PRÁCTICA 5

MADS

Benziane Mohammed Adel
Alejandro Company Rincón
Luis Alfonso Culiañez

Índice

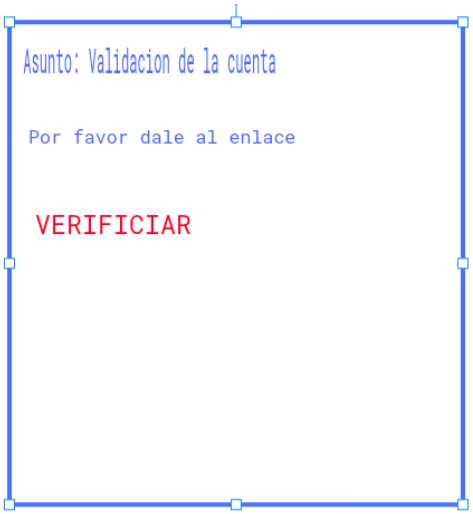
Sprint Backlog

1. Validación usuario mediante correo de confirmación

- **Descripción**
  Enviar correo de verificación al correo del usuario en el que o bien mediante un link que pinche o un código de activación verifique el correo

- **Detalles**
  - Enviar un correo al usuario que intente registrarse.
  - Esperar a que se verifique antes de activar su cuenta.
  - No activaremos la cuenta hasta que el usuario verifique la cuenta pulsando en el enlace.
  - Una vez pulsado automáticamente le deja acceder.
  - El usuario no podrá darle dos veces al enlace de la manera que cuando se da click la primera vez se guarda en la bd y se le devuelve a dar le aparecerá error de que ya está verificada la cuenta.

- **Borrador del aspecto de la interfaz de usuario resultante**
  Tiene que llegar un correo de la manera



- **Condiciones de satisfacción**
  Cuando un usuario le da al botón de registrar le llevará a una página avisando que tiene que verificar la cuenta y una vez verificada le deja entrar en su cuenta.

2. Filtro usuarios registrados (admin, por username)

- **Descripción**
  En el listado de usuarios, visible solo por el administrador, pondremos un buscador por el nombre del usuario para así facilitar el trabajo de este.

- **Detalles**
  - Añadir un buscador por nombres.

- Buscar en la base de datos usuarios que contengan ese nombre.
- En primer lugar añadiremos un input en el listado de usuarios así el administrador podrá buscar el usuario deseado sin mirar todos los usuarios.
- Va a ser un input contain en vez de equal asi haria mejor la búsqueda.
- Devolverá un listado pequeño de uno o varios usuarios.
- En total habrá dos listados en la página uno de buscador y otro que es listado por defecto.

- **Borrador del aspecto de la interfaz de usuario resultante**

- **Condiciones de satisfacción**
  Estamos en la página de usuarios registrados introducimos un nombre en el buscador le damos a buscar y tendrá que aparecer el usuario o los usuarios que contengan esa cadena introducida en el buscador.

3. Filtro tareas (nombre)

- **Descripción**
  Podremos hacer una búsqueda entre las diferentes tareas por el nombre de estas
- **Detalles**
  - Añadir un buscador por nombres.
  - Buscar en la base de datos tareas que contengan ese nombre.
  - En primer lugar añadiremos un input en el listado de tareas así el usuario podrá buscar la tarea deseada sin mirar todas las tareas.
  - Va a ser un input contain en vez de equal asi haria mejor la búsqueda.
  - Devolverá un listado pequeño de una o varias tareas.
  - En total habrá dos listados en la página uno de buscador y otro que es listado por defecto.
- **Borrador del aspecto de la interfaz de usuario resultante**

- **Condiciones de satisfacción**
  Cuando el usuario esté en la página de listado de tareas introduce un nombre de tarea deberá aparecer la tarea cuyo nombre contenga los caracteres introducidos en el buscador.

4. Filtro búsqueda equipos (por nombre)

- **Descripción**
  Podremos hacer una búsqueda entre los diferentes equipos por el nombre de estos

- **Detalles**
  - Añadir un buscador por nombres.
  - Buscar en la base de datos equipos que contengan ese nombre.
  - En primer lugar añadiremos un input en el listado de equipos así el usuario podrá buscar el equipo deseado sin mirar todos los equipos.
  - Va a ser un input contain en vez de equal asi haria mejor la búsqueda.
  - Devolverá un listado pequeño de un o varios equipos.
  - En total habrá dos listados en la página uno de buscador y otro que es listado por defecto.

- **Borrador del aspecto de la interfaz de usuario resultante**

- **Condiciones de satisfacción**
  Estamos en la página de equipos introducimos un nombre en el buscador le damos a buscar y tendrá que aparecer el equipo o los equipos que contengan esa cadena introducida en el buscador.

