METODOLOGÍAS ÁGILES DE DESARROLLO DE SOFTWARE

Práctica 4: Sprint de desarrollo. Documentación

Alumnos

- Andres Tebar Moreno
- Brais Valencia García
- Raúl García Carrión

Introducción **3**

Historias de usuario (Sprint Backlog) **3**

Prioridad de tareas 3
Creación de proyectos para organización de tareas específicas 5
Administrador de equipo 7
Estado de tarea y buscar tarea 8
Añadir comentario a las tareas de proyecto 10
Pulir la interfaz de usuario y capacidad de edición de los datos del usuario 12
Tareas por Categorías 13
Añadir propiedades a proyectos y gestión de fechas 16

Funcionalidades implementadas **18**

Se expondrá una breve descripción para el usuario y una breve descripción técnica de cada funcionalidad implementada. 18
Añadir prioridad de tareas 18
Añadir un administrador de equipo 18
Creación de proyectos y tareas de proyecto 19
Añadir estado a las tareas y búsquedas de las mismas 20
Posibilidad de añadir comentarios a las tareas de proyectos 20
Posibilidad de editar los datos del usuario y pulir interfaz de usuario 21
Añadir clasificación de tareas por categorías 22
Añadir propiedades a los proyectos y gestión de fechas de finalización 22

Informe sobre la evolución del desarrollo **23**

Trello 23
GitHub 25

Informe sobre la sesión de pair programming **28**

Resultado de la retrospectiva **29**

Conclusiones 29
Siguiente Sprint 29

Introducción

En este documento explicaremos todo lo que hemos estado haciendo en la aplicación ToDoList durante un mes de desarrollo.

Iniciamos con las historias de usuario escogidas para el desarrollo del sprint y explicaremos en qué consiste cada una de ellas.
Continuaremos explicando las distintas funcionalidades que hemos implementado y que son visibles en nuestra aplicación.
Además, vamos a mostrar distintas capturas de pantalla de nuestros tableros de GitHub y Trello y explicar cómo han ido evolucionando a lo largo de las semanas de desarrollo.

Por otro lado, explicaremos como se ha desarrollado la sesión de pair programming que hicimos durante una de las semanas del sprint. En ella detallaremos como lo hemos hecho, las distintas sesiones que hizo cada uno y ventajas o desventajas que hemos podido ver.

Finalmente terminaremos con una pequeña reflexión sobre cómo ha ido este sprint y qué cosas podríamos mejorar o añadir para futuros desarrollos del proyecto.

Historias de usuario (Sprint Backlog)

Historias de usuario escogidas para el sprint (copiar la descripción, las condiciones de satisfacción y el borrador de interfaz de usuario tal y como aparecen en la Wiki)

Prioridad de tareas

Descripción y detalles:

Como usuario quiero poder indicar la prioridad de una tarea para saber cuál debería hacer primero.

- Asignar un valor de prioridad de 1 a 3.
- Al dar al botón *Crear tarea* se incluye en el formulario la posibilidad de asignar una prioridad a la tarea.
- Al dar al botón *Editar tarea* se incluye en el formulario la posibilidad de cambiar la prioridad en la tarea.
- Al listar la tarea se podrá visualizar la prioridad asignada.

Condición de satisfacción (COS):

- Cuando un usuario acceda a nueva tarea podrá elegir la prioridad que quiere darle a esta.
- Cuando un usuario acceda a editar una tarea podrá cambiar la prioridad que quiere darle a esta.

- Un usuario debe poder tener control absoluto sobre la prioridad asignada a cada una de sus tareas.
- Un usuario debe poder visualizar de forma correcta las prioridades de sus tareas

Aspecto interfaz de usuario:

Creación de proyectos para organización de tareas específicas

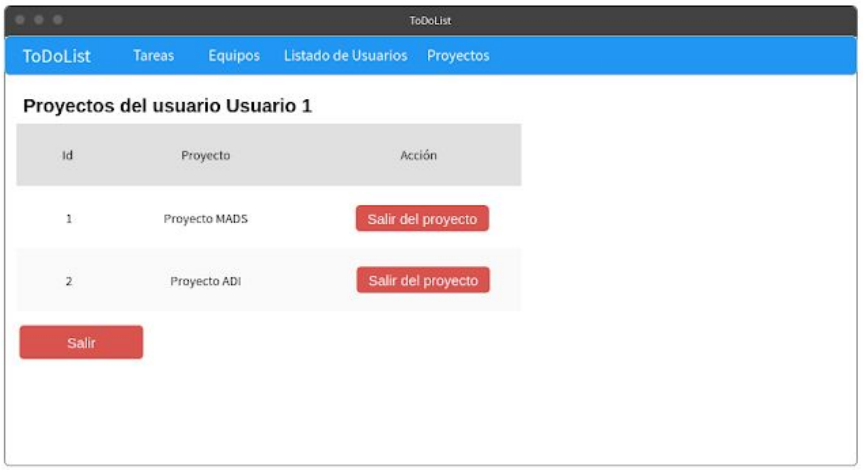
Descripción y detalles:

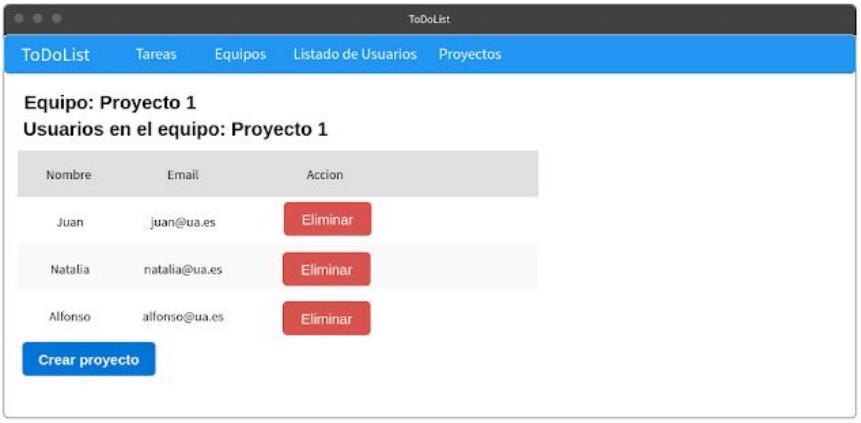
- Como administrador de equipo quiero poder crear proyectos a los que asociarles una o varias tareas.
- Desde la página del equipo correspondiente, el administrador del equipo podrá dar al botón `Crear proyecto` para crear un nuevo proyecto asociado al equipo.
- Desde cualquier página podremos dar al botón `Proyectos` en la barra del menú para acceder a mis proyectos.
- Desde la página del proyecto podremos crear una tarea dando a `Nueva Tarea` que forme parte de ese proyecto.

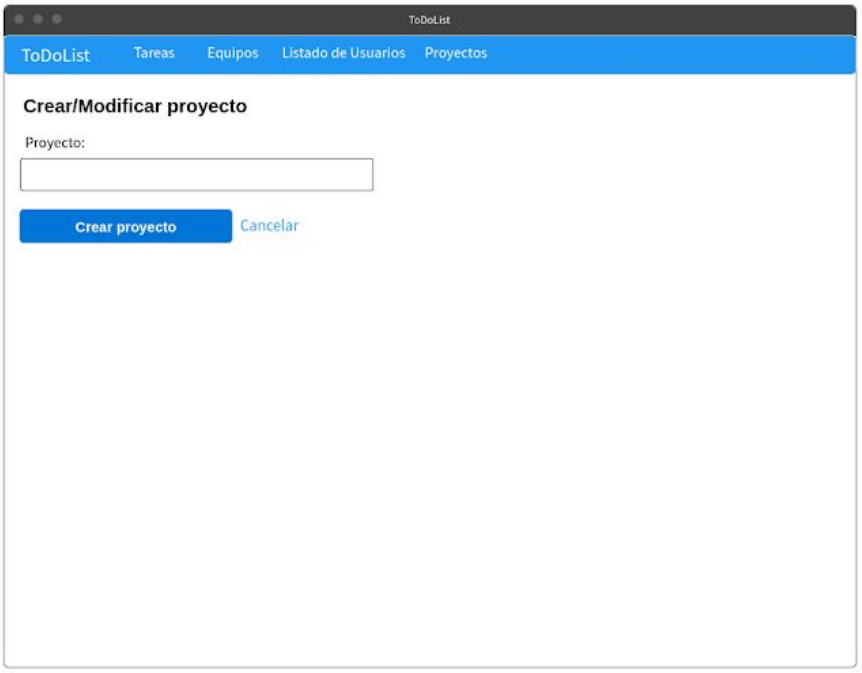
Condición de satisfacción (COS):

- Acceder como usuario a la web e ir, mediante el enlace de de la barra del menú, a la página *Proyectos.*
- Acceder al proyecto que tenemos asignado por pertenecer a un equipo y comprobar que podemos crear una tarea.
- Crear una tarea nueva en para ese proyecto, y comprobar que se ha creado correctamente.
- Salirse del equipo que tiene asociado un proyecto y comprobar que efectivamente ya no podemos acceder a él ni ver sus tareas.
- Borrar un proyecto de la base de datos y comprobar que se borra completamente de la base de datos y que no aparece en el listado

Aspecto interfaz de usuario:







Administrador de equipo

Descripción y detalles:

Como usuario que ha creado un grupo me convierto en su administrador, el cual tiene permisos para editarlo o eliminarlo, además puede eliminar de este equipo a otros usuarios.

- La figura del administrador del equipo se asignará automáticamente cuando se crea el equipo, no requerirá indicarlo explícitamente. El administrador del equipo no necesariamente tiene porqué pertenecer a él, puede que solamente quiera gestionarlo de manera externa, por lo que no se unirá automáticamente.

- Los botones, ya existentes, de **Editar Equipo** y **Eliminar Equipo** quedarán restringidos a la vista de todo aquel usuario del equipo que no sea administrador.

- En el listado de usuarios que pertenecen al equipo aparecerá, para el administrador del equipo, un botón **Eliminar** para quitar al usuario seleccionado de ese equipo.

Condición de satisfacción (COS):

- Creamos un nuevo equipo.

