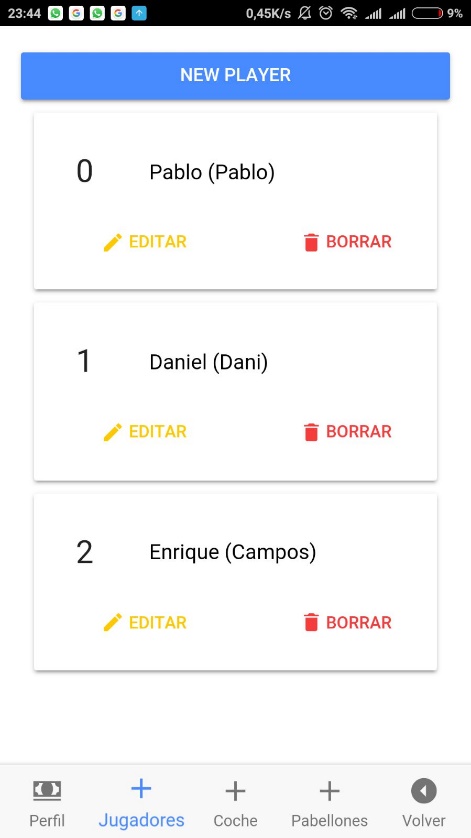
****

Documentación TeamRide

Teamride | ULE

Documentación aplicación ingeniería de software 2

DANIEL ÁLVAREZ FERNández

enrique campos martínez

Pablo Ángel Álvarez Fernández

2018

Índice

**[1.](#_Toc516182473)****[Descripción de la funcionalidad final](#_Toc516182473)** [3](#_Toc516182473)

[**2.** **Metodología** 3](#_Toc516182474)

[**2.1.** **Metodología usada** 3](#_Toc516182475)

[**2.2.** **División de tareas** 3](#_Toc516182476)

[**3.** **Fases del trabajo** 4](#_Toc516182477)

[**3.1.** **Backlog completo** 4](#_Toc516182478)

[ID1: Base de Datos 4](#_Toc516182479)

[ID2: Mostrar equipos 4](#_Toc516182480)

[ID3: Registro de equipo 4](#_Toc516182481)

[ID4: Mostrar coches de equipos 4](#_Toc516182482)

[ID5: Eliminar coches de equipos 4](#_Toc516182483)

[ID6: Mostrar información jugadores 4](#_Toc516182484)

[ID7: Eliminar jugadores 4](#_Toc516182485)

[ID8: realizar viaje 5](#_Toc516182486)

[ID9: login 5](#_Toc516182487)

[ID10: Actualizar información 5](#_Toc516182488)

[ID11: Editar coches de equipos 5](#_Toc516182489)

[ID12: Editar jugadores 5](#_Toc516182490)

[ID13: Editar pabellones 5](#_Toc516182491)

[ID14: Calcular deudas 5](#_Toc516182492)

[ID15: Mostrar Histórico 6](#_Toc516182493)

[ID16: Orientación a objetos back-end 6](#_Toc516182494)

[ID18: Pagar deuda 6](#_Toc516182495)

[**3.2.** **Sprint 1** 6](#_Toc516182496)

[3.2.1. Pila de backlog implementada 6](#_Toc516182497)

[3.2.2. Reuniones 6](#_Toc516182498)

[**3.3.** **Sprint 2** 8](#_Toc516182499)

[3.3.1. Pila de backlog implementada 8](#_Toc516182500)

[3.3.2. Reuniones 8](#_Toc516182501)

[**3.4.** **Sprint 3** 8](#_Toc516182502)

[3.4.1. Pila de backlog implementada 8](#_Toc516182503)

[3.4.2. Reuniones 9](#_Toc516182504)

[**4.** **Pruebas de Caja blanca y Caja negra** 9](#_Toc516182505)

[**4.1.** **Caja blanca** 9](#_Toc516182506)

[**4.2.** **Caja negra** 11](#_Toc516182507)

[4.2.1. Caso 1: Nombre erróneo y contraseña errónea 11](#_Toc516182508)

[4.2.2. Caso 2: Email correcto y contraseña incorrecta 11](#_Toc516182509)

[4.2.3. Caso 3: Email incorrecto y contraseña correcta 12](#_Toc516182510)

[4.2.4. Caso 4: email correcto y contraseña correcta 12](#_Toc516182511)

[**5.** **Control de versiones** 13](#_Toc516182512)

[**5.1.** **Commits a master** 14](#_Toc516182513)

[**5.2.** **Commits a development** 14](#_Toc516182514)

[**5.3.** **Commits a OOandbugFix:** 15](#_Toc516182515)

# **Descripción de la funcionalidad final**

La aplicación desarrollada hasta el momento cumple con estas funcionalidades:

* Contiene un login funcional que comprueba que quien se loguea es un usuario real de la aplicación con un email y contraseña válido.
* Un registro que permite tanto agregar un administrador a un equipo ya existente como crear un nuevo equipo con su respectivo administrador que sería el que añadiría todos los demás datos de su equipo.
* Permite ver la lista de jugadores de un equipo y la deuda que estos tienen.
* Permite crear un nuevo viaje hacia un pabellón seleccionando los coches, jugadores y destino al que se va a ir.
* Muestra la lista de los viajes ya introducidos, es decir, el destino y la fecha de cuando se realizó, pudiendo ver en más detalles los datos de este, mostrando así los jugadores y coches usados en el viaje.
* En cuanto a la administración del equipo se divide en tres partes:
  + Jugadores: se pueden ver los jugadores que hay, crear nuevos, modificarlos o eliminarlos.
  + Coches: se pueden ver los coches que tiene disponible cada equipo pudiendo agregar nuevos, modificar los datos de los existentes o eliminar alguno.
  + Pabellones: muestra los pabellones que el equipo a añadido y la distancia a la que están, pudiendo añadir editar y borrar los mismos.
* También contiene un botón de deslogueo.

# **Metodología**

## **Metodología usada**

Para la realización de este proyecto se ha optado por el uso de la metodología Scrum, debido a su simpleza y facilidad de uso. Aunque ni el proyecto ni el equipo era muy extenso como para rentabilizar y notar una mejoría en el tiempo y calidad del desarrollo.

## **División de tareas**

Para simular scrum, se han dividido las tareas de Scrum Master, Product Owner y Equipo de Desarrollo. Quedando así:

* En el papel de Product Owner queda Pablo.
* En el de Scrum Master queda Enrique
* Como equipo de desarrollo Daniel.

# **Fases del trabajo**

## **Backlog completo**

### ID1: Base de Datos

Se quiere almacenar todos los datos en una base MySQL para poder almacenar, sacar y actualizar la información.

**Pruebas de aceptación:** se puede acceder a la bbdd. Se insertan, seleccionan, actualizan y borran los datos. Esté siempre accesible.

### ID2: Mostrar equipos

Como usuario sin registrar quiero ver los equipos para poder seleccionar uno.

**Pruebas de aceptación:** se muestran todos los equipos.

### ID3: Registro de equipo

Como usuario sin registrar quiero poder añadir un equipo para pertenecer a él, en caso de que todavía no haya sido insertado.

**Pruebas de aceptación:** el nuevo equipo queda guardado en la base de datos

Dependencias:

### ID4: Mostrar coches de equipos

Como jugador quiero ver los coches de mi equipo para seleccionarlos a la hora de realizar un viaje.

**Pruebas de aceptación:** se muestran todos los coches pertenecientes al equipo.

Dependencias: base de datos

### ID5: Eliminar coches de equipos

Como jugador quiero poder eliminar los coches del equipo al que pertenezco.

**Pruebas de aceptación:** el coche se borra correctamente de la base de datos.

Dependencias: Mostrar coches, base Datos.

### ID6: Mostrar información jugadores

Como jugador quiero ver la información del resto de jugadores para poder ver datos como las deudas.

### ID7: Eliminar jugadores

Como jugador quiero poder eliminar otros jugadores del equipo.

**Pruebas de aceptación:** el jugador se borra correctamente de la base de datos.

Dependencias: Mostrar jugadores, base Datos.

### ID8: realizar viaje

Como jugador quiero poder añadir un nuevo viaje que se vaya a realizar para que se calculen automáticamente los gastos del mismo.

**Pruebas de aceptación:** se seleccionan correctamente coches, jugadores y el pabellón. Se agrega dicho viaje al histórico. Se guardarán los datos de los coches y jugadores que realizan el viaje.

Dependencias: Mostrar los jugadores, Mostrar coches, mostrar Pabellones.

### ID9: login

Como usuario ya registrado quiero una página de Login para acceder a la aplicación.

**Pruebas de aceptación:** se emplea usuario y contraseña para acceder. La contraseña será de al menos cinco caracteres.

Dependencias: Base de Datos.

### ID10: Actualizar información

Como usuario quiero poder actualizar información del equipo u otros jugadores para poder corregir cualquier error.

Dependencias: que estén todos los métodos de administración.

### ID11: Editar coches de equipos

Como jugador quiero poder variar los datos de los coches del equipo, por si se produce algún cambio en el modelo, consumo…

Dependencias: Mostrar coches.

### ID12: Editar jugadores

Como jugador quiero poder cambiar ciertos datos del jugador para poder corregir errores.

**Pruebas de aceptación:** la información se cambia correctamente. En caso de dejar en blanco algún parámetro no se actualizará.

Dependencias: Mostra jugadores

### ID13: Editar pabellones

Como jugador quiero cambiar los parámetros de los pabellones para actualizar o corregir.

**Pruebas de aceptación:** la información se cambia correctamente. En caso de dejar en blanco algún parámetro no se actualizará.