5. Olvidar Contraseña

- **Descripción**
  En la página de login añadiremos un enlace para que en caso de olvidar la contraseña poder recuperar la cuenta.

- **Detalles**
  - Añadir un enlace en la página de login
  - Enviar correo de recuperación.
  - Recuperar la cuenta introduciendo una nueva contraseña.
  - Enviar al cliente un formulario mediante un enlace por correo electrónico con el cual podrá solicitar una nueva contraseña.

- **Borrador del aspecto de la interfaz de usuario resultante**

- **Condiciones de satisfacción**
  Cuando el usuario se olvide de la contraseña podrá recuperarla de manera que dándole al link le enviaremos correo y ya se soluciona introduciendo una nueva.

6. Gestión miembros del equipo (banear para siempre, expulsar)

- **Descripción**
  En esta tarea el líder (creador del equipo) puede expulsar a uno de los miembros de forma indefinida, es decir no podrá volver a entrar nunca.

- **Detalles**
  - En la lista de equipos al usuario administrador aparecerá el botón de eliminar el equipo o editar su nombre.

- El administrador tendrá posibilidad de gestionar miembros del equipo.
- El líder del equipo puede expulsar a los miembros.
- El líder del equipo puede editar el equipo a su manera.
- El líder del equipo puede editar los miembros del equipo de manera que expulsarlos del equipo o modificar el nombre del equipo.
- También el miembro podrá banearlos de manera que no puedan acceder al equipo otra vez es decir cada equipo tendrá usuarios prohibidos.

- **Borrador del aspecto de la interfaz de usuario resultante**

- **Condiciones de satisfacción**
  - En la ventana de equipo (ver miembros) el usuario líder expulsara los usuarios.
  - En dicha ventana el líder podrá banear a los usuarios (expulsar de manera definitiva para que no puedan entrar).

7. Definir diferentes roles

- **Descripción**
  Introduciremos diferentes roles, a parte del líder del grupo, para definir diferentes funcionalidades.

- **Detalles**
  - Añadir nuevos roles de usuarios de equipo
  - El líder puede asignar cualquier rol de miembros de equipo.
  - Cada rol tendrá unas funcionalidades distintas.
  - El primer rol sería el líder que tendrá todas las funcionalidades como por ejemplo es el único que puede eliminar el equipo creado o modificar su nombre, crud de tareas del equipo, asignar tareas a los miembros de equipo, eliminar miembros de equipo (tarea 006),modificar roles a los miembros.
  - El segundo rol será gestor que consiste en crud de tareas y asignar tareas a los miembros.
  - Otro rol participante este rol no tiene funcionalidades solo es como lector.
  - Un detalle muy importante cuando un usuario crea un equipo se asigna por defecto que es el líder del equipo y cuando un usuario se añade al equipo desde el listado de equipos se le asigna por defecto

participante, entonces ya desde allí si el líder le quiere dar otro rol (líder,gestor,participante).

- **Borrador del aspecto de la interfaz de usuario resultante**
  Esto sería un poco complicado pero de todos modos, la idea visual es añadir una columna en el listado de miembros de los roles que tienen cada uno y otra columna(visible para el líder) para editar ese rol que sería como un desplegable (select).

- **Condiciones de satisfacción**
  Siendo líder del equipo le asignó a un miembro rol de editor y ya podrá editar el nombre de equipo o crear tareas de equipo.

8. Crud Usuario

- **Descripción**
  Para realizar esta tarea vamos a implementar varios puntos
  Una template en la que podamos visualizar todos los datos del usuario que se encuentra logueado.
  Dentro de esta template podremos modificar nuestros datos, así como borrar para siempre nuestro usuario
  Implementaremos todas las posibles funcionalidades del usuario, que son: Crear, modificar, borrar y consultar.

- **Detalles**
  Usuario podrá cambiar su datos en la cuenta.
  En la página cuenta podrá editarlo.

- **Condiciones de satisfacción**
  Cuando un usuario este logueado podrá acceder a la página cuenta para editarlos y después se guarde correctamente en la bases de datos y podrá visualizarlos.
  También podrá borrar su cuenta para siempre, cuando lo haga se borrarán también los equipos y las tareas creadas por este usuario. Además mostraremos un mensaje para confirmar que se ha eliminado el usuario.

9. Definir las columnas "en proceso" y "por hacer" y poder mover las tareas entre ellas

- **Descripción**
  Podremos cambiar el estado de una tarea para que indique en el punto en el que se encuentra. Tendremos varias columnas que servirán también para indicar el estado de una forma más visual.
  En cada una de las columnas se encontrarán las diferentes tareas en función del estado en el que se encuentren:
  - Por hacer
  - En proceso
  - Finalizada

- **Detalles**
  - Añadir columnas de estados en el listado de tareas de los equipos.
  - Añadir botón para modificar el estado de la tarea.
  - Añadir formulario para modificar el estado de la tarea.
  - Añadir fila que muestre el estado de la tarea.