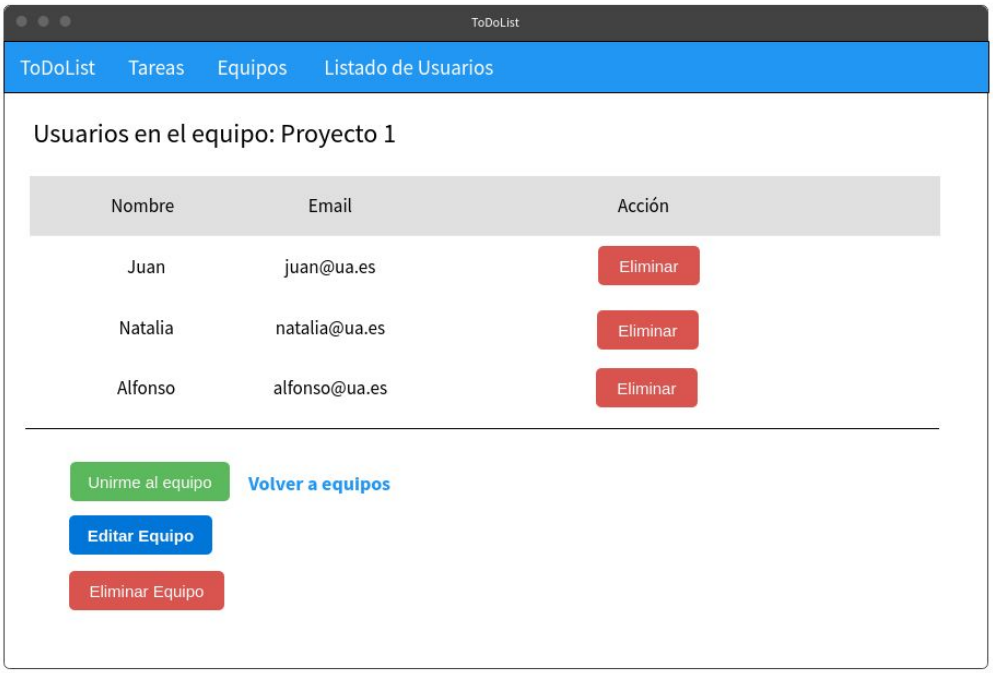
- En la página de ese equipo vemos como, al ser su administrador, podemos editar su nombre. Cambiamos el nombre.

- Desde otro usuario nos unimos al equipo. Y comprobamos como no se muestran las opciones de *eliminar* o *editar* el equipo, ya que no somos los administradores de este. Además tampoco podremos eliminar al otro usuario del equipo.

- De vuelta al equipo, otra vez con el usuario administrado, comprobamos que se puede eliminar al usuario que se acaba de unir. Lo eliminamos y vemos como desaparece del equipo.

- Eliminamos el equipo y vemos como este desaparece.

Aspecto interfaz de usuario:



Estado de tarea y buscar tarea

Descripción y detalles:

Como usuario registrado y con tareas pendientes para realizar, quiero poder asignar un estado a las tareas que tenga registradas en mi cuenta. Además podré agrupar u ordenar las tareas por estado, y podré buscar una tarea en concreto sabiendo el nombre.

- En la lista de tareas de cada usuario, tendremos una **lista desplegable** con varias opciones que determinarán el **estado de la tarea**.

- Se podrá indicar que las tareas en un **determinado estado** se muestren **primero**.

- Se podrán **filtrar** las tareas **por estado**. Mostrando, por ejemplo, solo las tareas *pendientes*.

- También existirá la posibilidad de **excluir** las tareas en determinados **estado**, mostrándose las tareas que no pertenezca a ese estado.

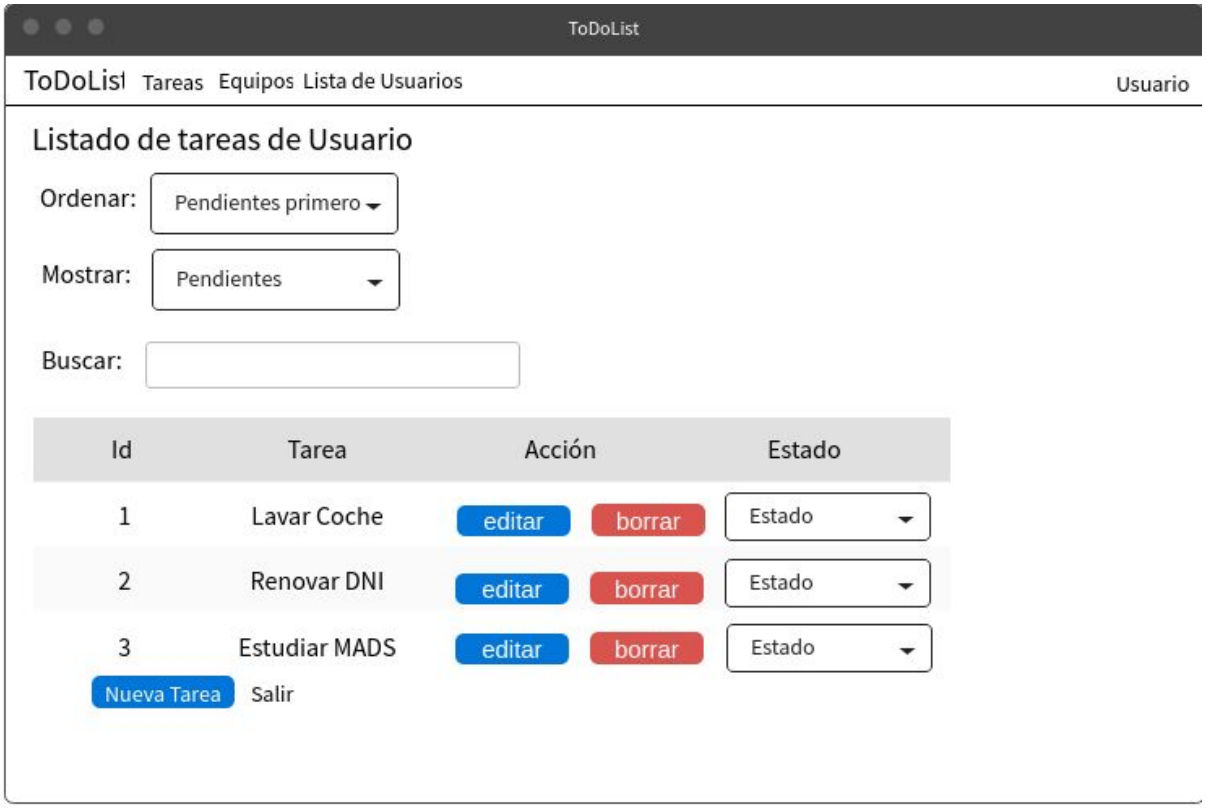
- Se podrá **buscar** una **tarea**, escribiendo en una barra de búsqueda su nombre.

- Cuando cambiemos el estado de alguna de las tareas, debe de **cambiar** también en la **base de datos** sin errores.

Condición de satisfacción (COS):

- Acceder a la lista de *tareas*.
- Añadir, por lo menos, el mismo número de tareas que de posibles estados se acepten.
- Cambiar el estado de las tareas para que cada una tenga un estado y que, como mínimo en una tarea, todos los estados se vean representados.
- Ordenar las tareas por estado. Probando con todos los estados posibles y comprobando que efectivamente la primera del listado corresponde.
- Filtrar las tareas por estado, probando todos los estados y comprobando que solamente se muestran las tareas que se encuentran en ese estado.
- Buscar cualquiera de las tareas creadas y comprobar que se muestra correctamente.

Aspecto interfaz de usuario:



Añadir comentario a las tareas de proyecto

Descripción y detalles:
Como usuario que tiene asociado un proyecto puedo hacer comentarios a las distintas tareas del propio proyecto para dar indicaciones o para informar de la evolución de la misma.

- En el listado de tareas del proyecto podremos acceder a cada una de ellas mediante un enlace en el nombre.

- Dentro del detalle de cada tarea, los usuarios del equipo, podrán añadir comentarios en la tarea para especificar ciertos aspectos o para incorporar cambios en la tarea.

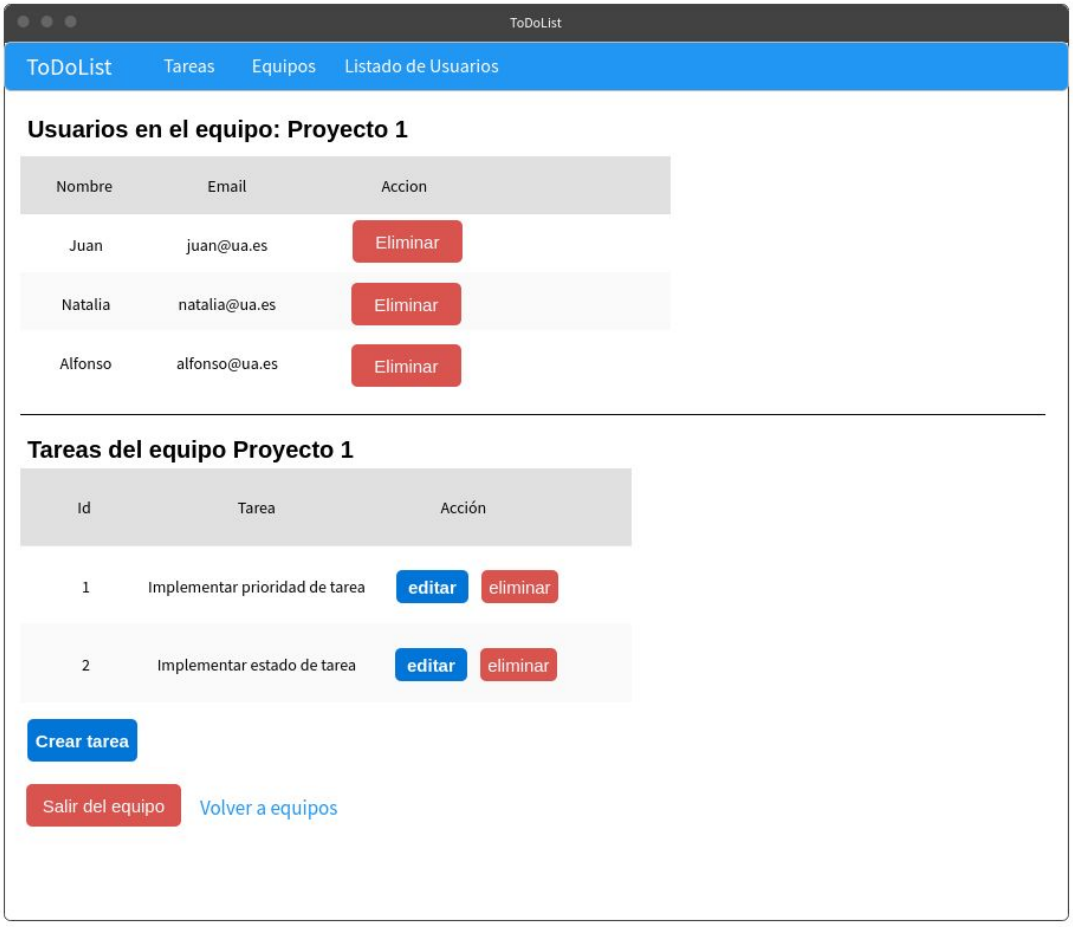
- Todos los usuarios del mismo proyecto podrán ver estos comentarios de la tarea.

