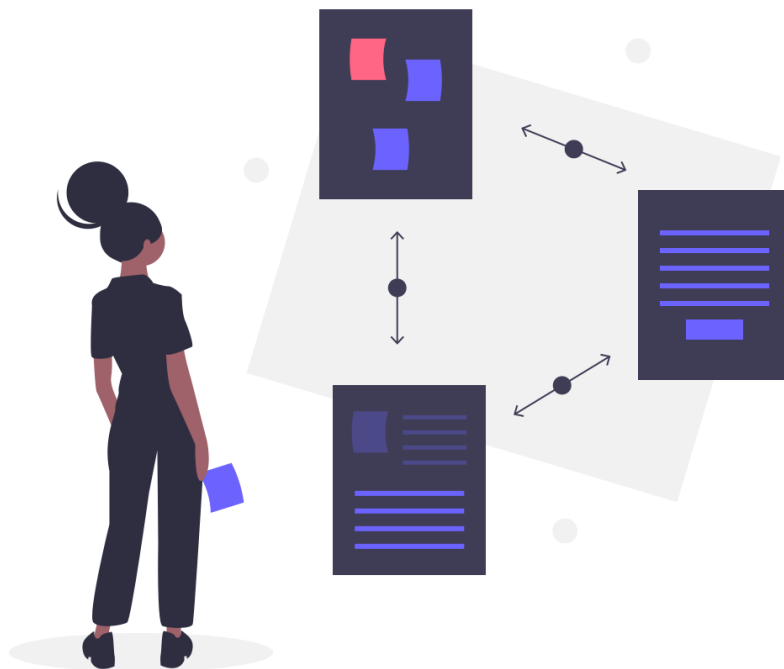


MEMORIA

Proyecto de final de curso



Jaime Martínez Rincón

Desarrollo de Aplicaciones Multiplataforma

Curso 2019 – 2020 | IES Francisco de Goya

Índice

Contenido

Índice	2
1. Introducción	4
1.1 Estructura	5
1.1.1 Infraestructura de despliegue	5
1.1.2 Base de datos	5
1.2 Requisitos y objetivos.....	7
1.2.1 Escala de la aplicación	7
1.3 Tecnologías.....	9
1.3.1 Tecnologías/herramientas Comunes	9
1.3.2 Tecnologías específicas del frontend	15
1.3.3 Tecnologías específicas del backend	19
1.3.4 Tecnologías específicas del cliente de escritorio	23
2. Análisis del funcionamiento de la aplicación	25
2.1 Plataforma web	25
2.1.1 Página de inicio de sesión	25
2.1.2 Página de registro.....	26
2.1.3 Página de recuperación de contraseña	27
2.1.4 Página de lista de proyectos.....	28
2.1.5 Página de información de proyecto	29
2.1.6 Página de sesión	30
2.1.7 Página de configuración de proyecto.....	31
2.2 Aplicación de escritorio	32
2.2.1 Vista de inicio de sesión	32
2.2.2 Vista de lista de proyectos	32
2.2.3 Vista de sesión.....	33
3. Pruebas.....	34
Prueba 1	34
Prueba 2	34
Prueba 3	34
Prueba 4	34
Prueba 5	35
Prueba 6	35
Prueba 7	35

Prueba 8	35
Prueba 9	36
Prueba 10	36
Prueba 11	36
Prueba 12	36
4. Conclusiones.....	37
4.1 Grado de logro de objetivos iniciales	37
4.2 Futuras líneas de investigación	37
4.3 Dificultades resueltas más destacables.....	39
Uso de React Hooks.....	39
Diseño de la API REST.....	39
Código Multiplataforma en C++	39
Autenticación	40
Permisos	40
Diseño de la interfaz.....	40
Consultas SQL de estadísticas	41
Tema oscuro de componentes.....	43
Rechazos del servidor de correo	43
5. Menciones especiales	43
6. Bibliografía	44

1. Introducción

Este documento forma parte del desarrollo de la aplicación “Time It” que he realizado como proyecto final del Grado Superior de Desarrollo de Aplicaciones Multiplataforma.

Mi proyecto es una plataforma para la gestión y monitorización del tiempo, puedes crear proyectos e iniciar sesiones de trabajo, en las cuales se almacenarán tus actividades (ventanas que has usado) y el tiempo que han durado.

Las ventanas que hemos usado son monitorizadas por un cliente de escritorio básico el cual envía información de la actividad actual al servidor cada 5 segundos.

Esta información se accede a través de una página web en la que puedes crearte una cuenta.

He creado este proyecto porque a mí me hubiese venido bien en el pasado tener una plataforma completamente gratuita y de código abierto para monitorizar mi actividad mientras programaba. Además, este proyecto me ha permitido aprender de tecnologías modernas y punteras las cuales quería aprender desde hace bastante tiempo.

Los usuarios potenciales pueden ser individuos los cuales quieren monitorizar su actividad con fin de mejorar la eficiencia, o trabajadores “freelancer” que quieren facturar las horas invertidas por proyecto. Pienso que también puede ser útil para empresas que requieran de una herramienta para monitorizar el trabajo de empleados teletrabajando.