- **Condiciones de satisfacción**
  En el listado de tarea seleccione una tarea y puedo realizar las siguientes tareas:
  - Modificar su estado
  - Modificar la tarea
  - Borrar esa tarea

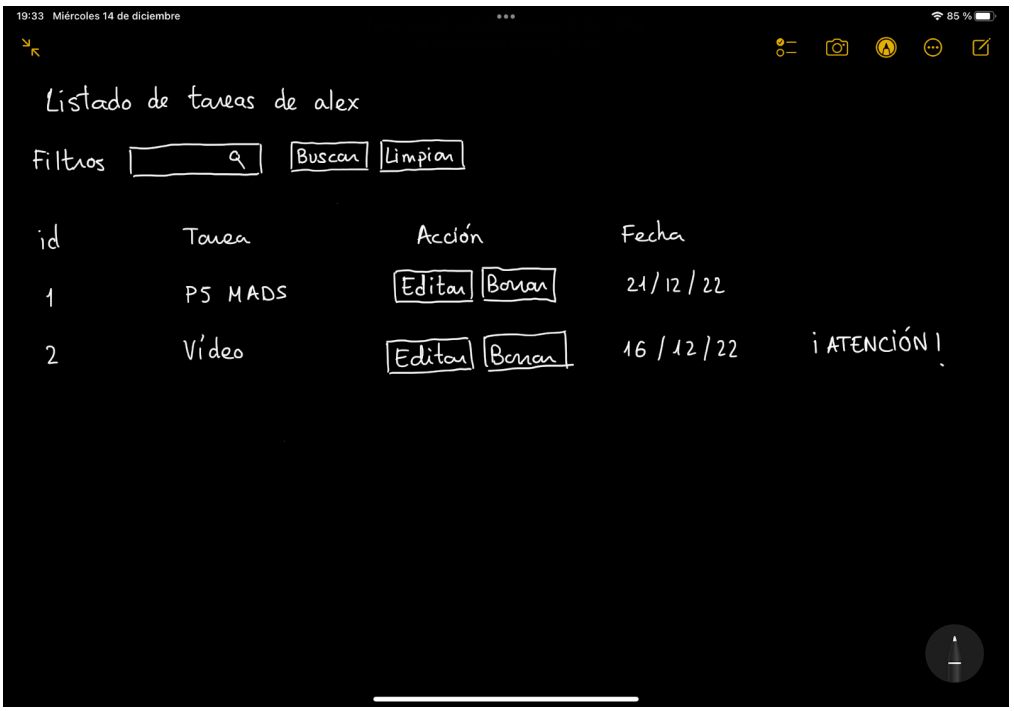
  Podré en todo momento si hay una tarea cambiarle el estado entre cualquiera de los 3 de los que disponemos.

10. Fechas en tareas

- **Descripción**
  En esta tarea podremos asignar una fecha de realización límite para las tareas. Añadiremos al modelo un campo más para asignarlo y modificaremos plantillas para la incorporación del mismo

- **Detalles**
  - Las tareas tendrán una fecha límite para realizarse.
  - En el panel principal de tareas se podrá ver la fecha límite de cada una.

- **Borrador del aspecto de la interfaz de usuario resultante**



- **Condiciones de satisfacción**

    - En el listado de tareas se podrá ver qué tareas requieren mayor prioridad.
    - Podremos modificar la fecha de entrega de las tareas.

11. Componente header

- **Descripción**
  Tenemos mucho código repetido en las plantillas de la aplicación, por lo que es necesario componentizar el header y evitar duplicidad.

- **Detalles**
  Incorporar el header en los fragmentos para unificarlo.
  Importarlo en el resto de plantillas.
  NO HAY DISEÑO PORQUE SIGUE SIENDO EL MISMO HEADER.

- **Condiciones de satisfacción**
  Mejoramos considerablemente la calidad de código y futuro mantenimiento en el header ya que nos evitamos corregir lo mismo en muchos archivos.

12. CRUD tareas por equipo

- **Descripción**
  Añadiremos todas las funcionalidades CRUD que se van a poder realizar para las tareas, pero esta vez las tareas están vinculadas con un equipo en específico.

- **Detalles**
    - Podremos:
        - Visualizar todas las tareas que van vinculadas a cada equipo
        - Crear nuevas tareas para un equipo en específico
        - Modificar una tarea en concreto
        - Borrar la tarea

- **Condiciones de satisfacción**
  Crear tanto el modelo como el servicio que se encargue de gestionar todas las tareas de los equipos
  Crear el controlador para hacer las diferentes templates
  Crear un data para controlar los datos de inserción de las tareas de equipos
  Crear las diferentes templates para visualizar y realizar las diferentes funcionalidades de esta tarea.