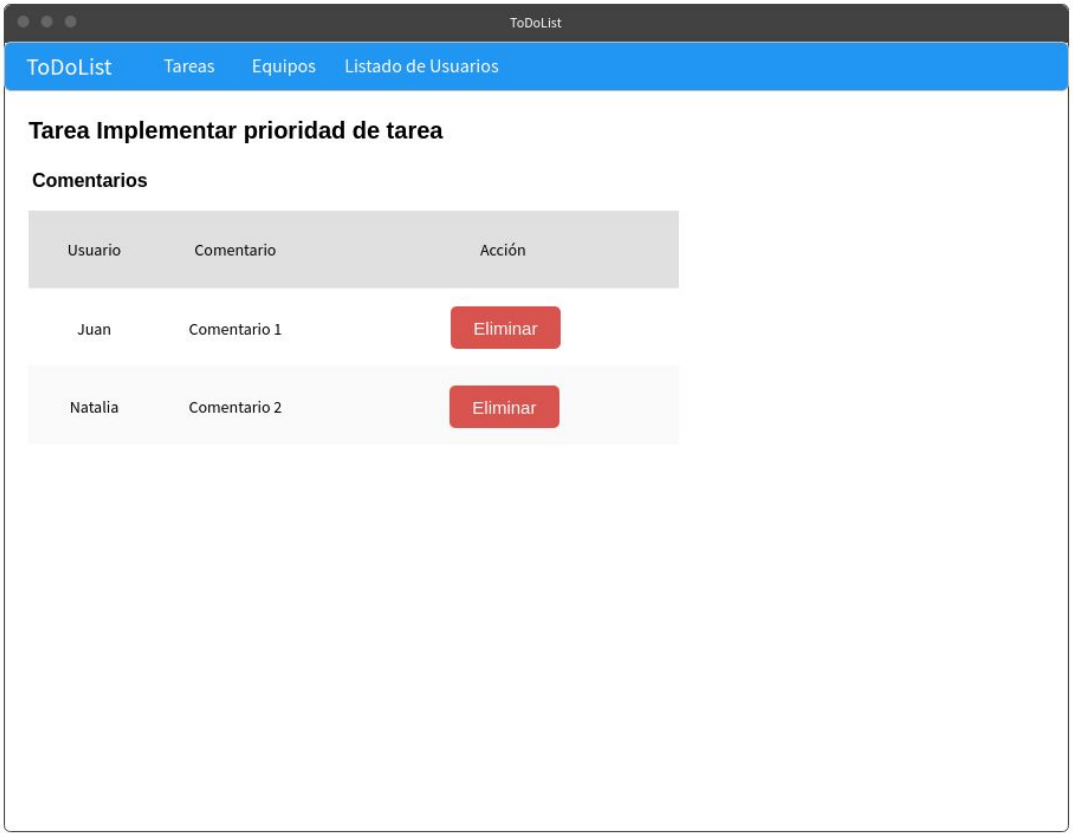
- Los usuarios que no estén asociados al proyecto no podrán ver estos comentarios

Condición de satisfacción (COS):

- En el listado de proyectos de un usuario podremos acceder a cada uno de los proyectos asociados al usuario
- Dentro de los proyectos tendremos un listado con todas las tareas del proyecto y podremos acceder al detalle de cada una de las tareas.
- En las propias tareas tendremos el nombre de la tarea y todos los comentarios hechos de esa tarea.
- Podremos escribir nuevos comentarios que se guardarán en la misma tarea.
- Si un usuario que no pertenece al proyecto intentará hacer un comentario a una tarea del mismo, no se le permitirá

Aspecto interfaz de usuario:





Pulir la interfaz de usuario y capacidad de edición de los datos del usuario

Descripción y detalles:
Como usuario o administrador me gustaría poder editar los datos de mi perfil.
Además, como usuario me gustaría eliminar mi cuenta si lo deseo.

- Implementar en la lista de tareas de usuario que no se muestren las tareas de proyectos asociadas al usuario.

- Implementar los cambios hechos en la lista de tareas del usuario también en la lista de tareas de un proyecto (estado de la tarea, filtros…)

- Mejorar la vista de perfil de usuario con un interfaz más agradable.

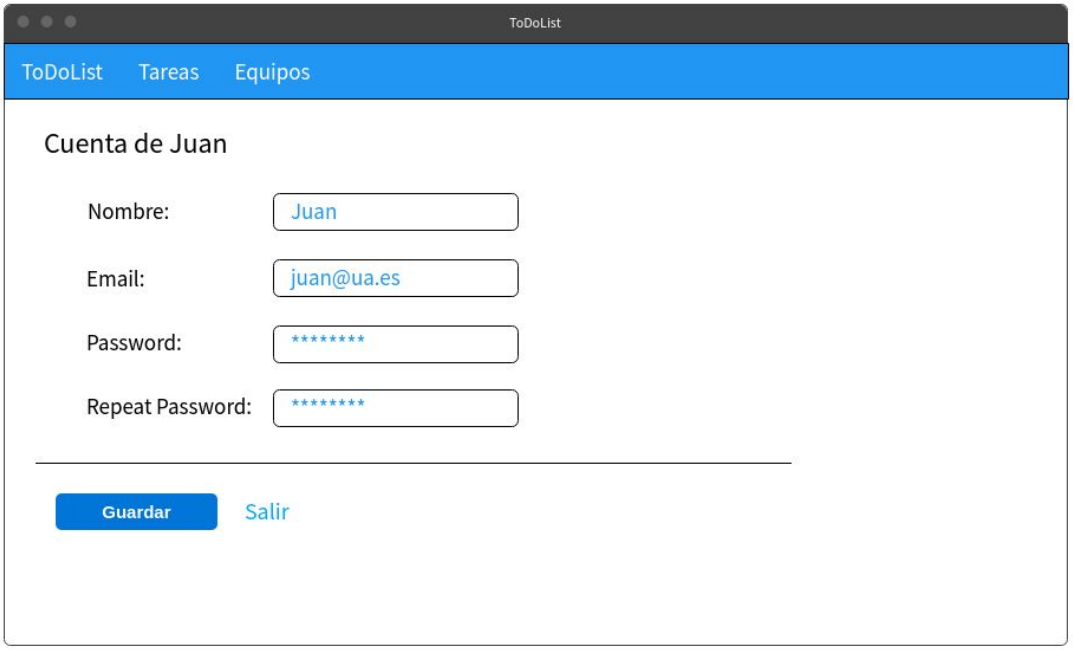
- Introducir un formulario con el que editar los datos del usuario

- Añadir la posibilidad de eliminar la cuenta.

Condición de satisfacción (COS):

- Acceder al perfil personal que contiene toda nuestra información sobre la cuenta.
- Como usuario se debe poder modificar cualquier atributo.
- Como usuario debe poderse apreciar una mejora visual respecto a la versión anterior.
- Como usuario administrador podremos ver y modificar cualquier atributo de un usuario.
- Como usuario se debe poder eliminar de forma permanente la cuenta en la aplicación web.

Aspecto interfaz de usuario:



Tareas por Categorías

Descripción y detalles:

Como usuario de la web podrás crear categorías.
Dentro de un proyecto se compartirán las categorías.
Como usuario se podrá organizar de forma efectiva mis tareas por categorías.

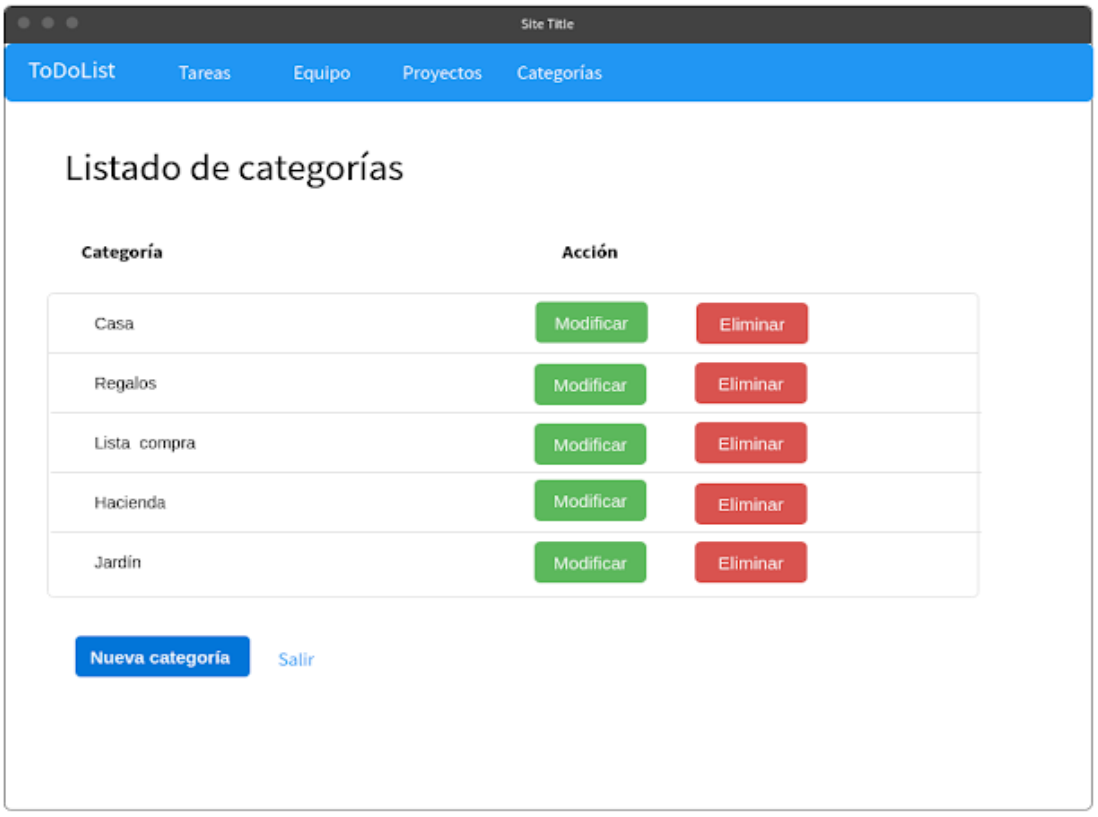
- Crear una o más categorías.