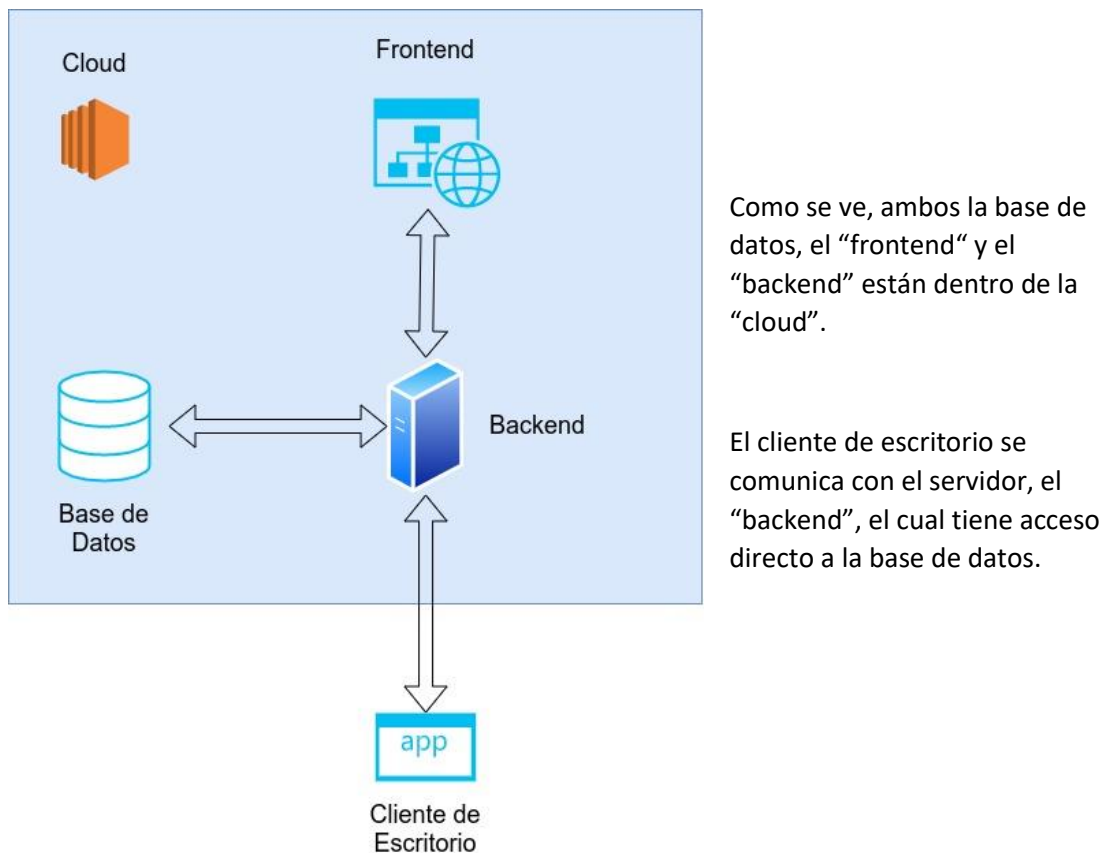
1.1 Estructura

1.1.1 Infraestructura de despliegue

La aplicación consta de 4 partes el “frontend”, el “backend”, la base de datos y una aplicación de escritorio.

La aplicación de escritorio sería distribuida en forma de un “.exe” para Windows y de un archivo ejecutable ELF para Linux.

El resto de la plataforma sería desplegada en la nube, como puede ser AWS EC2, lo cual nos daría una gran fiabilidad y sería gratis durante un tiempo gracias a los créditos para estudiante además de la capa gratuita la cual dura 1 año.



1.1.2 Base de datos

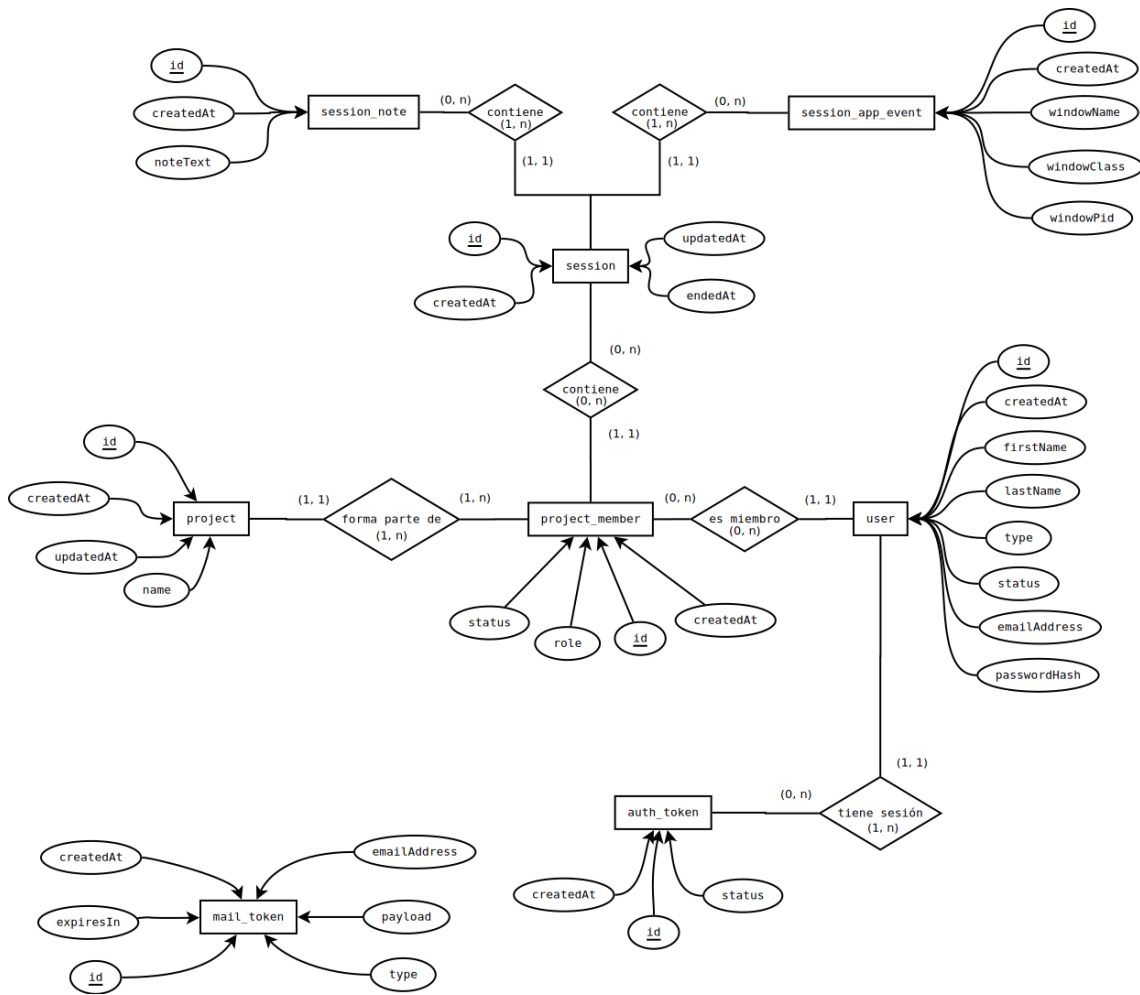
La plataforma requiere del uso de una base de datos SQL, en este caso es PostgreSQL.

El diseño de la estructura de la base de datos se ha centrado en funcionar bien con el ORM que usamos, TypeORM.