Dependencias: Mostrar Pabellones.

### ID14: Calcular deudas

Como jugador quiero que se calculen las deudas de los jugadores para facilitar los pagos.

**Pruebas de aceptación:** se calcula la deuda correctamente. A los conductores se le suma el gasto. A todos los jugadores se les resta el porcentaje que pagan.

Dependencias: realizar viaje.

### ID15: Mostrar Histórico

Como usuario quiero ver el historial de los viajes del equipo para organizar viajes futuros.

**Pruebas de aceptación:** se muestran un máximo de 10 viajes. Se puede ampliar este máximo. Para cada viaje se puede mostrar los jugadores y coches que fueron seleccionados.

Dependencias: Realizar viaje.

### ID16: Orientación a objetos back-end

Como desarrollador quiero un back-end orientado a objetos para mejorar su futuro mantenimiento.

### ID18: Pagar deuda

Como jugador quiero poder reflejar que he pagado para que no haya confusiones.

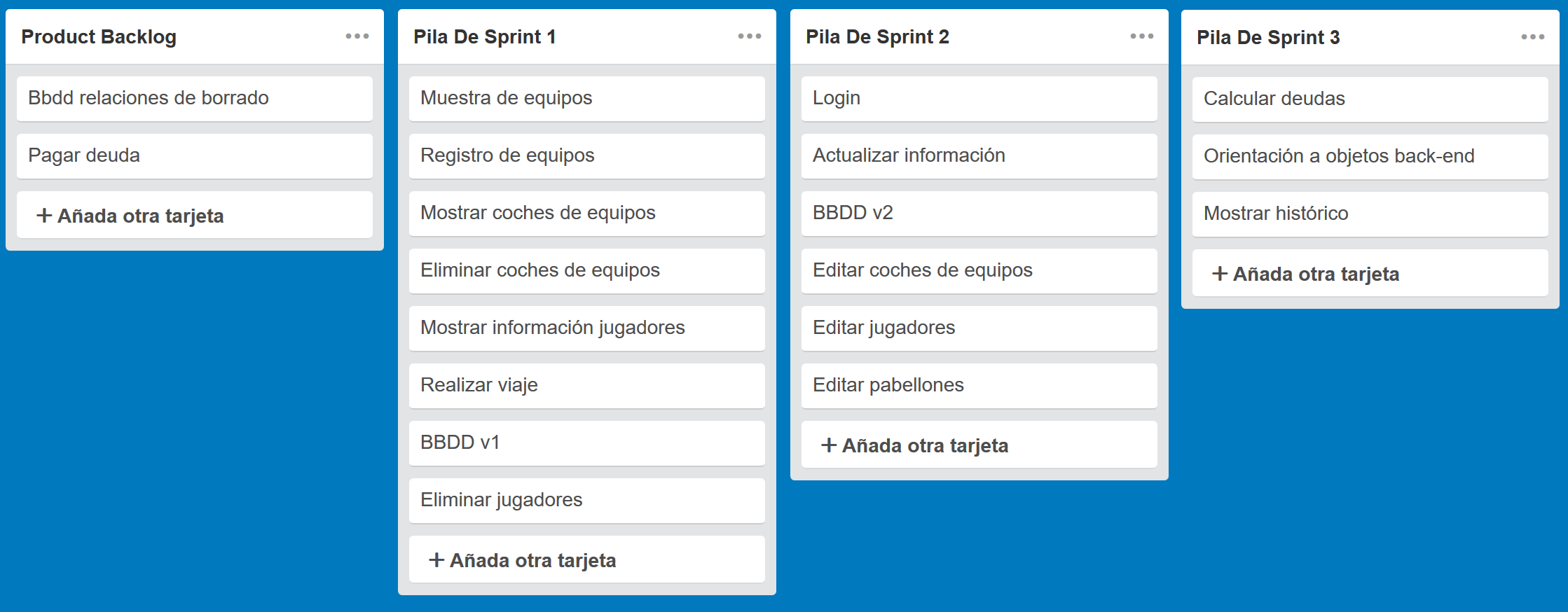
**Pruebas de aceptación:** se actualiza la deuda a cero para el jugador que marca como pagado.

Dependencias: realizar viaje, calcular deudas.

## **Sprint 1**

Este primer sprint duró el primer mes de desarrollo, del 10-04 al 10-05.

### Pila de backlog implementada



Para este sprint se detalla esta pila a implementar. Aunque este sprint está marcado por aprender el entorno a utilizar. Estas historias en primera instancia son sencillas, de hecho el login no se incluye.

### Reuniones

Primera reunión 10/04/2018

En esta reunión se concretan los objetivos iniciales del proyecto, como las obligaciones de cada uno:

* Pablo se encargará del back-end.
* Daniel se encargará del front-end
* Enrique de la base de datos y un apoyo para el resto de compañeros.

