. El actor selecciona “tareas” en el menú principal. 2. El actor selecciona “crear tarea” en el segundo menú. 3. El actor introduce uno a uno los distintos datos del nuevo usuario. 4. Tras hacer clic en “aceptar” en la última ventana de inserción de datos, se crea el nuevo usuario en el sistema. 5. Se muestra un mensaje de éxito. 6. Al clicar en “aceptar” se vuelve al menú principal.
Extensiones	4.b: si alguno de los datos numéricos es incorrecto, se muestra un mensaje indicando error en el formato. 4.c: si tiene lugar algún fallo durante la inserción de datos, se muestra un mensaje indicando que no se ha creado la tarea.

ID	10
Nombre	Eliminar tarjeta
Descripción	Elimina una tarea (tarjeta) del sistema.
Actor principal	Actor en web
Precondición	El usuario ha iniciado sesión.
Postcondición	La tarea seleccionada para su eliminación ya no existe.
Escenario de éxito	1. El usuario selecciona un tablero y accede a su vista. 2. El usuario hace clic secundario sobre una tarea. 3. El usuario selecciona la opción “eliminar tarea”. 4. Se elimina la tarea del sistema y desaparece de la vista.

ID	11
Nombre	Reconocer marcador.
Descripción	Muestra la información de una etapa al leer un marcador.
Actor principal	Actor en app.
Precondición	El actor ha iniciado sesión.
Postcondición	Se muestra en pantalla la información de una etapa.
Escenario de éxito	1. El usuario clica en “camera mode” en la pantalla de selección de tableros. 2. El usuario coloca un marcador en el campo de visión de la cámara. 3. Se muestra en pantalla el nombre y descripción de una etapa, y debajo sus tareas.
Extensiones	3.b: si el marcador reconocido no está asociado a ninguna etapa, se muestra un mensaje “NO ASOCIADO” en pantalla. 3.c: si no se tiene permiso para acceder a la etapa reconocida, se muestra un mensaje indicando “NO AUTORIZADO” en pantalla.

ID	12
Nombre	Crear usuario.
Descripción	Crea un usuario nuevo en el sistema.
Actor principal	Actor en consola de administración.
Precondición	El actor ha iniciado la consola de administración.
Postcondición	Existe un usuario nuevo en el sistema.
Escenario de éxito	1. El actor selecciona “usuarios” en el menú principal. 2. El actor selecciona “crear usuario” en el segundo menú. 3. El actor introduce uno a uno los distintos datos del nuevo usuario. 4. Tras hacer clic en “aceptar” en la última ventana de inserción de datos, se crea el nuevo usuario en el sistema. 5. Se muestra un mensaje de éxito. 6. Al clicar en “aceptar” se vuelve al menú principal.
Extensiones	4.b: si alguno de los datos numéricos es incorrecto, se muestra un mensaje indicando error en el formato. 4.c: si tiene lugar algún fallo durante la inserción, se muestra un mensaje indicando que no se ha insertado el nuevo usuario.

ID	13
Nombre	Editar usuario.
Descripción	Modifica la información de un usuario existente.
Actor principal	Actor en consola de administración.
Precondición	El actor ha iniciado la consola de administración.
Postcondición	Los datos del usuario elegido han cambiado.
Escenario de éxito	1. El actor selecciona “usuarios” en el menú principal. 2. El actor selecciona “modificar usuario” en el segundo menú. 3. El actor introduce uno a uno los distintos datos del nuevo usuario. 4. Tras hacer clic en “aceptar” en la última ventana de inserción de datos, se editan los datos del usuario buscándolo por su identificador. 5. Se muestra un mensaje de éxito. 6. Al clicar en “aceptar” se vuelve al menú principal.
Extensiones	4.b: si alguno de los datos numéricos es incorrecto, se muestra un mensaje indicando error en el formato. 4.c: si tiene lugar algún fallo durante la inserción de datos, se muestra un mensaje indicando que no se ha editado el usuario.