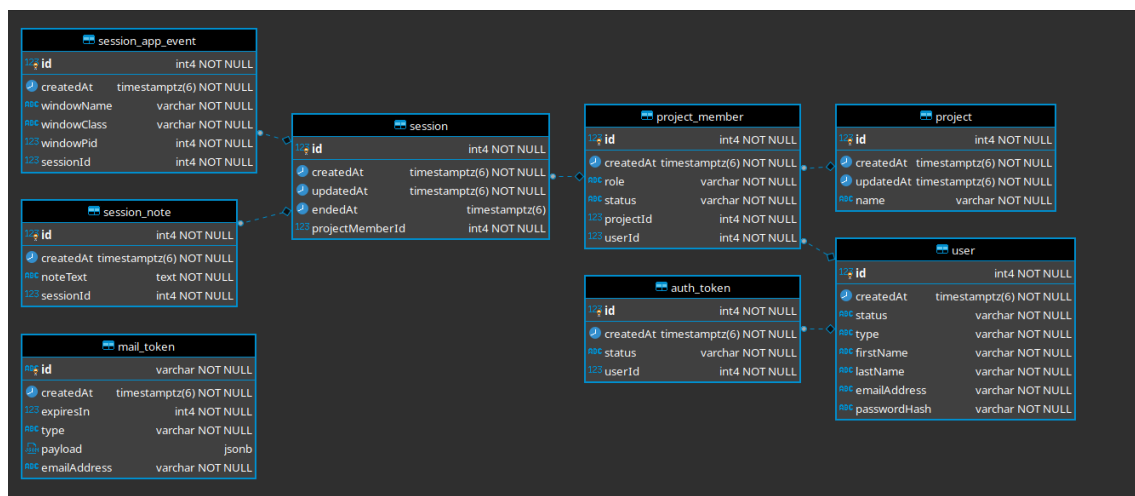
Este ORM tiene ciertas restricciones la prohibición de usar múltiples columnas como índice primario, por eso en la mayoría de los casos he tenido que recurrir a columnas simples con enteros automáticamente incrementados.

Además, se han realizado ciertas optimizaciones para dar más flexibilidad a la aplicación, como el uso de blobs JSON para guardar la carga (o “payload”) de algún objeto en la base de datos.

Este es un diagrama entidad / relación del resultado final de la base de datos.



También podemos ver un diagrama generado automáticamente, sin cardinalidades, pero donde podemos ver los tipos de las columnas y las relaciones.



1.2 Requisitos y objetivos

El proyecto tiene como objetivo crear una plataforma para monitorizar las horas invertidas por proyecto. Las características que me plantee fueron:

- Un sistema de inicio de sesión basado en correo electrónico y contraseña (y cierre de sesión)
- Un sistema de registro con confirmación de cuenta mediante correo electrónico
- Un sistema de recuperación de contraseña mediante correo electrónico
- Una aplicación para enviar a la plataforma las ventanas de escritorio activas
- Una página de creación de proyectos en los que puede haber múltiples miembros con diferentes permisos (admin, manager, trabajador)
- Una página para ver la información del proyecto
 - o Gráfico de líneas con las horas invertidas por día por miembro
 - o Filtrar por miembro del grupo (si tenemos permiso)
 - o Seleccionar el rango de tiempo a mostrar
 - o Ver una lista las sesiones creadas con la duración y quien la creó.
- Una página de configuración para los admin y manager donde se pueda cambiar el nombre del proyecto, invitar a usuarios, expulsar a miembros, ascender/descender grupo de permisos o borrar el proyecto.
- Un sistema de confirmación de invitación a un proyecto mediante correo electrónico.
- Sistema para cambiar el tema de la aplicación web a un modo oscuro básico.
- Una interfaz moderna, minimalista y bonita, para ambos la plataforma web y el cliente de escritorio.
- La plataforma tiene que ser eficiente en tema de recursos y rápida.

1.2.1 Escala de la aplicación

1.2.1.1 Backend

La aplicación está diseñada para soportar tantos usuarios y conexiones como el hardware pueda.

El usar Node.js como lenguaje de servidor significa que nuestra aplicación escalará bien en procesadores multinúcleo sin complicarnos.

Además, nuestro “backend” es prácticamente “stateless” (sin estado) por lo que el uso de memoria es mínimo.

1.2.1.2 Frontend

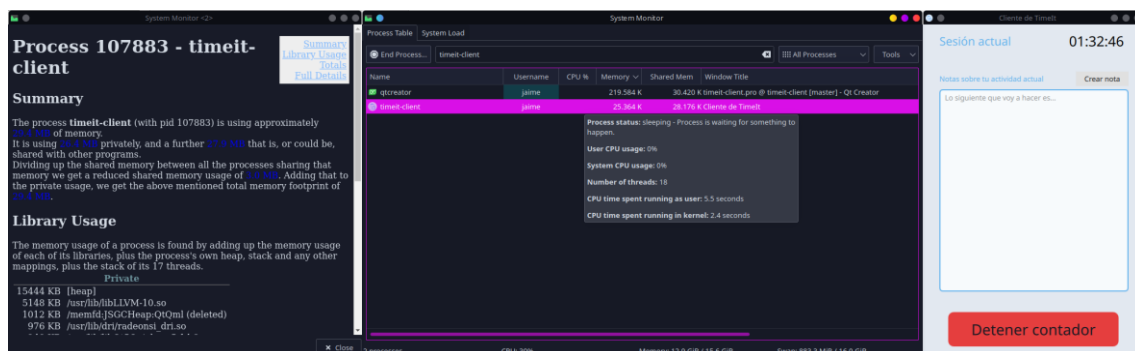
La aplicación puede ser servida de forma estática, no hace falta que se ejecute ninguna lógica al servir la página web. Sería perfectamente posible descargarse los archivos HTML y JS generados al construir la aplicación y usarla de esa manera.

Obviamente, para realizar cualquier operación, dependería del “backend”, pero servir el “frontend” sería muy fácil y barato, esto es lo conocido como “arquitectura serverless”.

React es una librería muy eficiente y rápida, por lo que se adapta a dispositivos con especificaciones bajas. Además, al construir nuestra aplicación usando “Create React App” (la utilidad por defecto para empezar a programar una aplicación usando React) todo el código y todos los recursos se optimizan y están minificados para que sean mucho más ligeros comparados con sus archivos originales.