13. Mejorar visualización equipos que pertenece el usuario

- **Descripción**
  Modificar y agregar una nueva interfaz para saber fácilmente a los equipos que pertenece un usuario

- **Detalles**
  En la ventana de cuenta añadir una pestaña para visualizar a los equipos que pertenece el usuario logueado.
  Para ello lo que haremos será introducir en el apartado de cuenta, bajo de nuestro perfil una sección llamada Mis Equipos, en la que mostraremos únicamente los equipos a los que pertenecemos.
  Además añadiremos la funcionalidad de salir del equipo y la de editar/eliminar equipos si eres administrador.

- **Condiciones de satisfacción**
  Podremos tener una lista con todos los equipos a los que pertenece el usuario en la cuenta y podremos realizar las funciones normales de esta clase:
  1. Eliminarse como integrante del equipo
  2. Consultar las tareas y los miembros del equipo
  3. Editar o borrar el equipo en caso de ser administrador

Funcionalidades implementadas

1. Validación de la cuenta

Cada cuenta tiene que ser validada por el correo electrónico. Cada usuario al registrarse tiene que darle al enlace enviado al correo introducido en el registro.
Para el correcto funcionamiento de esta funcionalidad he añadido dos columnas en la tabla usuarios (enabled,verification code) uno para activar o desactivar la cuenta y otro para generar un código de verificación.
Por lo tanto el procedimiento técnico será el siguiente:
Un usuario al rellenar el registro y le da al botón registrar, se crea una fila del usuario con el enabled a false y verificación code con el código, si el usuario directamente va a login e intenta iniciar sesión no podrá porque el campo enabled está a false. Para poder iniciar sesion tiene que ir a su cuenta de correo y darle al enlace enviado así directamente le direcciona a login con un mensaje de verificación fue correcta, finalmente el campo enabled se pone a true y verificación code a null porque el código ya ha sido utilizado para la verificación.

2. Crud de usuario

En este apartado vamos a hacer toda la integración de la parte de usuario, es decir implementar todas las funcionalidades que puedes realizar siendo miembro de nuestra página.
En primer lugar, lo que haremos será crear un página en la que se puedan visualizar todos los datos que estén vinculados a nuestra cuenta. Es decir mostraremos una template en la que se mostrará:

- Id
- Nombre
- Email
- Fecha de nacimiento
- Administrador

(En un futuro se ampliará este apartado para visualizar también a los equipos a los que está vinculado el usuario)

Una vez tengamos ya este apartado hecho, pasaremos a implementar las otras funcionalidades que nos faltan, que son las de modificar la información de un usuario y la de borrar el usuario.
Para el apartado de modificar usuario lo que haremos será crear una nueva template en la que podamos introducir de nuevo los campos que queramos modificar. Para el apartado de borrar simplemente mostraremos un botón con una confirmación para cuando realice el borrado del usuario esté seguro y que no haya sido un error.

3. Olvidar contraseña

Si al cliente se le olvida la contraseña podrá recuperar dandole al link que está en la página de login.
Para esta funcionalidad se ha tenido que añadir dos plantillas nuevas una para indicarle al usuario donde tiene que introducir el correo. Una vez comprobado si existe el correo en el sistema enviaremos al correo otra plantilla que consiste en un formulario para introducir nueva contraseña. El usuario cuando acceda a la plantilla nueva para introducir la contraseña le pasamos por parámetro del enlace el identificador del usuario en la Bases de datos así nos facilita encontrar el usuario en el caso de poner una nueva contraseña.
Por último para desarrollo de esta funcionalidad y la de validar la cuenta se ha utilizado smtp para el envío de correo, se ha añadido la configuración en el archivo application-properties-prod. Para ejecutar la aplicación se tiene que utilizar el perfil postgres-prod ya que en el fichero comentado anteriormente fue añadida una configuración para el envío de correos.

4. Filtro tarea, filtro equipo, filtro usuarios registrados

Son tres funcionalidades distintas pero como hacen la misma función y añadido en los tres casi el mismo código.

Añadido al encabezado de la página un input de buscador con dos botones uno para limpiar y devuelve todo el listado que se encuentra en la bases de datos, y otro para buscar por el valor introducido en el input.
El buscador busca en la bases de datos por cualquier cosa es decir si introducimos identificador o nombre o descripción del objeto lo va a encontrar gracias a la función añadida en el repository que es concat de los atributos deseados.

5. Gestión de equipo

Esta función depende de la función definir roles porque se ha añadido como una acción que puede realizar el líder del equipo que es expulsar usuarios.
Una cosa importante es que por defecto el administrador de la aplicación puede realizar cualquier cosa con cualquier equipo.
Un administrador de la aplicación tiene el derecho de realizar cualquier acción con cualquiera cosa que pertenece al sistema.