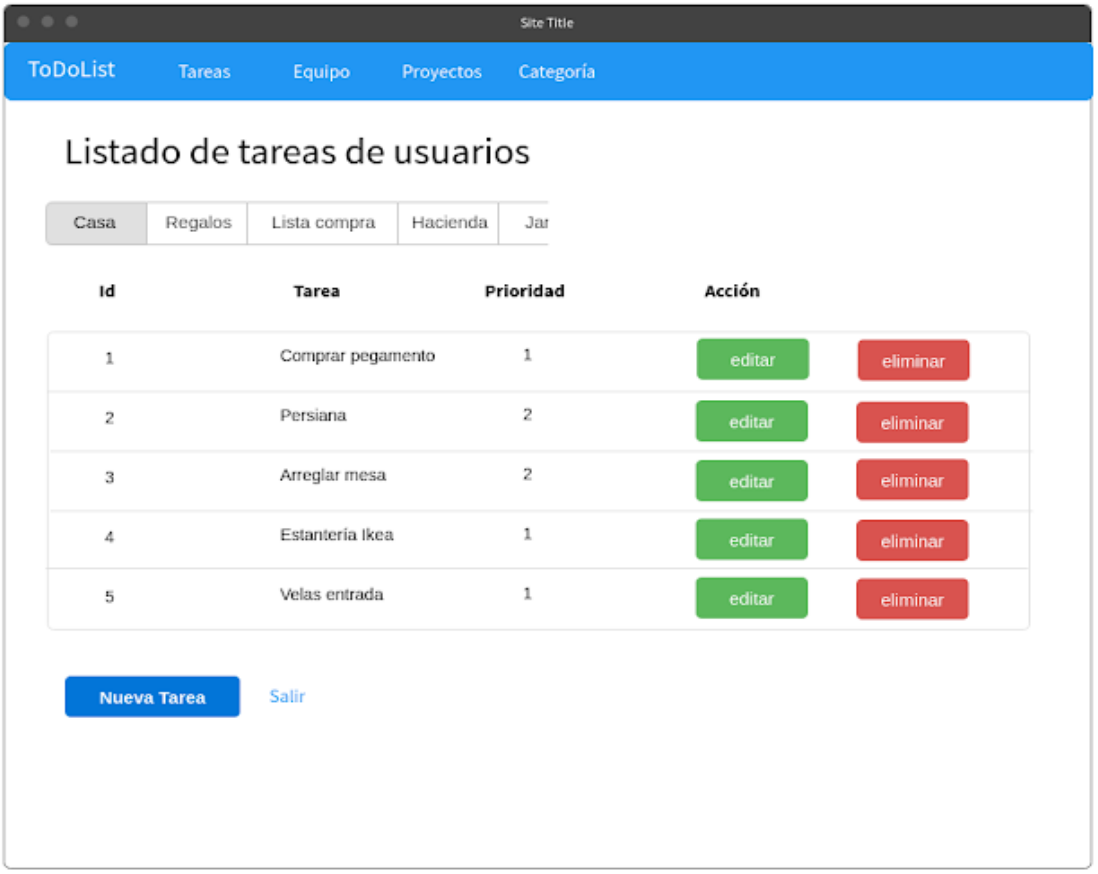
- Modificar una categoría ya creada. Independientemente de que tenga tareas asignadas.
- Eliminar una categoría que contenga o no tareas vinculadas. (Las tareas vinculadas pasarán automáticamente a “sin categoría”. No se borrarán de ninguna manera.
- Al dar al botón `Crear tarea` se incluye en el formulario la posibilidad de asignar una categoría a la tarea.
- Al dar al botón `Editar tarea` se incluye en el formulario la posibilidad de cambiar la categoría en la tarea.
- Al listar la tarea se podrá visualizar la categoría vinculada.

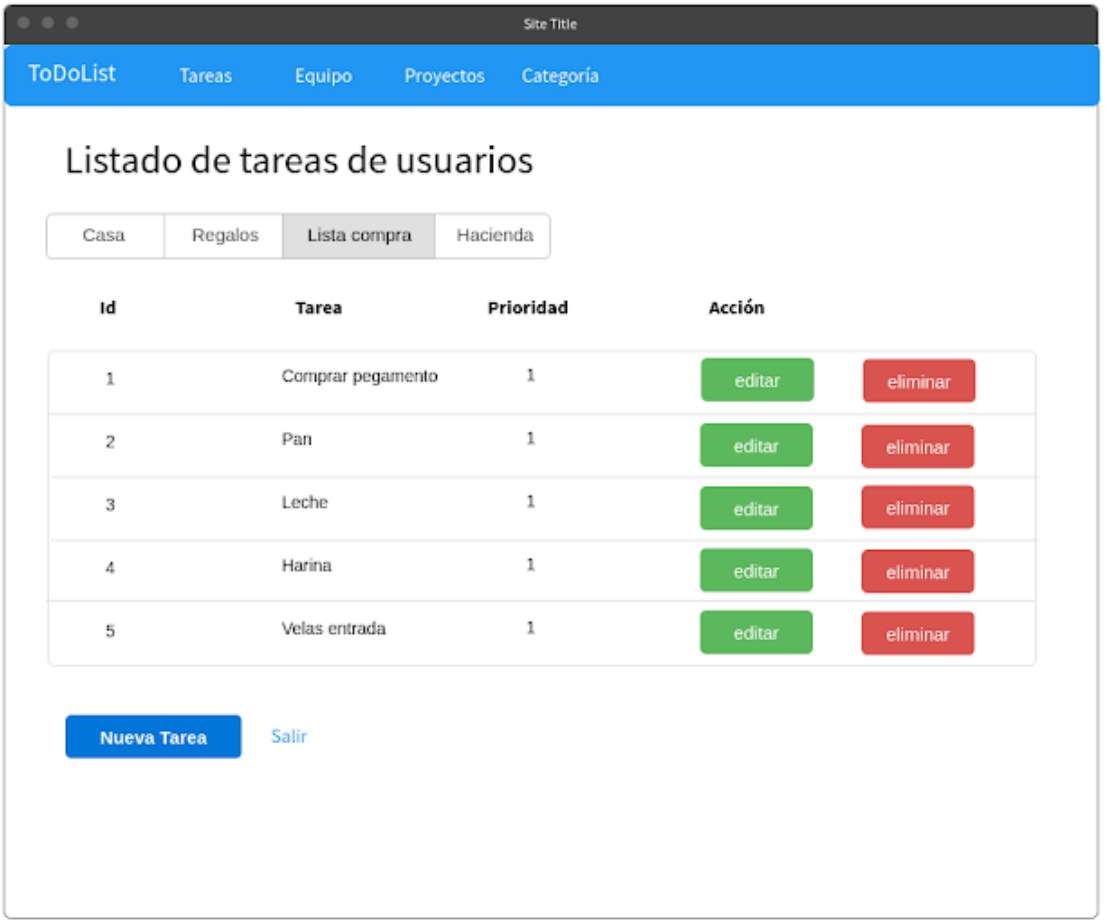
Condición de satisfacción (COS):

- Como usuario debe poder crear una categoría .
- Como usuario debe poder modificar una categoría .
- Como usuario debe poder eliminar una categoría .
- Como usuario podrá tener tareas vinculadas a una categoría o no.
- Como usuario debe poder visualizar y vincular tareas a categorías.
- Como miembro de un proyecto las categorías se comparten entre los integrantes del grupo.
- Como miembro de un proyecto se podrán realizar todas las funcionalidades de un usuario normal.

Aspecto interfaz de usuario:







Añadir propiedades a proyectos y gestión de fechas

Descripción y detalles:

Como administrador de proyecto quiero poder eliminar o editar proyectos que haya creado anteriormente. Así como escribirles una breve descripción e incluirles una fecha límite. También poder eliminar tareas de ese proyecto.

- La fecha límite y la descripción será algo opcional, por tanto no será necesario incluirlo en la creación del mismo (luego podremos añadirlo).

- Desde la página de **Proyectos** dar al botón *Eliminar* para eliminar el proyecto.

- Desde la página de **Proyectos** dar al botón *Editar* para editar el proyecto.

- Cuando creo el nuevo proyecto, además del nombre se indicará una descripción y una fecha límite.

- Cuando edite el proyecto, podré editar también la descripción y la fecha límite.

- Poner fecha en los comentarios de las tareas de los proyectos.

Condición de satisfacción (COS):

- Crear un proyecto indicando una fecha límite y escribiendo la descripción.
- Editar el proyecto cambiando el nombre. Comprobar que se ha realizado el cambio correctamente.
- Editar el proyecto cambiando la fecha límite. Comprobar que se ha realizado el cambio correctamente.
- Editar el proyecto cambiando la descripción. Comprobar que se ha realizado el cambio correctamente.
- Añadir alguna tarea al proyecto para poder eliminarla y comprobar que no se muestra.
- Eliminar el proyecto y comprobar que no se muestra.

Aspecto interfaz de usuario:

Funcionalidades implementadas

Se expondrá una breve descripción para el usuario y una breve descripción técnica de cada funcionalidad implementada.