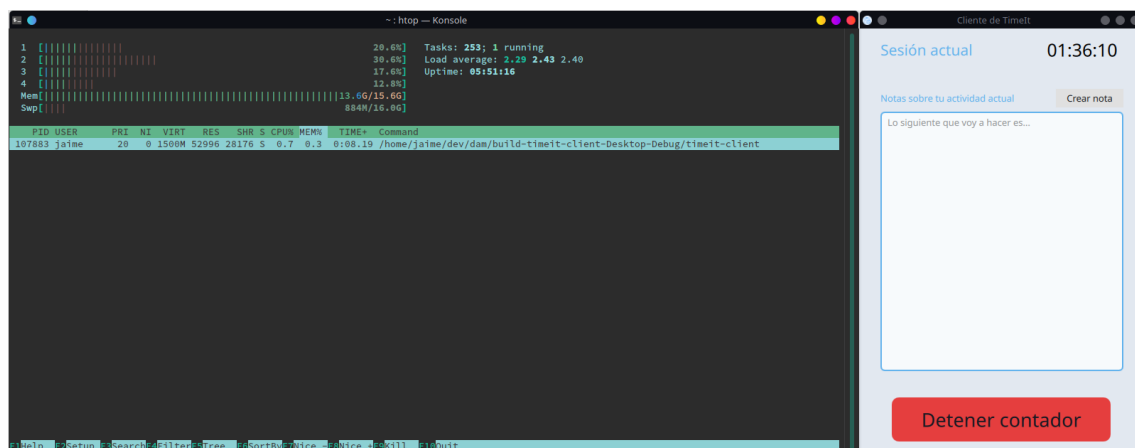
1.2.1.3 Aplicación de escritorio

La utilización de Qt significa que el consumo de recursos de la aplicación es mínimo. Mientras que la app está funcionando y reportando los cambios de actividad al servidor, usa aproximadamente 29.4 MB de memoria y como la aplicación la mayoría del tiempo está durmiendo, no usa casi nada de CPU.



Cada 5 segundos, que es cuando la aplicación reporta los cambios de actividad, se puede observar un uso de 0.3% - 1.2% de CPU. Esto se ha probado con un procesador Intel Core i5 4460 (procesador de gama baja teniendo en cuenta los estándares de hoy en día).

Y esto es ejecutando la aplicación compilada en modo “debug”, sin optimizaciones. Si ejecutásemos la aplicación en el modo “release”, con todas las optimizaciones por defecto aplicadas, el uso de recursos de la aplicación podría mejorar bastante.



Pienso que el que haya usado Qt para la aplicación de escritorio ha hecho que resulte en una aplicación mucho más fácil de mantener, eficiente y agradable estéticamente que si hubiese utilizado otras tecnologías.

1.3 Tecnologías

Para el desarrollo de esta aplicación he usado una gran variedad de tecnologías las cuales me han ayudado a desarrollar la aplicación eficientemente y no preocuparme mucho por la parte de configuración de servidor.

1.3.1 Tecnologías/herramientas Comunes

Para el desarrollo de este proyecto he usado muchas herramientas y tecnologías comunes entre las diferentes partes de la aplicación.

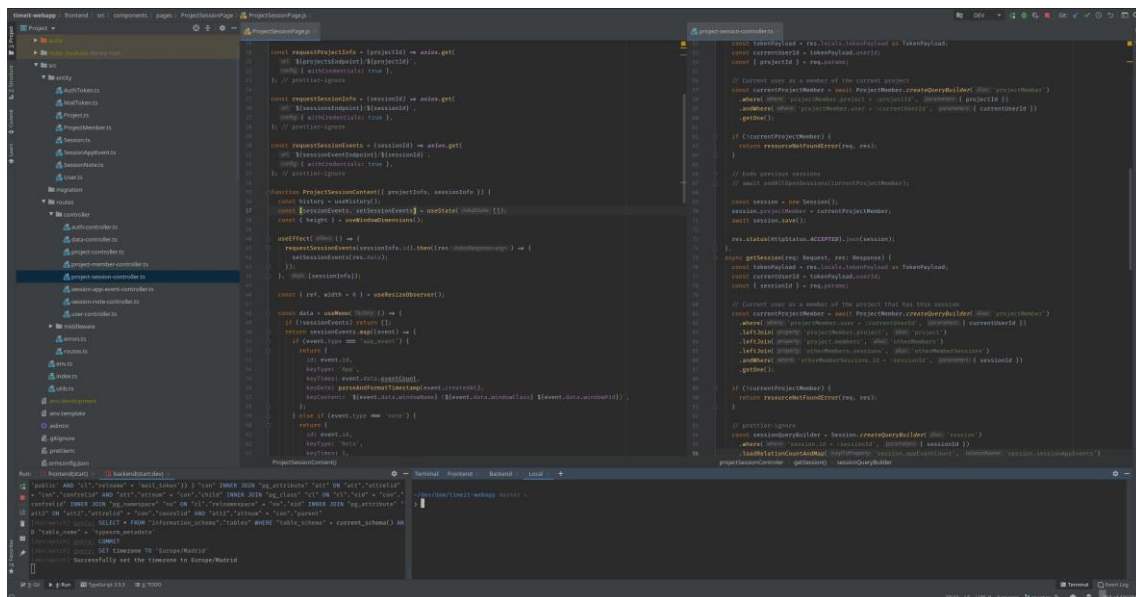
1.3.1.1 WebStorm (<https://www.jetbrains.com/es-es/webstorm/>)

Para programar he usado el IDE WebStorm, que forma parte de la suite de IDEs de JetBrains, los mismos creadores de Android Studio.

Prácticamente todos los IDEs de JetBrains son gratuitos mientras somos estudiantes, y en mi opinión, aunque consume demasiados recursos, es una herramienta extremadamente potente que cuando conoces bien puede hacer tu experiencia al desarrollar mucho más eficiente.

Puedes dividir el editor en varias partes, yo normalmente tengo el editor dividido en dos columnas, una para la parte del backend y otra para la parte del frontend.

Debajo, tengo las pestañas del frontend y el backend ejecutándose y al lado tengo 3 terminales para, una para el frontend, otra para el backend y otra para la raíz del repositorio Git.