Asimismo, se divide los componentes del scrum, de forma que el Scrum Master es Enrique, Pablo es el Product Owner y como equipo de desarrollo queda Daniel. Como Stakeholders se basará en feedbacks dados por el equipo de balonmano al que Pablo pertenece.

Por último, se detalla la nomenclatura a usar a lo largo del proyecto:

* Las rutas serán todas en minúsculas, y en caso de tener más de una palabra se separará con "\_". Las rutas serán en plural y si después de esa ruta se encuentra un número se devolverá ese id en singular.
* Los nombres de variables y los métodos serán en minúscula, en caso de contener más de una palabra, ésta comenzará en mayúscula.
* Las variables globales comenzarán con "\_\_".
* Tanto los métodos como las variables serán en inglés.
* A poder ser, los métodos serán privados.
* El texto ha de estar correctamente indentado.
* Todas las variables utilizadas en el front-end se declararán como globales de su typeScript.
* Las variables que no cambien se declararán como constantes o const.
* Se intentará, en su mayor medida, no declarar variables como var.
* Los nombres de las tablas en la base de datos serán completamente en mayúsculas.

Segunda reunión 17/04/2018

Se establece el repositorio de GIT a utilizar.

En él se sube la primera versión de la Base de datos.

También se levanta el servidor node con dos peticiones.

Se marcan futuros hitos:

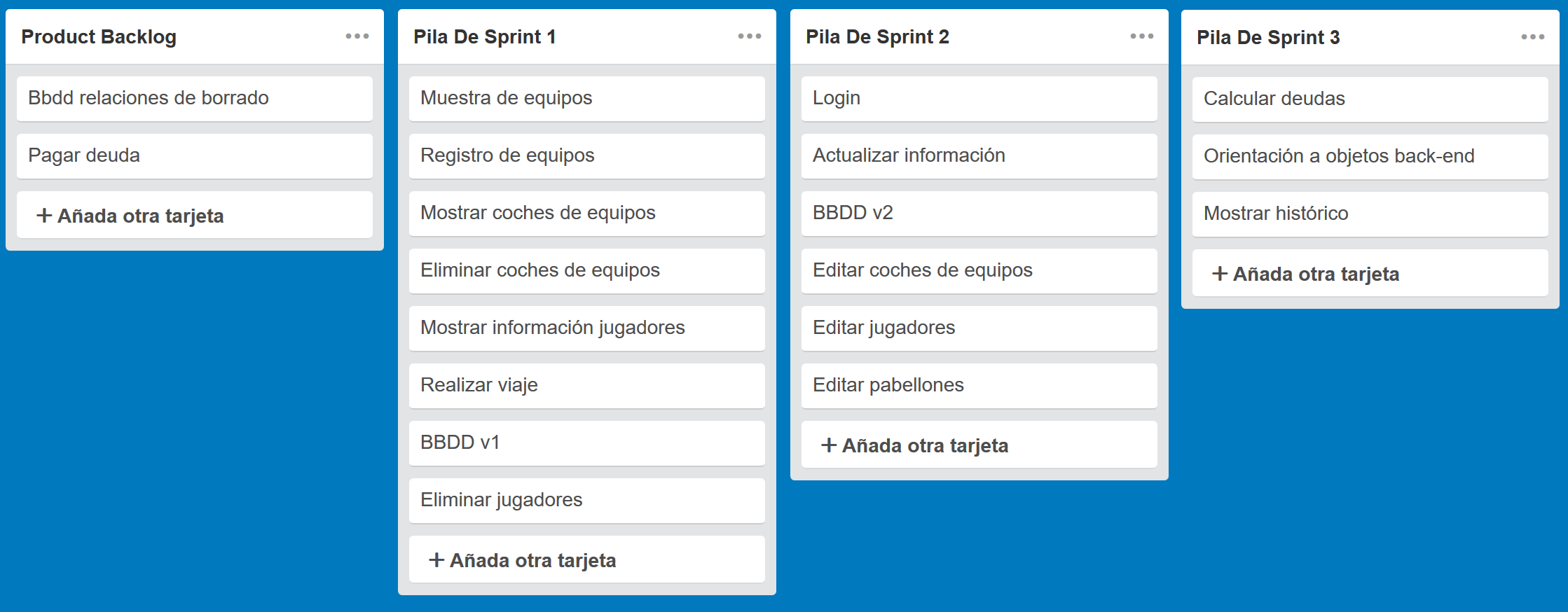
-Crear las historias de usuario a realizar en el proyecto.

-Avanzar en conocimientos de la tecnología de ionic.

## **Sprint 2**

El segundo sprint duró dos semanas de desarrollo, del 10-05 al 1-06.

### Pila de backlog implementada



Las historias a realizar se resumen en una ampliación de las anteriores. A destacar solamente el login y que se reescribe la bbdd.