Añadir prioridad de tareas

Esta es una nueva funcionalidad que nos permite asignar una prioridad a las tareas que vayamos creando, de esta forma podremos darles más o menos importancia a la hora de tenerlas en cuenta y poder cumplir con las mismas. Estas prioridades irán del 1 al 3 por orden de importancia. Cada vez que creamos una nueva tarea, nos pedirá que asignemos una prioridad con un selector de opciones que al clicar nos aparecerán listados los números 1, 2 y 3. Una vez creada la tarea nos aparecerá en el listado de tareas con esa prioridad asignada, además cuando intentemos editar la tarea, también podremos editar esta prioridad.

Técnicamente esta nueva funcionalidad consiste en añadir a la entidad Tarea una nueva columna que represente la prioridad de la tarea, por tanto tuvimos que modificar el archivo **Tarea.java** para poder añadir dicha columna. No será necesario crear un nuevo método del controlador para esta nueva funcionalidad, simplemente tendremos que cambiar el formulario para que nos aparezca el selector de la prioridad y que éste se guarde en el objeto correspondiente para que a la hora de que se ejecute el método POST a la dirección */usuarios/{id}/tareas/nueva* para crear una nueva tarea, también se guarde y almacene el valor de la prioridad asignada en la base de datos. Lo mismo tendremos que hacer en el método POST que usamos para modificar las tareas.

Añadir un administrador de equipo

Esta nueva funcionalidad consiste en que aquel usuario que cree un nuevo equipo, pasará a ser automáticamente el administrador del mismo. Como administrador, el usuario podrá editar y eliminar el equipo que ha creado siempre que quiera, además de poder eliminar a cualquier otro usuario que esté dentro del equipo que administra. El administrador del equipo, no tiene porqué estar dentro de aquel equipo que administra, es decir, lo podrá administrar desde fuera y pertenecer a otro equipo; obviamente puede pertenecer al equipo que administra.

Técnicamente hablando, esta nueva funcionalidad consiste en que crearemos otra asociación entre la entidad Equipo y Usuario en la que definiremos este rol de administrador de equipo, esto se traduce en una nueva columna en la entidad de Equipo en la que se define el id del administrador del equipo. De esta forma, cada vez que creamos un equipo, el id del usuario logeado lo meteremos en esta nueva columna para que pase a ser el

administrador del mismo y el único con la posibilidad de editar y borrar dicho equipo. Además tendremos que modificar el método POST del controlador a la dirección */equipos/nuevo* para que cuando creemos un nuevo equipo, le pasemos también el id del usuario administrador del mismo, de esta forma al crear el nuevo equipo tendremos bien identificado al administrador.

Creación de proyectos y tareas de proyecto

Esta nueva funcionalidad consiste en la incorporación a nuestra aplicación de proyectos asociados a equipos. Cada usuario tendrá una pestaña en la barra de navegación que ponga “Proyectos” y si accedemos a esta pestaña nos dirigiremos al listado de proyectos que tiene dicho usuario, además podremos acceder mediante un enlace al detalle de cada uno de los proyectos, donde podremos ver las distintas tareas del proyecto (que funcionan de la misma manera que las tareas de usuario que ya teníamos). Cualquier usuario que pertenezca a un proyecto podrá crear, editar y borrar las tareas del mismo. Para poder crear un nuevo proyecto tendremos que acceder al detalle del equipo (donde podemos ver el listado de usuarios que pertenecen a dicho equipo), solo si somos el administrador de dicho equipo tendremos la posibilidad de crear o borrar nuevos proyectos, en caso de serlo, al acceder el detalle del equipo, veremos que tenemos el listado de proyectos del equipo y un botón “Nuevo proyecto” para acceder a un formulario donde pondremos el nombre del proyecto. Una vez hecho esto clicamos en “Crear” y se creará este nuevo proyecto. Todos los usuarios que pertenezcan al equipo donde se ha creado el proyecto, podrán ver tanto en el detalle del equipo como en la pestaña “Proyectos” de la barra de navegación este nuevo proyecto creado y todos los del equipo. Al igual que para crear y borrar, solamente el administrador del equipo podrá modificar el nombre del proyecto. Si un usuario decide abandonar el equipo, automáticamente abandonará el/los proyecto/s asociados a dicho equipo y ya no los podrá visualizar ni acceder a ellos.

Técnicamente hablando, para incorporar esta nueva funcionalidad hemos tenido que crear una nueva funcionalidad Proyecto y asociarla por un lado con Equipo para poder determinar con qué equipo se asocia el proyecto y por otro lado con Tarea ya que cada proyecto tiene sus propias tareas a realizar, comunes a todos los usuarios que pueden acceder a él. Además de esto, obviamente hemos tenido que definir métodos en la capa de servicio para poder realizar todas aquellas funcionalidades que necesitamos (crear un proyecto, borrarlo, modificarlo, listado de tareas de un proyecto, usuarios que pertenecen a un proyecto…), todo esto apoyándonos en la capa de datos gracias al **ProyectoRepository.java** con el que podremos manipular la base de datos. Finalmente tendremos que crear también los métodos GET, POST, DELETE... correspondientes del controlador para las distintas direcciones que tendremos que atacar para, al igual que en la capa de servicio, crear un proyecto, borrarlo, modificarlo…

A todo esto cabe añadir también algunos métodos que hemos tenido que añadir necesarios para poder hacer más manejable la aplicación, como pueden ser obtener los proyectos de un equipo, obtener a partir de un usuario todos los proyectos a los que pertenece en función de los equipos a los que pertenece… además obviamente de todas las vistas necesarias para visualizar el listado de proyectos, formulario de creación/edición de proyecto, vista de las tareas del proyecto...

Añadir estado a las tareas y búsquedas de las mismas

Esta nueva funcionalidad consiste en que desde el listado de tareas de usuario podremos ver, y modificar, tres posibles estados: “PENDIENTE”, “ACTIVA” y “TERMINADA”. Cuando una tarea se crea aparecerá automáticamente como “PENDIENTE”.

Además, se añaden cuatro filtros a la lista de tarea: una barra de búsqueda, que busca por palabras; un filtro para mostrar solamente las tareas de un determinado estado, de los tres citados anteriormente; un filtro para excluir estado, que muestra todas las tareas que no se encuentren en ese estado; y, por último, un filtro que ordena por estado, mostrando primero las tareas del estado indicado y después el resto de tareas por id.

Para ello se crea en el modelo **Tarea.java** el enum *EstadoTarea* cuyos valores son: “PENDIENTE”, “ACTIVA” y “TERMINADA” y se declara la variable de ese tipo en la base de datos como enum de tipo ordinal @Enumerated(EnumType.ORDINAL), lo que posibilita la introducción del valor en datos-dev.sql y datos-test.sql. En **TareaService.java** se crean las funciones que permiten cambiar el estado de la tarea, además de funciones que devuelven listas filtradas con los parámetros explicados en el párrafo anterior.

Para mostrar las tareas filtradas, en **TareaController.java** hemos añadido los métodos GET correspondientes que nos permiten atacar a las direcciones siguientes:
/usuarios/{id}/tareas/filtrar
/usuarios/{id}/tareas/excluir
/usuarios/{id}/tareas/ordenar
/usuarios/{id}/tareas/buscar
Y para actualizar el estado desde la vista de lista de tareas el método POST:
/tareas/{id}/estado

Posibilidad de añadir comentarios a las tareas de proyectos

Esta nueva funcionalidad consiste en que desde el listado de tareas de un proyecto específico, podamos acceder al detalle de las mismas y poder visualizar comentarios que nosotros u otros usuarios hayan podido dejar en la tarea para informar de la evolución de la tarea, informar de cosas a tener en cuenta en la tarea o cualquier otro tipo de comentario. Los usuarios que pertenezcan a un proyecto podrán acceder a este y visualizar todas las tareas del mismo, al mismo tiempo podrán acceder a cada tarea y ver los distintos comentarios y a parte crear los suyos propios para informar de algo también. Estos comentarios estarán ordenados por fecha.

Técnicamente hablando, para esta nueva funcionalidad que hemos creado, en primer lugar hemos tenido que crear una nueva entidad Comentario que hemos relacionado por un lado con Usuario para poder saber quien hacía el comentario y por otro lado con Tarea para que esté relacionado con la tarea de proyecto en la que se ha creado el comentario. Al igual que para el resto de entidades hemos creado el correspondiente **Repository** para poder manejar los comentarios en la base de datos. Este repository lo

usaremos en la capa de servicios donde hemos creado los distintos métodos para crear comentario, borrarlo, listar los comentarios de una determinada tarea…
Además hemos creado los métodos GET, POST correspondientes para poder atacar las distintas direcciones de creación, listado de comentarios…
Finalmente creamos la vista de detalle de una tarea de proyecto donde podremos ver los distintos comentarios de dicha tarea y el usuario que lo ha hecho, junto con un botón de eliminar en el caso de que el comentario lo haya hecho el usuario logeado. Al final tendremos un *TextBox* y un botón de crear comentario donde escribiremos el comentario y al clicar en el botón de "Añadir comentario" se ejecutará el método POST en la dirección */usuarios/{id}/proyectos/{proyecto}/tareas/{tarea}* y se creará el nuevo comentario en la base de datos y volveremos a redirigirnos a la misma página, veremos que el nuevo comentario ya estará en el listado junto con el usuario que lo ha creado, nosotros, y el botón de borrar comentario.

Posibilidad de editar los datos del usuario y pulir interfaz de usuario

Para esta nueva funcionalidad lo que hemos pretendido hacer es que el propio usuario pueda cambiar sus datos del perfil bien porque se ha equivocado al escribir o al poner alguno de ellos o bien porque prefiere cambiarlos por cualquier otra razón. Desde el propio nombre de usuario que aparece en la barra de navegación a la derecha del todo, podremos clicar y acceder al perfil del usuario, donde veremos todos los datos correspondientes al mismo. Una vez dentro podremos ver un botón "Editar perfil" que al clicar sobre él nos dirigirá a un formulario donde podremos editar todos los datos que hemos puesto como editables (nombre de usuario, fecha de nacimiento y email). Al aceptar estos nuevos datos, se cambiarán automáticamente y podremos verlo reflejado.
En esta funcionalidad también hemos aprovechado para rectificar y pulir algunas cosas que nos faltaban como por ejemplo añadir a la lista de tareas del proyecto el tema de los estados y todos los filtros y búsquedas que hicimos en una historia anterior.