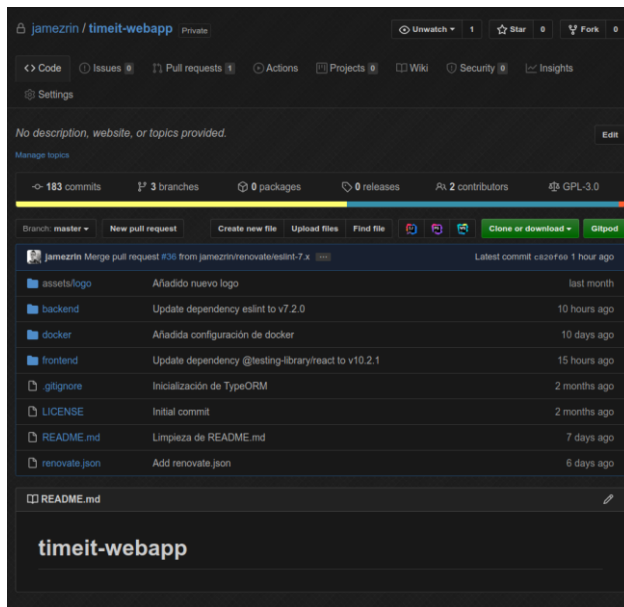


1.3.1.2 Git + GitHub (<https://github.com/>)

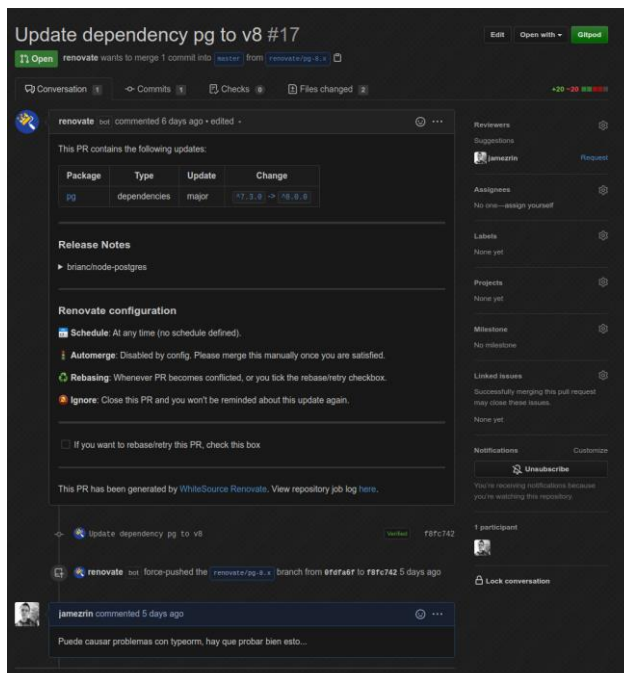
Para tener controlados los cambios que realizo a la aplicación y poder compartir el código entre ordenadores, uso Git. Para ello a veces uso la integración de WebStorm y otras veces utilizo la terminal directamente, depende de lo que esté haciendo.

El proyecto está dividido en dos repositorios, uno para la plataforma web y otro para el cliente de escritorio.

Este es el repositorio que he usado para la plataforma web.



Como se ve, tengo cuatro directorios, uno para la configuración de los contenedores docker, otro para el frontend, otro para el backend y otro para los assets comunes entre todas las aplicaciones.

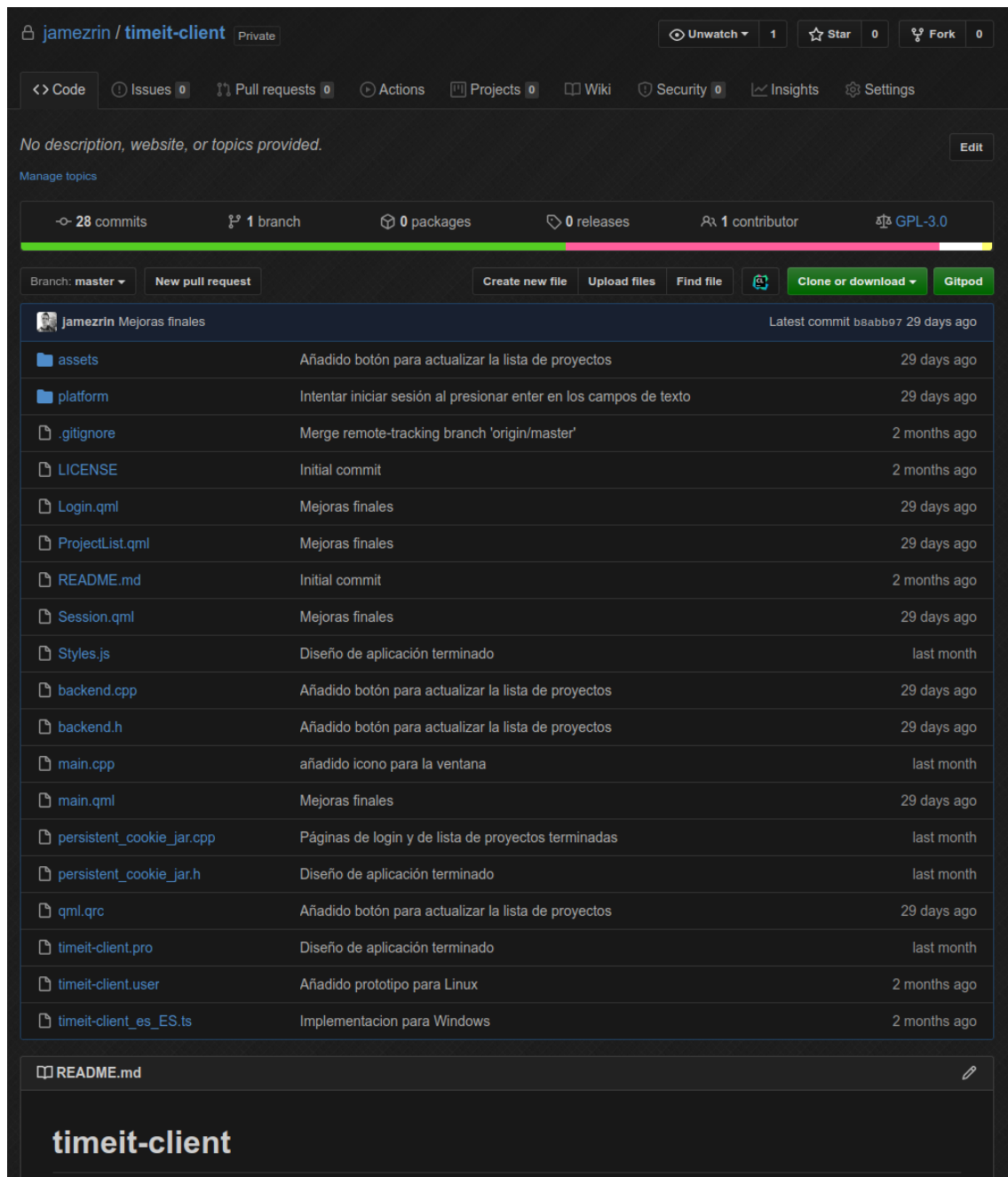


En el repositorio de la plataforma web, he instalado un bot llamado “Renovate” el cual te ayuda a que las dependencias usadas estén siempre a la última versión.