### Reuniones

Tercera reunión 11/05/2018

Se prosigue con la implementación de las historias de usuario. En la que se actualizan partes de la base de datos, entre ellos el uso de forever para mantener el servidor siempre arriba.

Cuarta reunión 24/05/2018

Se han realizado las historias de usuario.

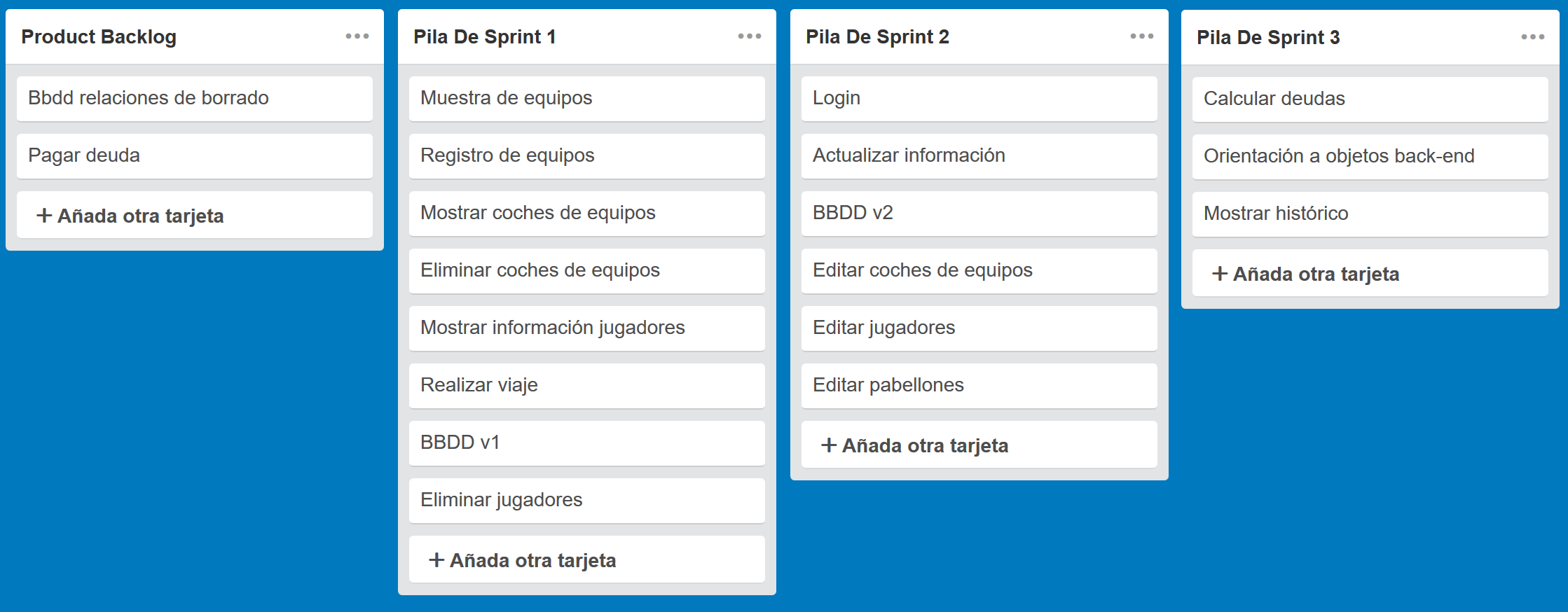
Con los cambios realizados en la base de datos y en el servidor se pretende terminar el controlador.

## **Sprint 3**

Con una duración de una semana de desarrollo, del 1-06 al 7-06.

Caracterizado por una “maratón de código” los últimos días.

### Pila de backlog implementada



Son pocas historias las que se implementan, pero muy costosas.

Además de realizar arreglos superficiales a la aplicación, como los estilos.

### Reuniones

Quinta reunión 1/06/2018

Se planifican las interfaces de administración.

Se establecen el uso de tokens para el login.

Sexta reunión 4/06/2018

Se establece realización y finalización tanto del apartado de visión de los viajes como de la creación de los mismos.