Técnicamente hablando, lo principal que hemos tenido que hacer es implementar los métodos correspondientes en la capa de servicio del usuario para modificar aquellos datos que se han cambiado en el formulario de edición, y del mismo modo, implementar los métodos GET y POST correspondientes del controlador que ataquen a las direcciones correspondientes para poder acceder a los formularios de edición y posteriormente el método POST cambie los datos en la base de datos. Otra cosa que podemos destacar que hemos podido meter es, al igual que en el listado de tareas de usuario, en los métodos de controlador de los proyectos (**ProyectoController.java**) hemos añadido los métodos GET correspondientes que nos permiten atacar a las direcciones siguientes:
/usuarios/{id}/proyectos/{proyecto}/tareas/filtrar
/usuarios/{id}/proyectos/{proyecto}/tareas/excluir
/usuarios/{id}/proyectos/{proyecto}/tareas/ordenar
/usuarios/{id}/proyectos/{proyecto}/tareas/buscar
para poder filtrar las distintas tareas de proyectos con distintos criterios de estados de tarea, al igual que ya hicimos con las de usuario.

Añadir clasificación de tareas por categorías

Para esta nueva funcionalidad pretendemos proporcionar al usuario una forma de clasificar u ordenar sus tareas. Desde un apartado específico dentro de la página web, se podrá crear y eliminar tantas categorías como un usuario quiera. Dichas categorías se vinculan con una tarea o varias. Una tarea podrá tener más de una categoría o ninguna, si a la hora de crearla o modificarla así se escoge.
Por último, se podrán visualizar las categorías vinculadas con las tareas; o bien en la lista principal de tareas, o todas las tareas que incluyan una categoría en concreto.

Técnicamente hablando, lo primero que hemos hecho ha sido crear una nueva entidad llamada categoría. Además, añadir las relaciones necesarias para que la base de datos funcione de la forma que queremos. Hemos añadido métodos en la capa de servicio. Tanto para la nueva entidad categoría como para tarea o usuarios. Por otro lado se han implementado los métodos en el controller necesarios para el correcto funcionamiento en el uso de categorías (métodos GET, POST y DELETE).
Por último hemos creado los html necesarios y actualizados los que ya existían como la lista de tareas para que reflejen dicha nueva información. Reutilizando código con el fin de no repetirlo. Podemos destacar el uso de desplegables dinámicos y condicionales dentro del propio html para mostrar o no la información según corresponda.

Añadir propiedades a los proyectos y gestión de fechas de finalización

En esta nueva funcionalidad hemos querido implementar algunos añadidos a los proyectos que ya implementamos en una historia anterior. En esta ocasión vamos a añadir a los proyectos una pequeña descripción donde podremos indicar la finalidad de dicho proyecto o indicar cualquier cosa que creamos necesaria que los usuarios que pertenezcan a dicho proyecto tengan que saber, además, por otro lado, vamos a añadir una fecha límite de finalización que usaremos para indicar cuando tiene que finalizar dicho proyecto o para cuando tienen que estar finalizadas las tareas. A partir de ahora, cuando queramos crear un nuevo proyecto nos aparecerán estos dos campos que tendremos que completar de forma opcional, no son obligatorios, y a la hora de crearlo se guardarán correctamente dichos datos. Estos nuevos campos, como usuarios los podremos visualizar en la interfaz. Por un lado en el listado de todos los proyectos podremos ver la fecha límite de finalización del mismo y, por otro lado, cuando accedemos al detalle del proyecto podremos ver la descripción del mismo donde veremos su finalidad o aquello que el creador del proyecto haya visto necesario colocar para el resto de usuarios.

Técnicamente hablando, esta nueva funcionalidad consiste principalmente en incluir en nuestra entidad Proyecto dos nuevos atributos o columnas, una con la descripción del proyecto y otra con la fecha límite. De este modo, cuando se cree la base de datos, se

crearán también estas dos nuevas columnas donde podremos almacenar esta nueva información. Además en la capa de servicio de los proyectos hemos tenido que modificar algunos métodos para tener en cuenta estos nuevos datos que tendremos que añadir, ya que tendremos que asignarlos al objeto proyecto para que el **ProyectoRepository** lo pueda almacenar correctamente, del mismo modo hemos tenido que actualizar los métodos POST del controlador de creación y edición del proyecto para tener en cuenta esto mismo también. Finalmente, como es obvio, para poder meter estos nuevos datos hemos tenido que modificar los formularios para meter los TextBox necesarios para poder meter las propiedades y la fecha.

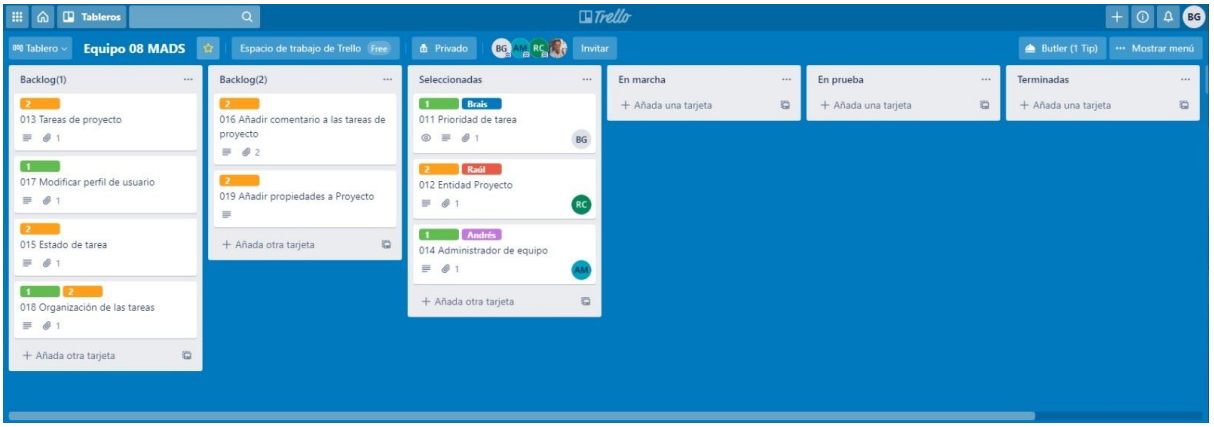
Informe sobre la evolución del desarrollo

Durante el sprint se realizaron diversas instantáneas de los tableros de Trello y de GitHub para reflejar el avance en la implementación de las historias de usuario.

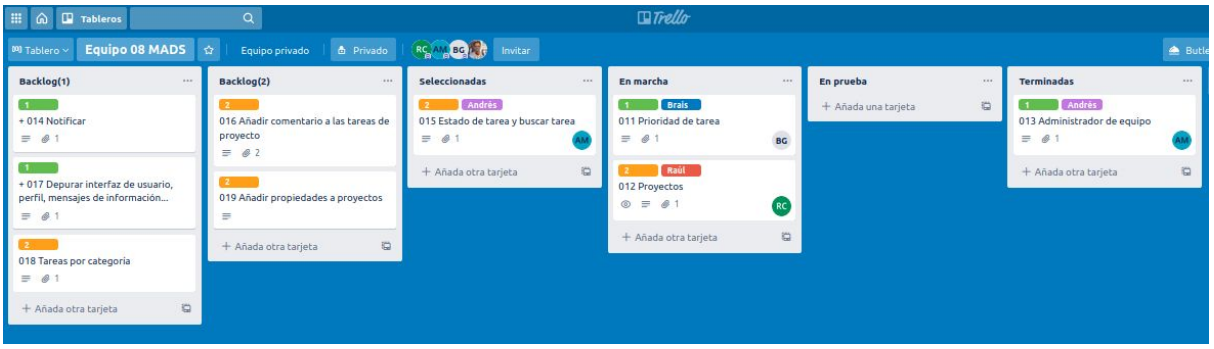
Trello

En estas instantáneas se puede ver de un vistazo el nombre de la historia y el peso que se le ha asignado (1 o 2), en función del tiempo de implementación estimado, así como el integrante del grupo que se va a encargar de su desarrollo.
También permite apreciar de un simple vistazo como de avanzado se encuentra el sprint, basándonos en la cantidad de historias que se encuentran en las columnas de la derecha, en comparación con las de las columnas de backlog o seleccionadas.

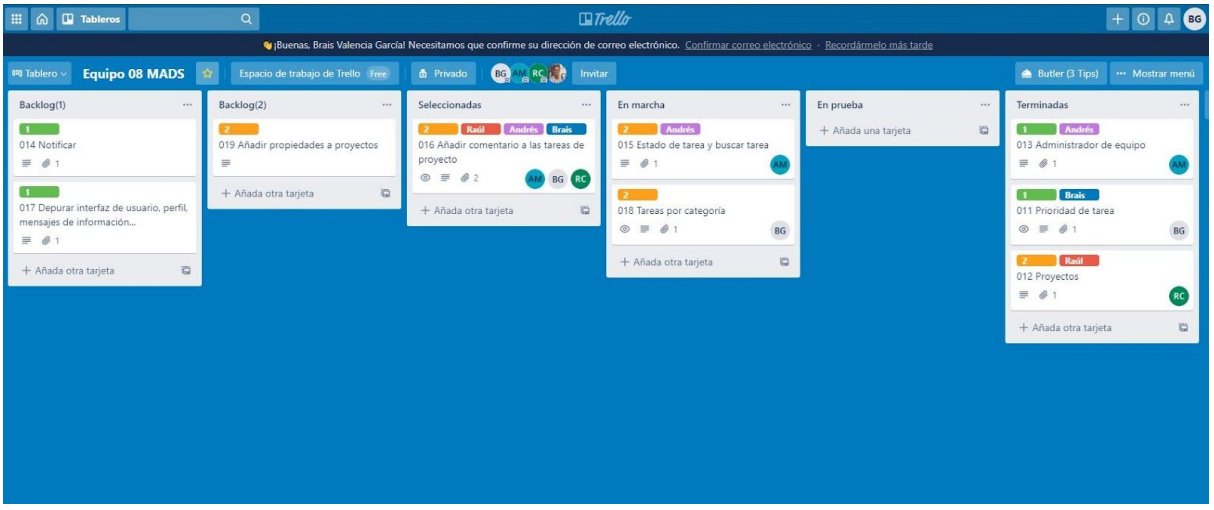
Captura del 9-12-2020: Con el sprint apenas empezado podemos apreciar como cada integrante del grupo ha escogido su primera historia de usuario a desarrollar.