6. Definir roles

Para esta funcionalidad que se ha tenido que crear una nueva tabla datos equipo usuario para guardar datos e información sobre los usuarios que se añaden al equipo, guardamos el único dato que nos hacía falta es él cuando un usuario se añada al equipo, pues necesitamos saber cómo se añade, es decir que tipo de rol se le asigna cuando suscriba al equipo.
Un ejemplo con los datos técnicos realizados en esta función:
Un usuario decide entrar en un equipo equipo01 al darle al botón añadirse por defecto se le asigna el rol participante, con dicho rol no puede hacer nada solo visualizar las tareas del equipo y sus estados (NOTA: no es que no puede realizar nada sino que no tiene visible los botones).
Me asignan otro rol Gestor, entonces puedo realizar cambios en la tareas como crear modificar cambiar de estado eliminar.
Me asignan otro rol administrador, entonces puedo realizar todas las funciones del rol Gestor y más funciones como cambiar roles a los miembros del equipo.
Otro Rol que no se puede asignar a nadie es Lider que solo puede existir un miembro con ese rol, ese miembro es el creador del equipo, por lo tanto además de poder realizar todas las funciones de administrador una función más que puede hacer el líder es expulsar miembros del equipo.

7. Crud de tarea equipo

Lo que haremos para esta tarea será replicar el apartado de tareas que tenemos ya implementado en nuestra aplicación, pero aplicadas únicamente para los equipos. Con esto lo que queremos decir es que vinculamos las tareas que creamos con un equipo en concreto.

Para ello implementaremos un CRUD de funcionalidades para estas tareas:

- Crear
- Ver
- Modificar
- Borrar

8. Definir columna de tarea (por hacer,en progreso, terminada)

En esta funcionalidad buscamos un poco que nuestra aplicación se pareciera en cierta forma a un tablero de Trello. Nosotros añadimos tres columnas fijas las cuales son:

- Por hacer
- En progreso
- Finalizada

Para realizar el cambio de estado de una tarea lo que hicimos fue añadir un campo más a las tareas de los equipos para saber en qué estado se encuentra la misma. Simplemente lo que hicimos fue añadir una template en la cual podemos seleccionar el estado en el que se encuentra la tarea o en el estado que queremos asignarle. También hicimos que cuando se crea una tarea automáticamente se le asigne el estado de *Por hacer*.

9. Asignar a la tarea fecha

Se ha añadido un campo nuevo a la tabla tareas que es fecha, la cual se realiza dicha tarea .
Cuando el usuario crea tarea nueva en el campo fecha se le despliega un calendario y tendrá que poner la fecha deseada para realizar la tarea.

Puesta en producción

En este apartado vamos a explicar como descargar la imagen de nuestro proyecto así como ponerlo en marcha para poder hacer una prueba dinámica del mismo.
Los comandos para realizar esta acción son los siguientes:

- docker pull alejandrocompany/mads-todolist:1.4.0
- docker network create network-equipo
- docker run -d --network network-equipo --network-alias postgres -v ${PWD}:/mi-host --name db-equipo -e POSTGRES_USER=mads -e POSTGRES_PASSWORD=mads -e POSTGRES_DB=mads postgres:13
- docker run --rm --network network-equipo -p8080:8080 alejandrocompany/mads-todolist:1.4.0 --spring.profiles.active=postgres-prod --POSTGRES_HOST=postgres

Una vez ejecutados todos estos comandos, lo que haríamos sería acceder a la siguiente dirección:

http://localhost:8080

Esta dirección nos redirigirá automáticamente al apartado de login/registro de nuestra aplicación.