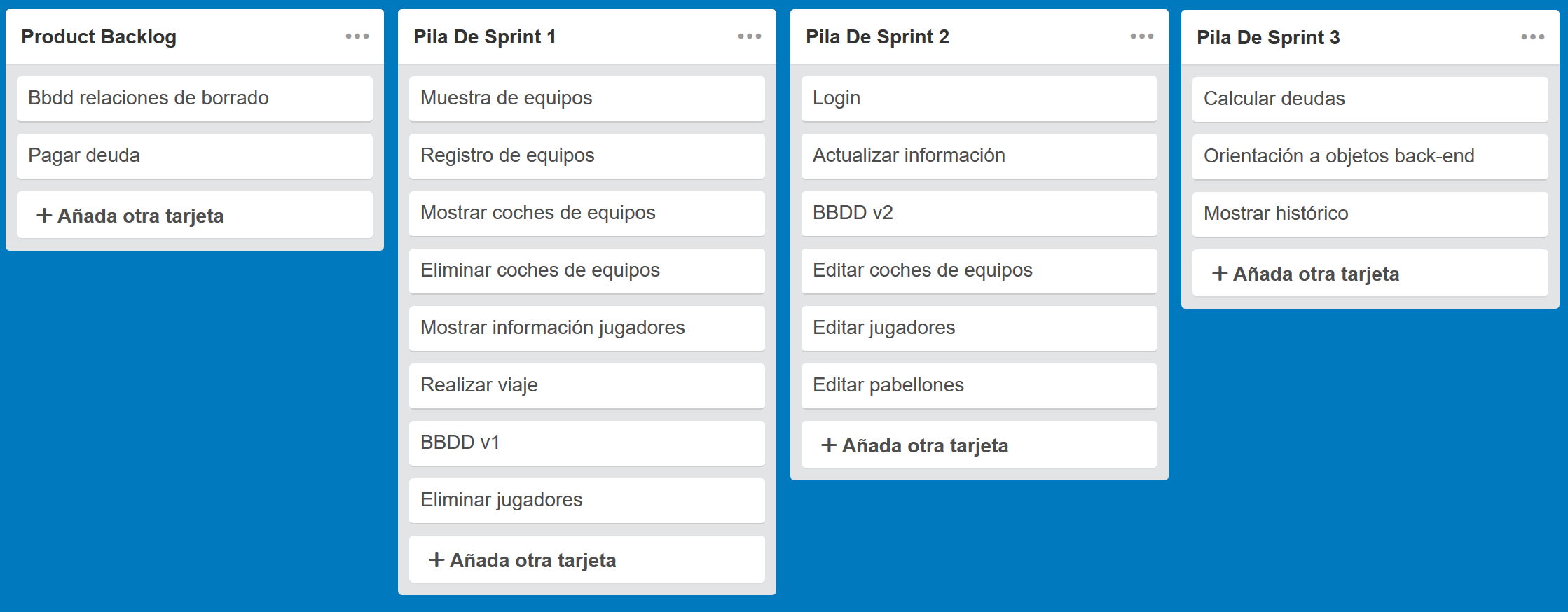
Se planificó que el back-end sería refactorizado para que estuviera orientado a objetos.

El front-end se reorganizará para que la distinción de las partes y el mantenimiento futuro sea más sencillo y tenga menos coste.

Séptima reunión 7/06/2018

Es como una reunión de retrospectiva del proyecto entero

Se realizan pruebas generales y se decide no terminar las historias de usuario siguientes.

****

La de pagar deudas por no tener suficiente feedback del equipo. La adición de las restricciones a la base de datos no ha dado suficiente tiempo ni conocimientos como para realizarlo.

# **Pruebas de Caja blanca y Caja negra**

## **Caja blanca**

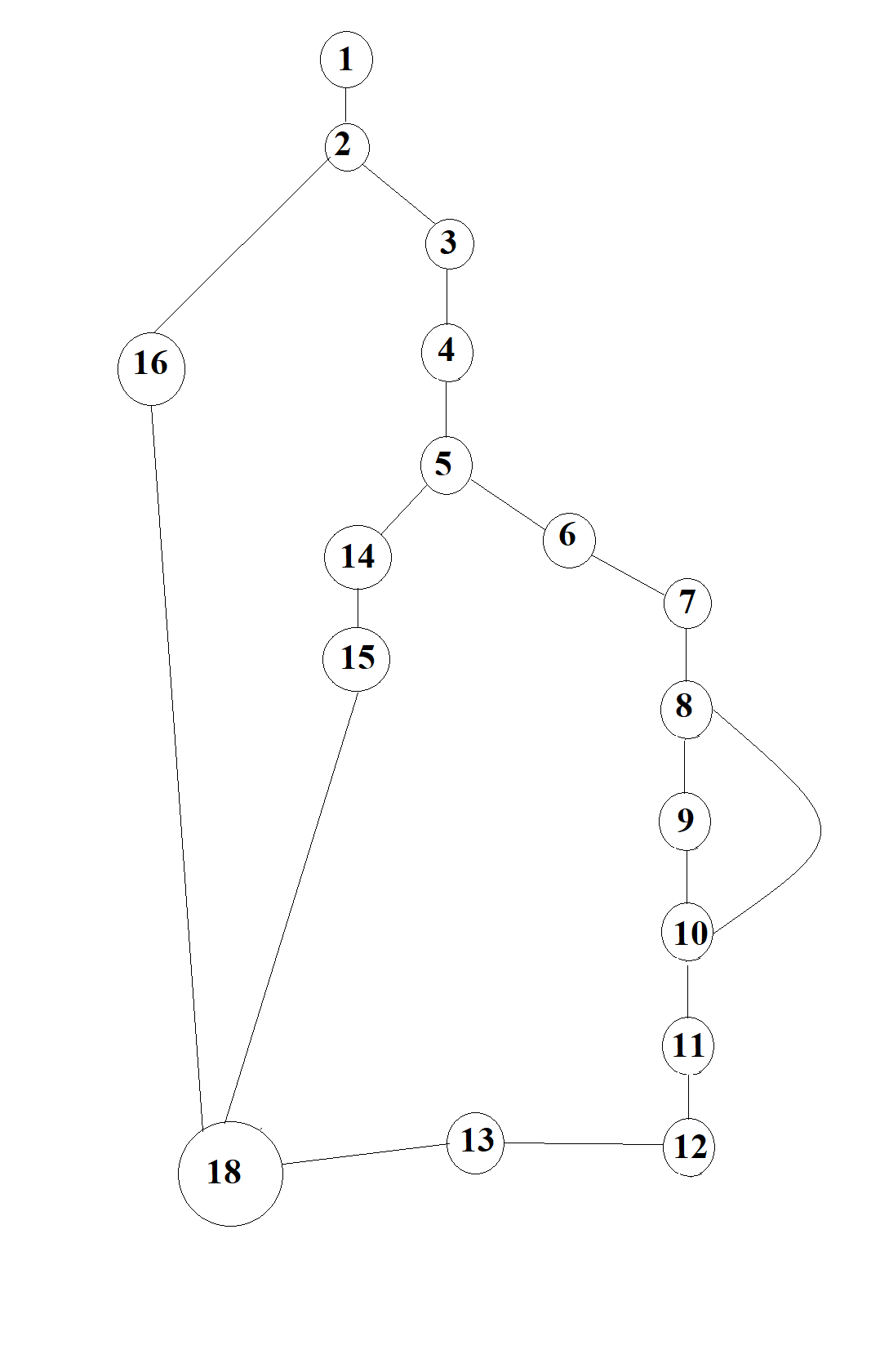
Código del que se ha hecho el esquema:





Solo se realiza el esquema de las instrucciones del código TypeScript, el código de acceso a la base de datos se omite porque solo son introducciones.

Esquema de la Caja blanca.



## **Caja negra**

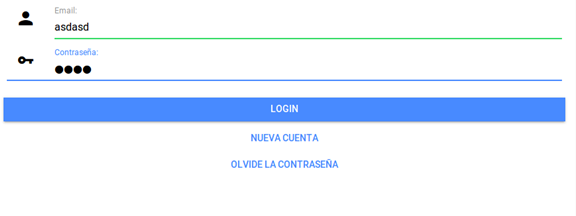
Ha sido realizado sobre el Login.

|  |  |  |
| --- | --- | --- |
| Condición | Caso valido | Caso no Valido |
| conectarse | Email y contraseña corresponde a un elemento de la base de datos | Email: correcto  Contraseña: incorrecto.  Email: incorrecto  Contraseña: correcto.  Email y contraseña incorrecto. |

### Caso 1: Nombre erróneo y contraseña errónea

Email: asdasd

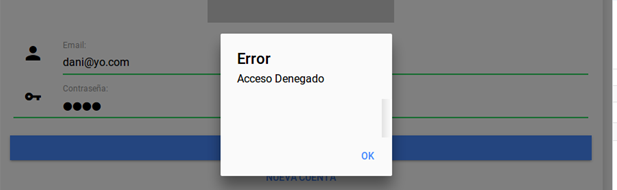
Contraseña asda



### Caso 2: Email correcto y contraseña incorrecta

Email: dani@yo.com

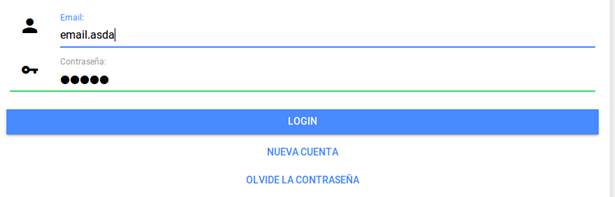
Contraseña: algo



### Caso 3: Email incorrecto y contraseña correcta

Email: email.asda

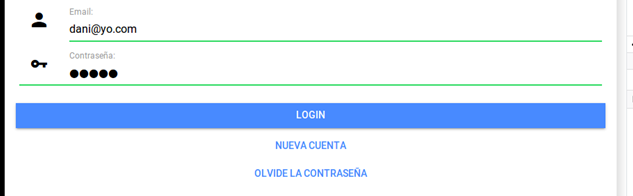
Contraseña Dani0



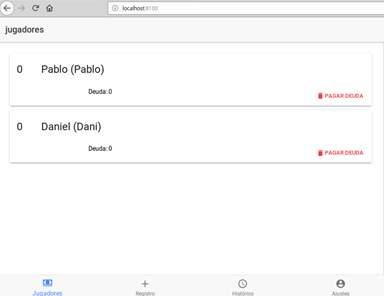
### Caso 4: email correcto y contraseña correcta