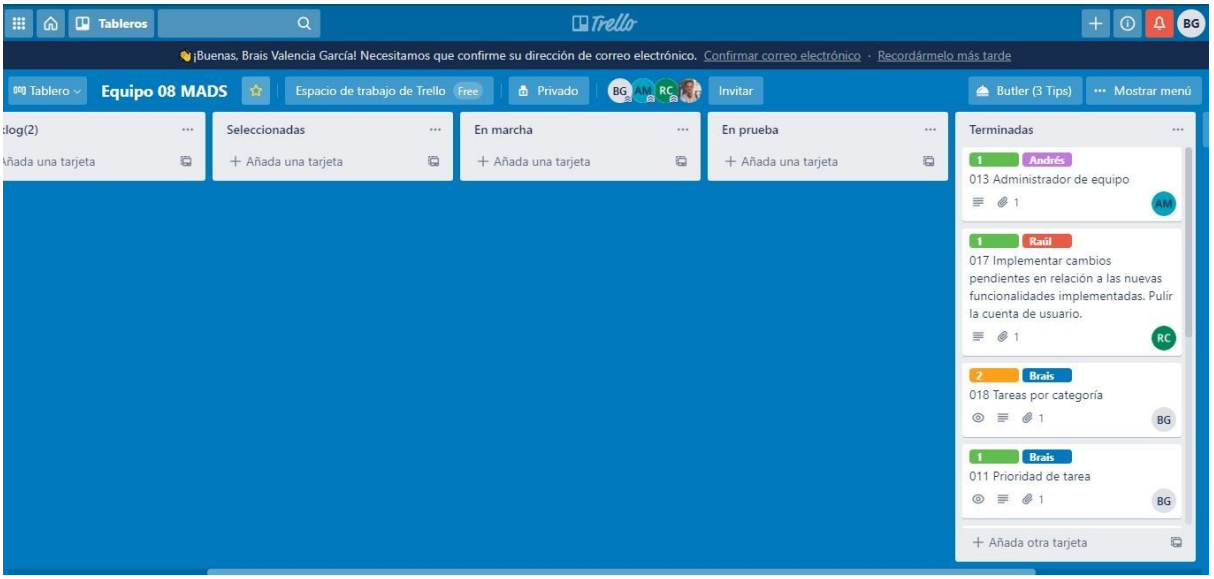
Captura del 15-12-2020: Con el sprint ya en marcha, se están desarrollando las historias básicas, de las que dependen algunas otras del backlog



Captura del 18-12-2020: En pleno sprint, la mayoría del trabajo ya se ha hecho o se encuentra en marcha.



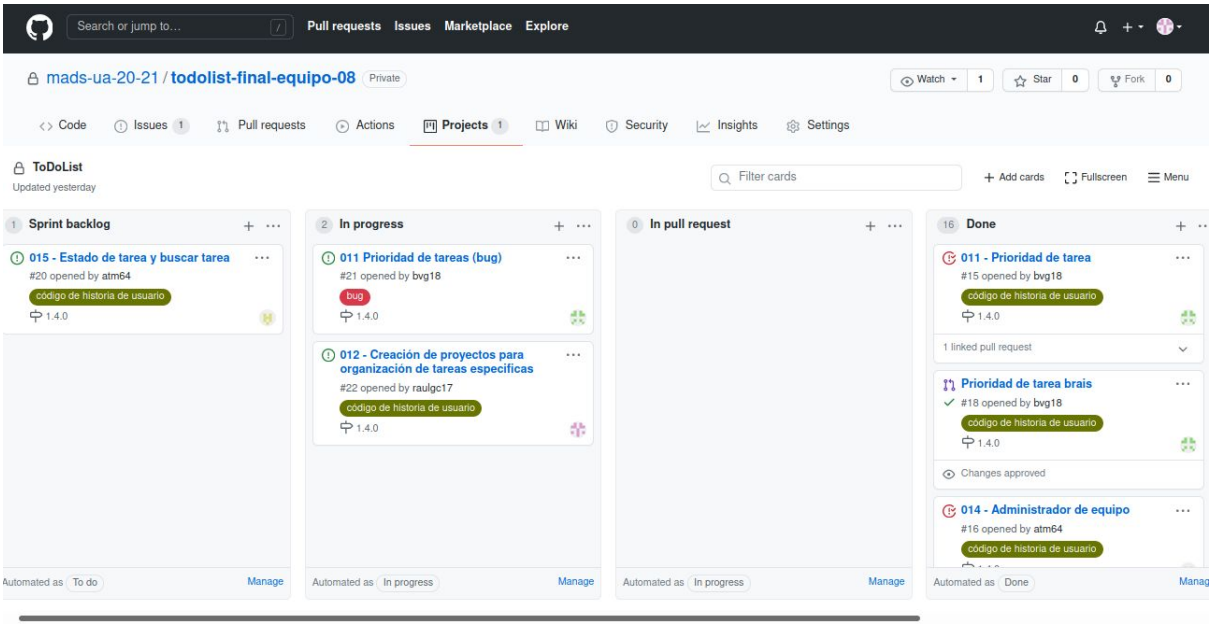
Captura del 22-12-2020: Con todas las historias implementadas y probadas, lo que les confiere el estado de “Terminadas”, podemos dar por concluido el sprint.



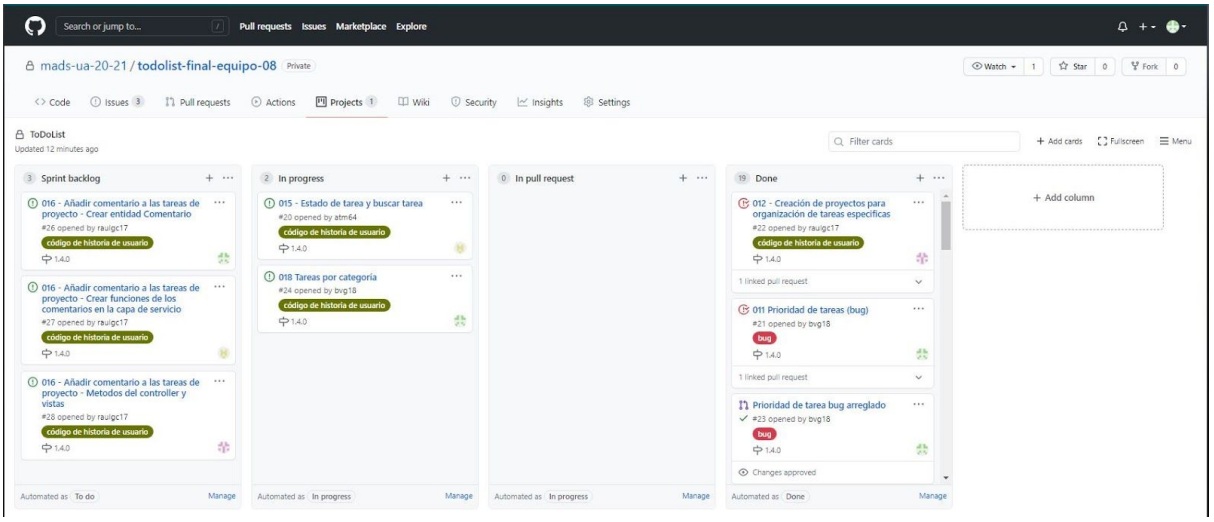
GitHub

En las siguientes instantáneas se reflejan, en la pestaña **Projects,** de GitHub, los avances del sprint. En lugar de historias de usuario, aquí se muestran issues que, tras implementarse en ramas propias y pasar por un proceso de aprobado en el pull request, se unirán a la rama principal.

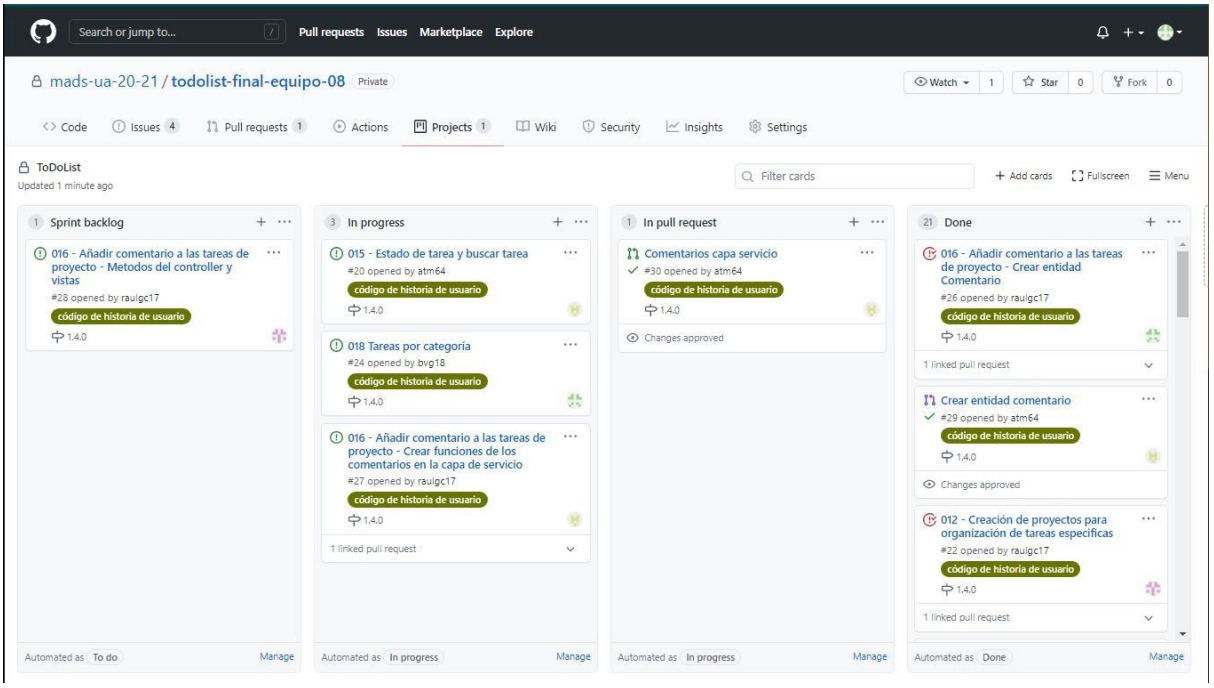
Captura tomada el 15-12-2020: En la siguiente captura se puede apreciar que, aunque la mayoría de issues están etiquetados como “código de historia de usuario”, en **In Progress** hay un issue de correción de bugs.