ID	14
Nombre	Eliminar usuario
Descripción	Elimina un usuario de la base de datos
Actor principal	Actor en consola de administrador
Precondición	El usuario ha iniciado la consola de administrador
Postcondición	El usuario indicado ya no existe en el sistema
Escenario de éxito	1. El usuario clica en “usuarios” en el menú principal. 2. El usuario clica en “eliminar un usuario” en el 2º menú. 3. El usuario introduce el ID del usuario a eliminar y hace clic en “aceptar” 4. Se elimina el usuario y se muestra un mensaje de éxito. 5. Tras clicar en “aceptar” se vuelve al menú principal.
Extensiones	4.b: si el usuario no existe, se muestra un mensaje de error. 4.c: si el ID introducido no tiene formato de número, se muestra un mensaje de error indicando que se ha escrito algo no válido.

ID	15
Nombre	Crear equipo
Descripción	Crea un nuevo equipo de trabajo sin empleados asociados
Actor principal	Actor en consola de administrador
Precondición	El usuario ha iniciado la consola de administrador
Postcondición	Existe un nuevo equipo en la base de datos con los datos proporcionados.
Escenario de éxito	1. El usuario clica en “equipos” en el menú principal. 2. El actor selecciona “crear un equipo” en el segundo menú. 3. El actor introduce uno a uno los datos del nuevo equipo. 4. Tras hacer clic en “aceptar” en la última ventana de inserción de datos, se crea el nuevo equipo en el sistema. 5. Se muestra un mensaje de éxito. 6. Al clicar en “aceptar” se vuelve al menú principal
Extensiones	4.b: si alguno de los datos numéricos es incorrecto, se muestra un mensaje indicando error en el formato. 4.c: si tiene lugar algún fallo durante la inserción, se muestra un mensaje indicando que no se ha insertado el nuevo equipo.

ID	16
Nombre	Eliminar equipo
Descripción	Elimina del sistema un equipo de trabajo
Actor principal	Actor en consola de administrador
Precondición	El usuario ha iniciado la consola de administrador
Postcondición	El equipo indicado ya no existe en la base de datos
Escenario de éxito	1. El usuario clica en “equipos” en el menú principal. 2. El usuario clica en “eliminar un equipo” en el 2º menú. 3. El usuario introduce el ID del equipo que quiere eliminar y hace clic en “aceptar” 4. Se elimina el equipo y se muestra un mensaje de éxito. 5. Tras clicar en “aceptar” se vuelve al menú principal.
Extensiones	4.b: si dicho equipo no existe, se muestra un mensaje de error. 4.c: si el ID introducido no tiene formato de número, se muestra un mensaje de error indicando que se ha escrito algo no válido.

ID	17
Nombre	Dar permisos
Descripción	Da permiso de acceso a un tablero a un equipo de trabajo
Actor principal	Actor en consola de administrador
Precondición	El usuario ha iniciado la consola de administrador
Postcondición	El equipo indicado puede acceder a la información del tablero proporcionado.
Escenario de éxito	1. El usuario clica en “equipos” en el menú principal. 2. El actor selecciona “conceder permisos de acceso” en el segundo menú. 3. El actor introduce el ID del equipo que quiere autorizar 4. El actor introduce el ID de un tablero existente 5. Tras hacer clic en “aceptar” en la segunda ventana de escritura, se crea una nueva entrada en la tabla de permisos. 6. Se muestra un mensaje de éxito. 7. Al clicar en “aceptar” se vuelve al menú principal
Extensiones	6.b: si alguno de los dos datos no es un número, se muestra un mensaje indicando error en el formato 6.c: si tiene lugar algún fallo durante la creación de permisos, se muestra un mensaje indicando que no se ha podido autorizar.

ID	18
Nombre	Quitar permisos
Descripción	Revoca el permiso de acceso de un equipo a un tablero
Actor principal	Actor en consola de administrador
Precondición	El usuario ha iniciado la consola de administrador
Postcondición	El equipo indicado no puede ver ni modificar la información del tablero indicado.
Escenario de éxito	1. El usuario clica en “equipos” en el menú principal. 2. El actor selecciona “revocar permisos de acceso” en el segundo menú. 3. El actor introduce el ID del equipo que quiere desautorizar 4. El actor introduce el ID de un tablero existente 5. Tras hacer clic en “aceptar” en la segunda ventana de escritura, se elimina la entrada de la tabla de permisos. 6. Se muestra un mensaje de éxito. 7. Al clicar en “aceptar” se vuelve al menú principal

Extensiones	6.b: si alguno de los dos datos no es un número, se muestra un mensaje indicando error en el formato 6.c: si tiene lugar algún fallo durante la eliminación del permiso, se muestra un mensaje indicando que no se ha podido realizar.