Este bot te crea “Pull Requests” en las cuales puedes ver los cambios entre las versiones y hacer un “Merge” si es que está todo bien.

De esta forma, me puedo asegurar que las librerías que uso tienen los últimos arreglos y parches de seguridad.

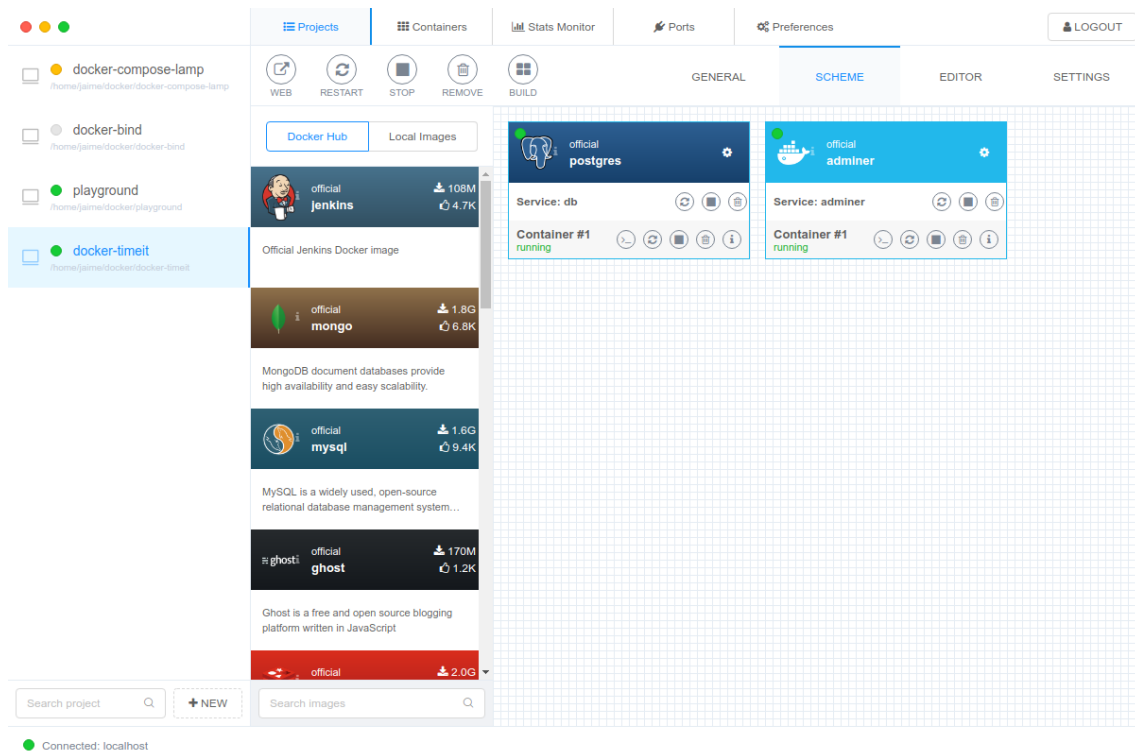
Este es el repositorio que he usado para la aplicación de escritorio.



1.3.1.3 Docker + Dockstation (<https://www.docker.com/>)

Para el servidor de base de datos he usado un par de contenedores Docker, los cuales a efectos prácticos funcionan como máquinas virtuales, pero sin un sistema operativo completo por debajo. Es mucho más eficiente en cuanto al uso de recursos que máquinas virtuales tradicionales.

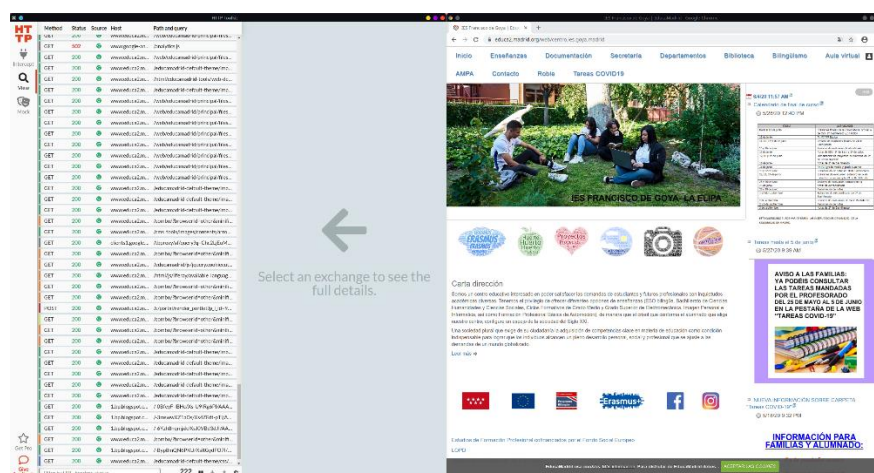
Para hacer más fácil el uso de Docker he usado Dockstation, una aplicación gratuita para manejar contenedores Docker de forma gráfica.



Como se ve, en mi proyecto “docker-timeit” tengo dos contenedores, uno para la base de datos “postgres” y otro para “adminer”, un gestor de base de datos parecido a “phpMyAdmin”.

1.3.1.4 HttpToolkit (<https://httptoolkit.tech/>)

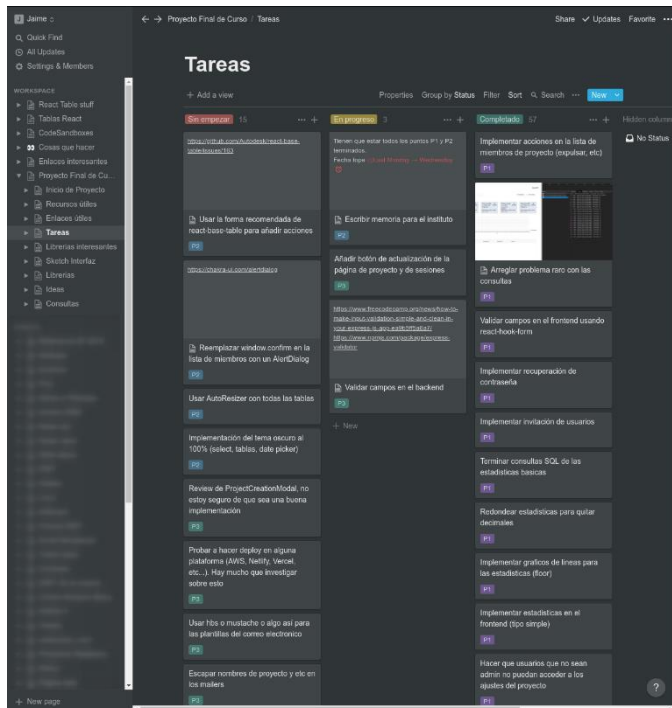
Para depurar las peticiones entre el backend y la aplicación de escritorio, he usado este programa el cual me permite ponerme en medio de los dos y ver con detalle las peticiones http que se realizan. Es gratuito y de código abierto.