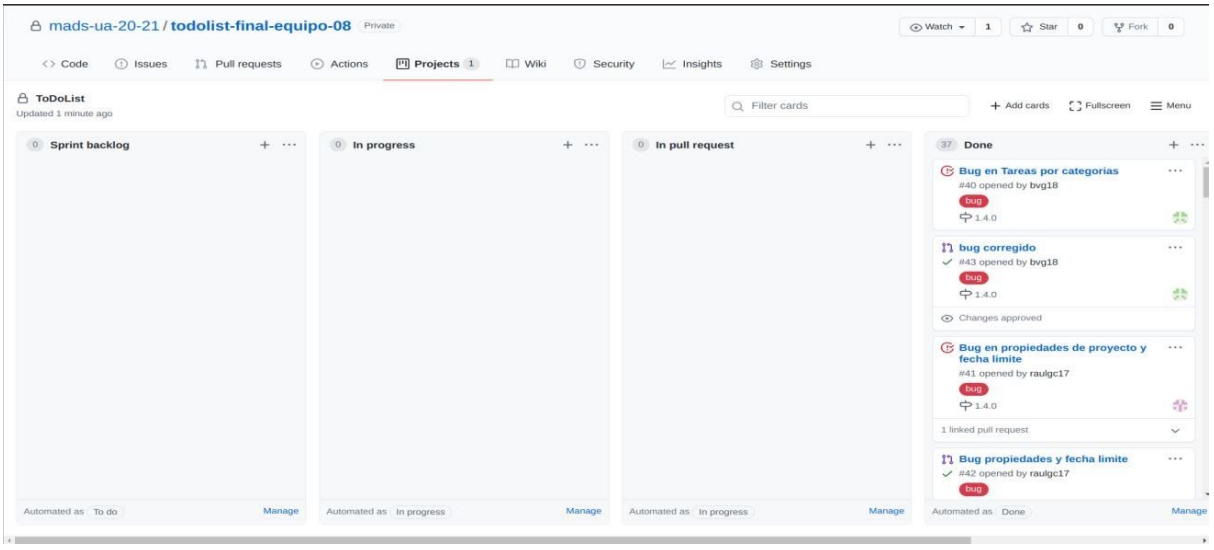
Captura tomada el 18-12-2020: Corresponde a la primera mitad del sprint cuando el equipo ya estaba adaptándose al sprint y la forma de trabajar en él.



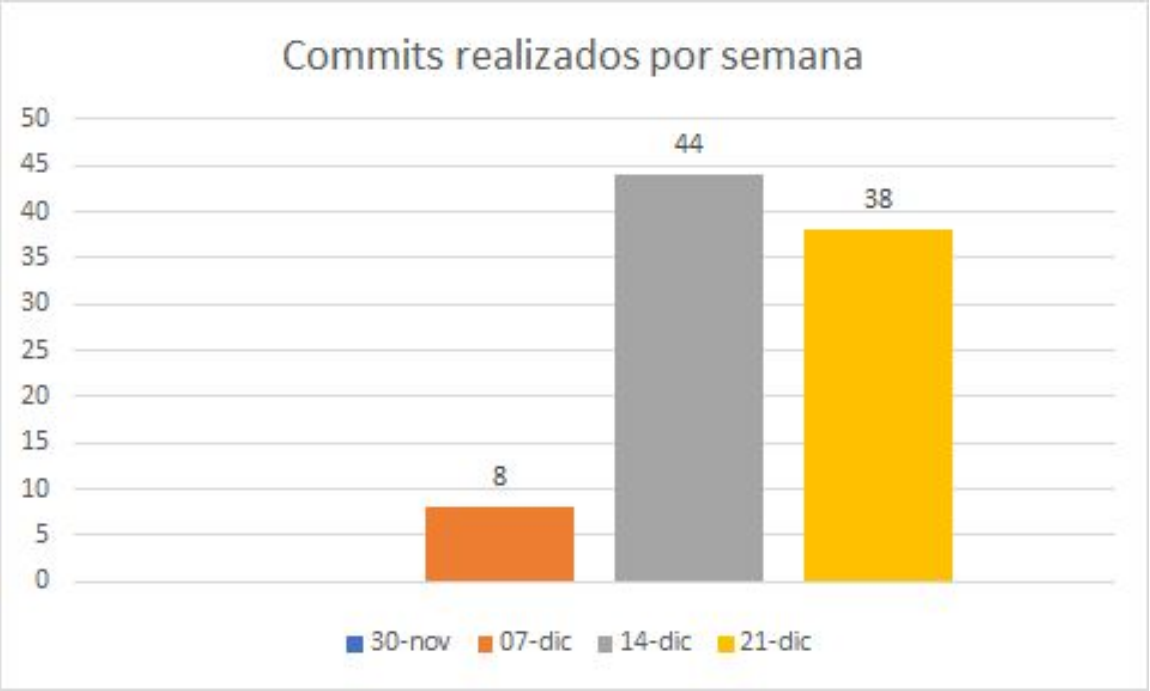
Captura tomada el 19-12-2020: En este momento el equipo entero estaba inmerso en el sprint, trabajando ya de manera óptima para acabar las issues que quedaban.



Captura tomada el 22-12-2020: Con todos los issues de historias de usuario y de bugs implementados, se puede dar por finalizado satisfactoriamente el sprint.



Desarrollo



De esta captura podemos observar que el número total de commits empleados ha sido 90. Destinando la primera semana de trabajo (30-nov) para organizar y repartir el trabajo, (trello y google docs). Y las semanas posteriores para repartir el trabajo de una forma más o menos equitativa.
Cabe mencionar que el número de commits no refleja en ningún caso la cantidad de horas trabajadas. Como se demuestra en la primera semana de trabajo.

Informe sobre la sesión de pair programming

PAIR PROGRAMMING SESSION
Raúl García Carrión
Andres Tebar Moreno
Brais Valencia García

La sesión de pair programming tiene como objetivo el desarrollo de código de forma grupal. Siendo un integrante del grupo quien escribe el código, mientras el resto del grupo observan, comentan o indican lo que hay que hacer. Una vez terminado el turno del primer integrante, pasará a ser el “anfitrión” el siguiente integrante.

Se ha elegido la historia de usuario 016 - “Añadir comentario a las tareas de proyecto”.

Se ha dividido en dos issues de GitHub:

- Uno para la implementación del modelo **Comentario**.
    - Se ha creado la rama:“crear-entidad-comentario”, que se usará para desarrollar esta issue.
    - Hora 10:52 - Brais crea la clase java, sus atributos y algunas funciones.
    - Hora 11:22 - Raúl continúa con el modelo y las relaciones en la base de datos.
    - Hora 11:51 - Andrés termina la implementación y realiza el pull request.

- Otro issue para la capa de servicio
    - Rama:“comentarios-capa-servicio”.
    - Hora 12:08 - Andrés crea la rama y clase java.
    - Hora 12:26 - Brais continúa con la implementación del servicio.
    - Hora 12:45 - Raúl añade más funciones a la clase.
    - Hora 13:00 - Andrés termina la implementación de la clase y realiza el pull request.

Fin de la sesión a la 13:16 por Andrés.
Aunque el sprint no ha cubierto todo el desarrollo de la historia, al tratarse de una historia bastante extensa, sí que se han asentado las bases para terminarlo con facilidad más tarde.

Para la sesión han participado los tres integrantes del grupo, en constante comunicación y feedback, empleando las tecnologías de llamada grupal y pantalla compartida de forma telemática, concretamente la aplicación Discord (https://discord.com).

Resultado de la retrospectiva

Conclusiones

Una vez realizado el desarrollo de este sprint con las nuevas funcionalidades y haber trabajado en equipo nos hemos dado cuenta de varios aspectos que queremos destacar.

Aunque en un primer momento los procedimientos a seguir para el desarrollo del sprint podían parecer un poco tediosos, una vez nos hemos acostumbrado a la metodología hemos podido comprobar que trabajando así en equipo podemos adelantar mucho más rápido, ya que por un lado podemos avanzar de forma paralela en el proyecto y por otro lado, si nos atascamos en algún punto del desarrollo, es más fácil solucionarlo y seguir trabajando, puesto que entre todos podemos llegar a una solución mucho más rápido.

Por otro lado, también nos hemos encontrado algún problema, como por ejemplo que aparezcan conflictos en algún archivo a la hora de integrar lo desarrollado con la rama principal, ya que a lo mejor dos personas habían escrito sobre la misma zona del archivo. En este caso hemos tenido que revisar las dos versiones de los archivos y dejar lo necesario para que la aplicación siga funcionando.

Durante el sprint, hemos visto que es muy importante la comunicación constante entre los miembros del equipo, para saber qué está haciendo cada uno y así no tocar archivos o partes del archivo que puedan producir conflictos. Además también nos ha servido para ir sabiendo como iba el desarrollo de las historias que cada uno de nosotros estaba haciendo y ver cómo de avanzados íbamos respecto al tiempo que teníamos. También hemos aprendido durante el sprint a asignar mejor el peso asignado a cada historia de usuario y a comprender el código de nuestros compañeros, a la hora de aprobar un pull request en GitHub.

A parte de esta constante comunicación, también nos ha sido muy útil el ir actualizando los tableros de Trello y GitHub al mismo tiempo que íbamos acabando los desarrollos ya que también nos permitía saber el avance de la aplicación como iba o si habían ciertas partes que teníamos que probar para comprobar que funcionaban bien. La descripción de lo implementado y el uso de un historial commits también ha facilitado la tarea a la hora de localizar algún fallo.

Siguiente Sprint

Con la experiencia adquirida en este sprint podríamos organizar mejor el orden de las historias de usuario a implementar, para evitar colisiones innecesarias en el código, así como asignarles un peso más acorde a su extensión. También podríamos trabajar con más fluidez usando las herramientas proporcionadas por Trello y GitHub. Otra cosa que podríamos mejorar para futuros desarrollos es documentar el código adecuadamente para que el resto de integrantes del equipo no tenga problema de saber que hacen algunas partes del código que no hayan hecho ellos y así evitar pequeñas pérdidas de tiempo preguntando qué hace el código o averiguando cómo funciona.