Respecto al script de migración, hubo muchos cambios desde la versión 1.3.0 a 1.4.0.
El script es el siguiente:

```
CREATE TABLE public.datos_equipo_usuario (
    id bigint NOT NULL,
    rol character varying(255),
    equipo_id bigint,
    usuario_id bigint
);
ALTER TABLE public.datos_equipo_usuario OWNER TO mads;
CREATE SEQUENCE public.datos_equipo_usuario_id_seq
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;

ALTER TABLE public.datos_equipo_usuario_id_seq OWNER TO mads;

ALTER SEQUENCE public.datos_equipo_usuario_id_seq OWNED BY public.datos_equipo_usuario.id;

ALTER TABLE public.usuarios
ADD COLUMN enabled boolean NOT NULL,
ADD COLUMN verification_code character varying(64);

CREATE TABLE public.tareasequipo (
    id bigint NOT NULL,
    descripcion character varying(255) NOT NULL,
    estado character varying(255) NOT NULL,
    fecha date,
    titulo character varying(255) NOT NULL,
    equipo_id bigint NOT NULL
);


ALTER TABLE public.tareasequipo OWNER TO mads;

CREATE SEQUENCE public.tareasequipo_id_seq
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;


ALTER TABLE public.tareasequipo_id_seq OWNER TO mads;

ALTER SEQUENCE public.tareasequipo_id_seq OWNED BY public.tareasequipo.id;

ALTER TABLE ONLY public.datos_equipo_usuario ALTER COLUMN id SET DEFAULT nextval('public.datos_equipo_usuario_id_seq'::regclass);

ALTER TABLE ONLY public.tareasequipo ALTER COLUMN id SET DEFAULT nextval('public.tareasequipo_id_seq'::regclass);

ALTER TABLE ONLY public.datos_equipo_usuario
    ADD CONSTRAINT datos_equipo_usuario_pkey PRIMARY KEY (id);

ALTER TABLE ONLY public.tareasequipo
    ADD CONSTRAINT tareasequipo_pkey PRIMARY KEY (id);

ALTER TABLE ONLY public.datos_equipo_usuario
    ADD CONSTRAINT fk5nbu06l8dxeun3aj9agl40r4p FOREIGN KEY (usuario_id) REFERENCES public.usuarios(id);

ALTER TABLE ONLY public.datos_equipo_usuario
    ADD CONSTRAINT fk752tnn8j8mq4hesjnnkrpfftw FOREIGN KEY (equipo_id) REFERENCES public.equipos(id);

ALTER TABLE ONLY public.tareasequipo
    ADD CONSTRAINT fkmxqmetbly1abh7nsl18dmbvyb FOREIGN KEY (equipo_id) REFERENCES public.equipos(id);
```

Informe sobre la evolución del desarrollo

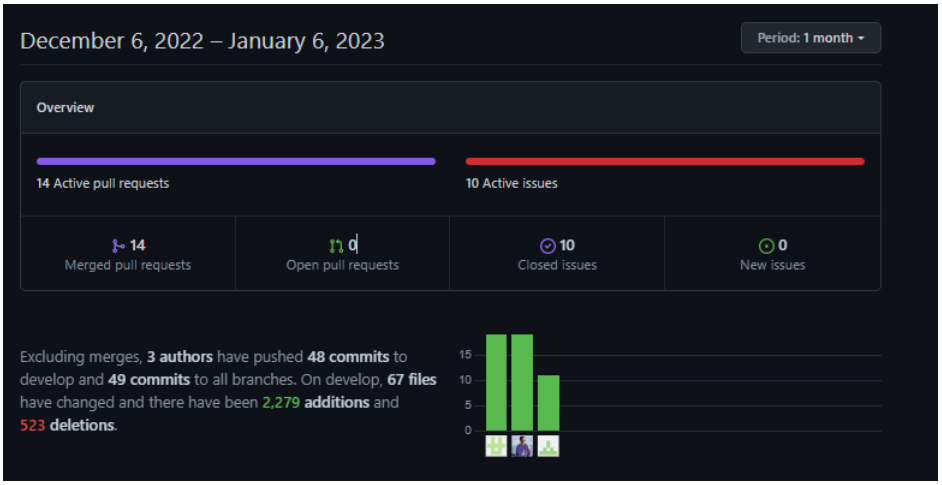
Para el desarrollo correcto de la práctica lo que hemos hecho ha sido coger y en primer lugar repartir las tareas que creamos en Trello.
Una vez que todos los integrantes del grupo teníamos asignadas las tareas que íbamos a realizar, lo que hicimos fue establecer una reunión semanal para así llevar un control más exhaustivo sobre el desarrollo de la práctica.

Las reuniones semanales fueron pequeñas reuniones de aproximadamente unos 20 minutos máximo, en las que explicamos un poco cómo había ido la semana, los problemas que habíamos encontrado y las siguiente metas que íbamos a establecernos para la siguiente reunión.

Para ver un poco más en detalle el desarrollo del proyecto vamos a poner una serie de gráficos extraídos directamente del repositorio de Github:

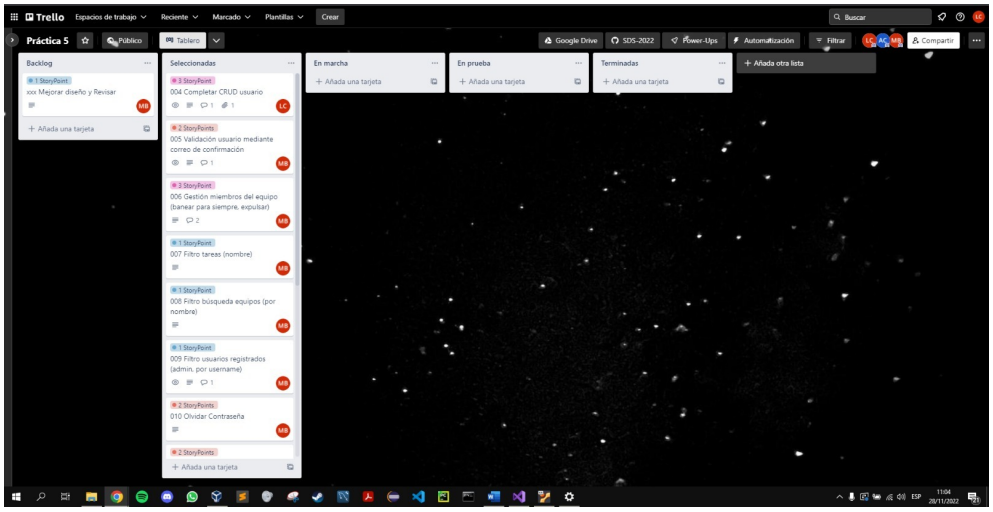
En los gráficos anteriores podemos ver un poco el desarrollo de las diferentes ramas que estaban asignadas a cada tarea, así como los diferentes commits que íbamos introduciendo.

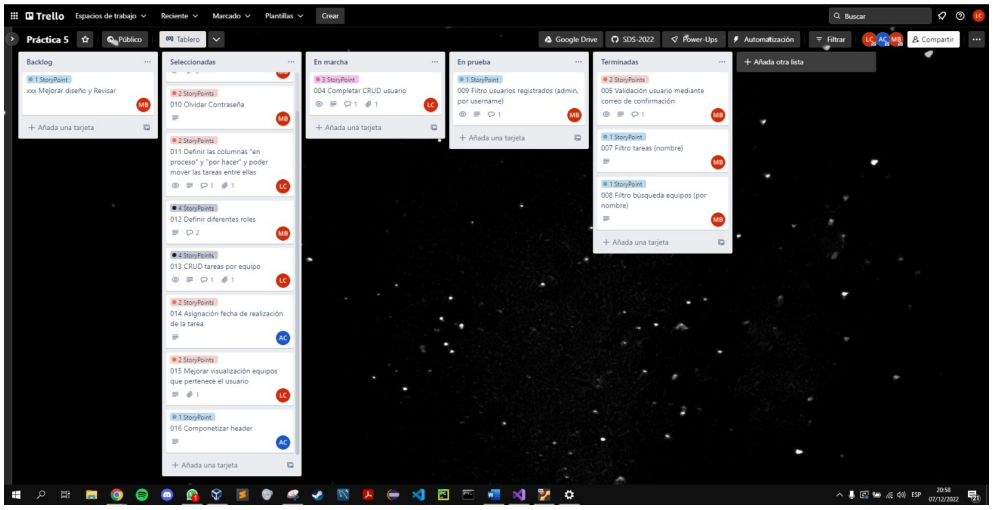
Si nos fijamos también en el número de pull request que hemos realizado podemos ver como ha sido el desarrollo del proyecto:

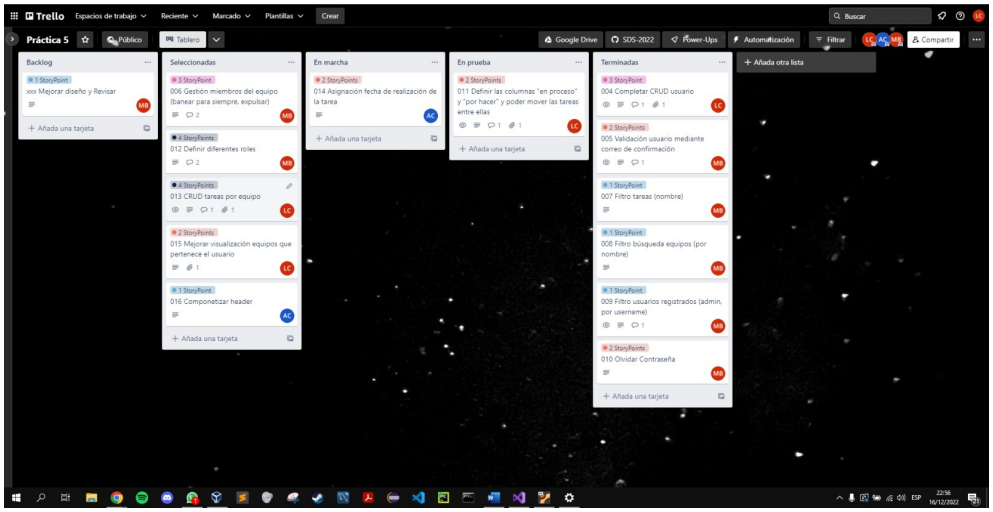


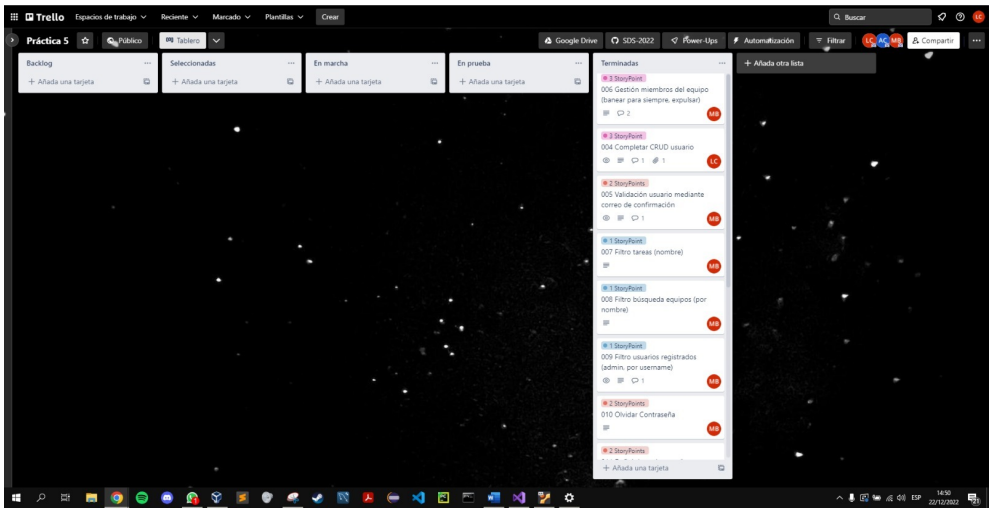
Vemos que este gráfico al ser de un único mes comienza a partir del 6 de Diciembre y no abarca por completo todo el proyecto.

También vamos a mostrar un poco más sobre cómo iba cambiando el tablero de Trello a medida que íbamos completando las tareas:









Informe Sesiones Pair Programming

Para la realización de las sesiones de pair programming hicimos un sorteo para ver cómo iban a ser los emparejamientos y salió lo siguiente:

- **Primera Sesión**: Adel con Luis
- **Segunda sesión**: Alejandro con Luis

Cómo vivimos cada uno en una zona diferente y estamos los tres trabajando, fue un poco difícil definir un horario que nos viniera bien a todos. Decidimos que la mejor idea era trabajar con una reunión online en la que el pair programming lo haríamos en un único ordenador haciendo uso de Team Viewer.

Primera sesión

En esta sesión resolvimos la tarea *“Filtro equipo por nombre”*. Para ello lo primero que hicimos fue organizarnos un poco y hacer un pequeño esquema de los pasos que íbamos a realizar durante la sesión:

1) Buscar información sobre cómo hacer la tarea, es decir buscar ejemplos sobre posibles soluciones parecidas a lo que queremos implementar
2) Proponer una solución posible para el problema
3) Ponernos a programar juntos, los primeros 20 minutos Adel y luego dejaba el control del PC a Luis y él seguía programando.
4) Verificamos y damos el visto bueno a la tarea ambos
5) Se hacía el merge con la Develop

Segunda sesión

Para la segunda sesión decidimos que era buena idea seguir con la metodología que habíamos aplicado para realizar la anterior. Por lo que seleccionamos la tarea de *“Añadido lista de mis equipos (cuenta)”.*

A la hora de realizar el tercer paso esta vez, empezó programando Luis y luego siguió Alejandro.

Resultado de la retrospectiva

Si analizamos todo el desarrollo que hemos realizado de la práctica y el producto final que hemos entregado, podemos sacar varias conclusiones o varios aspectos a mejorar.

Uno de los principales objetivos que nos hubiera gustado llegar hacer hubiera sido implementar más funcionalidades. Tener una parte pública de la aplicación, es decir antes de crear una cuenta tener una parte para poder conocer bien la aplicación antes de registrarse, ya que en nuestro caso únicamente tenemos el apartado de *About* donde se detalla poco sobre el proyecto.

Una cosa que nos gustaría mejorar es el diseño de la web, sobre todo el apartado de los botones. Creemos que hay muchas funcionalidades que no son del todo claras para acceder a ellas o que simplemente no se conocen que están ahí por desconocimiento de no haber visto el botón. Si que es verdad que intentamos que cada botón tuviera un color diferente en función de la tarea que iba a realizar.

Finalmente, en un principio creíamos que habíamos seleccionado una cantidad de tareas y funcionalidades bastante grande y teníamos dudas de poder acabarlas todas. Por suerte y después de haber realizado un trabajo muy bueno y haber formado un gran equipo pudimos sacar todo adelante.

Enlaces

- **Trello** → enlace Trello
- **Imagen-Docker(versión 1.4.0)** → enlace Docker Hub
- **Github** → enlace Github