Esto me ha ayudado a resolver conflictos entre la conexión de estas dos partes del proyecto.

Funciona en forma de un proxy, lo configuras en la aplicación que quieres interceptar y todas las peticiones realizadas pasarán a través de HttpToolkit, funcionando como un MITM (Man in the middle)

1.3.1.5 Notion (<https://www.notion.so/product>)



Esta herramienta me ha ayudado a llevar un control de las tareas y información relacionada con el proyecto.

Es totalmente gratuita y tiene todas las funcionalidades que te puedes imaginar.

Las funcionalidades que más he usado en el desarrollo de este proyecto han sido el tablero Kanban para distribuirme las tareas y los documentos, en los que me apuntaba información a tener en cuenta.

1.3.1.6 REST + JSON

Para toda la transmisión de datos he utilizado peticiones HTTP siguiendo la arquitectura REST y devolviendo documentos JSON para representar la información.

La arquitectura REST se basa en los métodos de las peticiones HTTP, donde el verbo o método indica la acción a realizar sobre un recurso.

Por ejemplo, si tenemos un recurso llamado “sessions” podríamos crear las siguientes rutas REST.

Método	Ruta	Acción
GET	/sessions/:sessionId	Visualiza la instancia del recurso especificado
GET	/sessions/	Lista todas las instancias del recurso (o una parte)
POST	/sessions/	Crea una nueva instancia del recurso
PATCH	/sessions/:sessionId	Actualiza la instancia del recurso especificado con nueva información
DELETE	/sessions/:sessionId	Elimina la instancia del recurso especificado
POST	/sessions/:sessionId/some_action	Realiza una acción no estándar de REST sobre una instancia

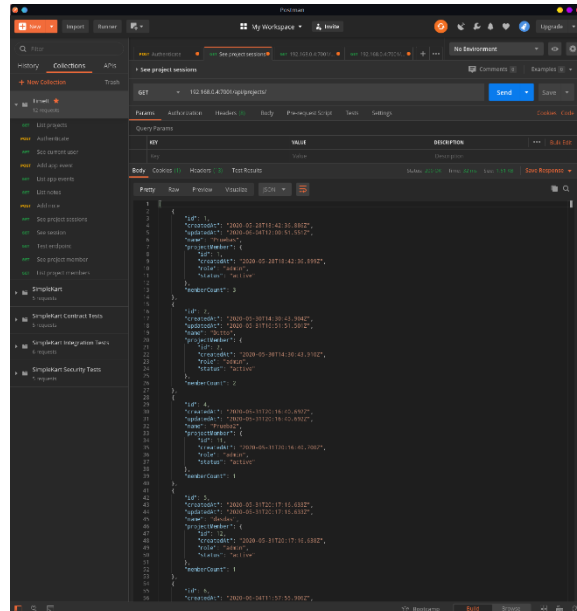
El tipo de documento JSON (JavaScript Object Notation) es una forma simple de mostrar objetos de JavaScript en texto plano. Es un estándar el día de hoy y basado en las estadísticas se usa más que SOAP, gracias a su simplicidad y compatibilidad con los diferentes lenguajes de programación.

1.3.1.7 Postman (<https://www.postman.com/>)

Cuando empecé el proyecto, decidí primero hacer la parte del backend y cuando estuviese listo empezar con el frontend.

Para probar el backend necesitaba hacer peticiones HTTP desde algún lado, simulando las acciones de un navegador.

Para eso use esta aplicación, la cual te permite crear peticiones y recibir una respuesta del servidor. Esto ha sido una parte crucial del desarrollo del backend.



1.3.1.8 Node.js (<https://nodejs.org/es/>) + Yarn (<https://yarnpkg.com/>)

Ambos el frontend y el backend están basados en Node.js para funcionar.

Node.js es un entorno para ejecutar JavaScript fuera del navegador, donde JavaScript estaba enfocado tradicionalmente.

Prácticamente todo en Node.js se ejecuta de forma asíncrona y gracias a esto se pueden construir aplicaciones muy rápidas y eficientes, si las programamos bien.

Node.js tiene su propio gestor de paquetes llamado “npm”, el cual contiene más de 350.000 mil paquetes hasta la fecha.

Yo he usado “yarn” que es parecido a “npm” (es más, los paquetes los saca de “npm”) pero es bastante más rápido y completo en cuanto a funcionalidad.

1.3.1.9 Prettier (<https://prettier.io/>) + ESLint (<https://eslint.org/>)

Prettier es un formateador de código que funciona de forma automática y te fuerza a tener un estilo y formato de código consistente y entendible en toda la aplicación. Soporta una gran cantidad de lenguajes de programación, pero en este proyecto lo he usado especialmente en el frontend y en el backend para la parte de JavaScript y Typescript respectivamente.

En cuanto a ESLint, es un comprobador de código que nos avisa si usamos funcionalidades en desuso o si hacemos cosas que no están bien, esto me ha ayudado mucho para aprender React.

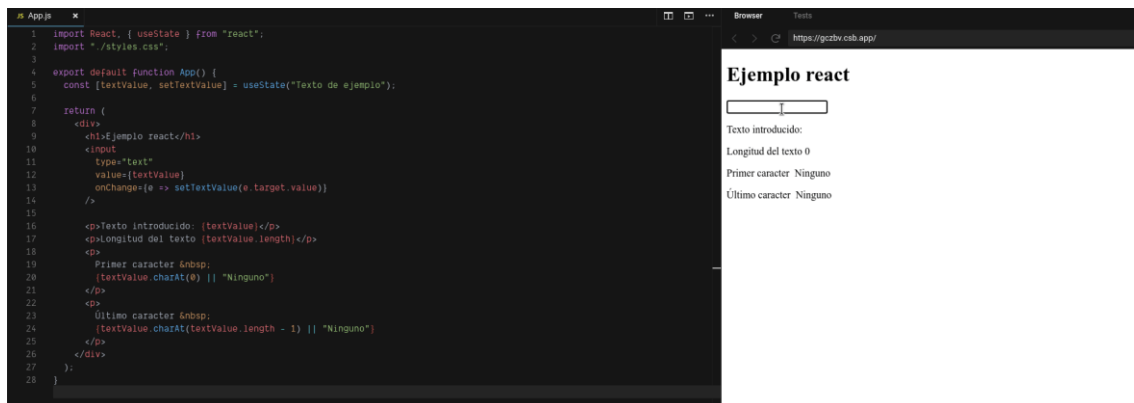
1.3.2 Tecnologías específicas del frontend

1.3.2.1 React (<https://es.reactjs.org/>)

React es una librería para construir lo que sería la parte de la vista de una aplicación. Esta librería se centra en el uso de componentes para separar las distintas partes de la aplicación. Los componentes están escritos en JavaScript y devuelven JSX, que es algo parecido a HTML pero mezclado con JavaScript.