Email: dani@yo.com

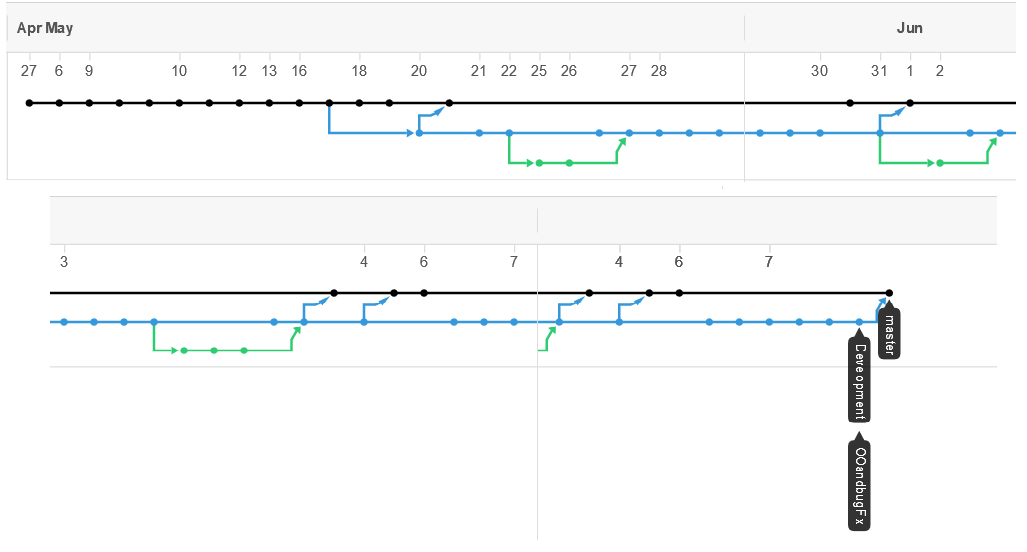
Contraseña: Dani0



Ya logueado:



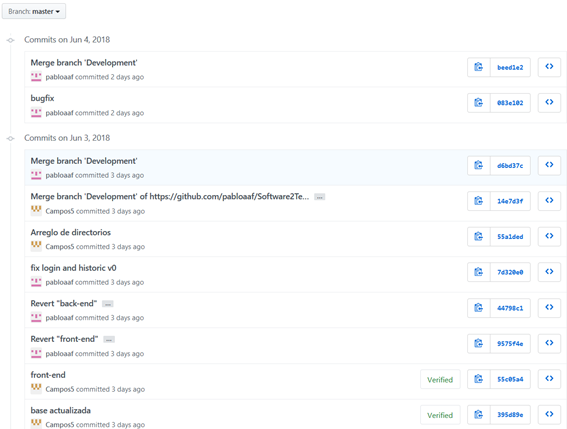
# **Control de versiones**



Se han realizado tres branch, uno que es el master, development, OOandbugFix, en el development se han realizado la mayoría de los cambios puesto que esta es la rama de pruebas y se hacen aquí para evitar problemas en el master y con pocos cambios para que no se produzcan errores en los commit subidos a la rama principal. En OOandbugFix se realizaron los pasos finales como la solución de uno de los métodos finales que es las deudas y la reorganización de archivos y orientación a objetos para que el proyecto fuera mucho más escalable, simple y entendible. Disminución de costes futuros para el mantenimiento.

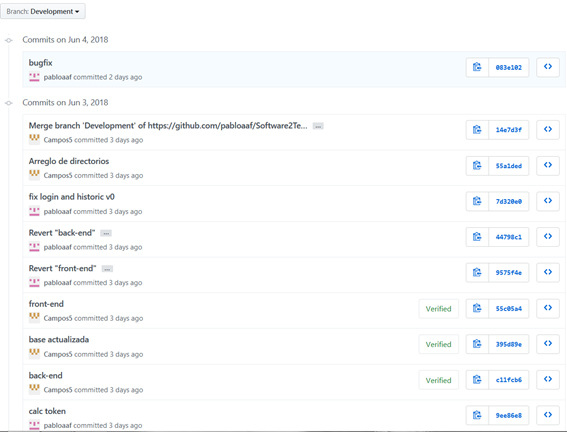
## **Commits a master**

Cambios ya completos que se subían para tener una versión completa del sistema, lista para ejecutar. También se subían los archivos de documentación realizados en cada reunión.



## **Commits a development**

Cambios subidos poco a poco a medida que se fueron realizando las tareas de los sprints, hay más subidas que tareas en los sprints pues cada vez que se hacía algo se subía para que todo el equipo tuviera acceso a estos cambios.



## **Commits a OOandbugFix:**

En estos commits se subieron las últimas versiones cambiando su estructura tras hacer refactorización orientado a objetos el back-end, creando la función de las Deudas en el back-end además de la simplificación del Front-end para que sea más entendible en el futuro.