Estos componentes pueden guardar su propio estado y compartirlo con otros componentes si es necesario o dejarlo encapsulado en cada componente.

En este ejemplo, se puede ver que al cambiar el texto no hace falta escuchar al evento de *onChange* y cambiar todos los párrafos que dependen de él, simplemente con cambiar el estado ya se actualiza donde se esté usando.



Video de demostración: <https://youtu.be/FVC6G4LJ1Fk>

Esto es lo básico de React, pero es mucho más que esto. Lo bueno es que aprender lo básico de React es muy sencillo, y se puede aprender progresivamente, no tiene una gran curva de aprendizaje.

Eso sí, React por sí solo es bastante sencillo, hay que complementarlo con otras librerías. En otras frameworks como Angular te viene todo en un paquete.

Yo prefiero la modularidad de React, que me permite elegir las partes que yo quiera para realizar la aplicación, pero todo depende de gustos.

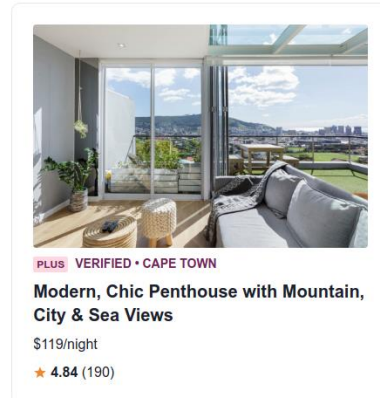
1.3.2.2 Chakra UI (<https://chakra-ui.com/>)

Chakra UI es una librería de componentes inspirada en Tailwind CSS. Se podría comparar con Bootstrap pero es específica para React y nos permite incluir componentes con su propio estilo de una forma que es muy elegante y que funciona muy bien con React.

Esta librería está enfocada a ser modular, simple, fácil de usar, agradable visualmente y accesible por todo tipo de usuarios.

```
// Sample component from airbnb.com

<Box>
  <Image rounded="md" src="https://bit.ly/2k1H1t6"/>
  <Flex align="baseline" mt={2}>
    <Badge variantColor="pink">Plus</Badge>
    <Text
      ml={2}
      textTransform="uppercase"
      fontSize="sm"
      fontWeight="bold"
      color="pink.800"
    >
      Verified &bull; Cape Town
    </Text>
  </Flex>
  <Text mt={2} fontSize="xl" fontWeight="semibold" lineHeight="short">
    Modern, Chic Penthouse with Mountain, City & Sea Views
  </Text>
  <Text mt={2}>$119/night</Text>
  <Flex mt={2} align="center">
    <Box as={MdStar} color="orange.400" />
    <Text ml={1} fontSize="sm"><b>4.84</b> (190)</Text>
  </Flex>
</Box>
```



1.3.2.3 axios (<https://github.com/axios/axios>)

Axios es una librería que permite hacer todo tipo de peticiones HTTP. Cada vez que se carga una página se hacen las peticiones necesarias al backend usando esta librería. Se usan las cookies almacenadas en el navegador usando la opción de configuración “withCredentials”.

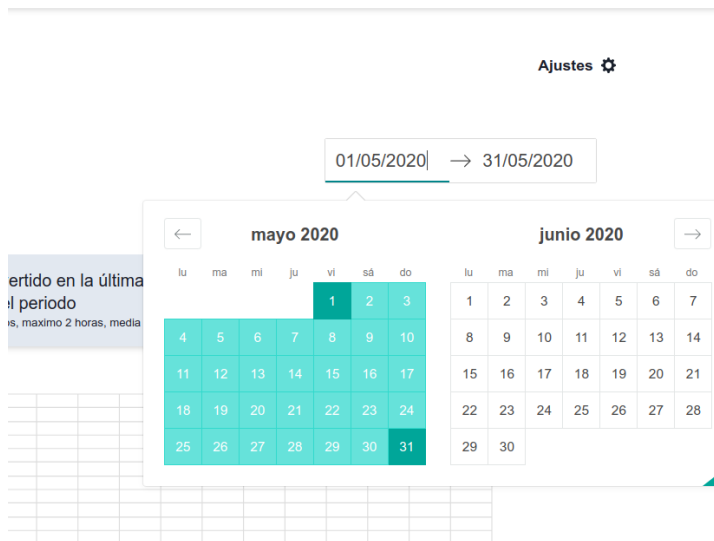
```
import axios from 'axios';

const requestSessionInfo = (sessionId) => axios.get(
  `http://localhost:7001/api/sessions/${sessionId}`,
  { withCredentials: true }
);

requestSessionInfo(10).then(res => {
  console.log('Respuesta: ', JSON.stringify(res.data));
}).catch(err => {
  console.err(err);
});
```


1.3.2.4 react-dates (<https://github.com/airbnb/react-dates>)

Esta librería desarrollada por Airbnb nos provee de componentes para elegir fechas. En este proyecto es usada para elegir el rango de fechas para las estadísticas de sesiones.



1.3.2.5 react-base-table (<https://github.com/Autodesk/react-base-table>)

Esta librería nos permite crear tablas con funcionalidad avanzada (ordenación, filtros, búsqueda, redimensión, etc) fácilmente, sin tener que usar las tablas básicas de HTML y tener que reinventar mucho código. Este componente es usado en la lista de sesiones, de miembros y de eventos.

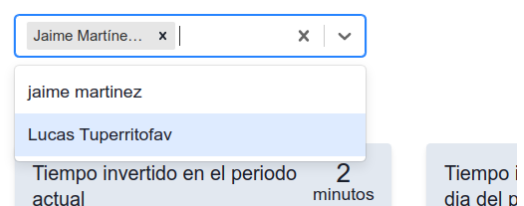
Sesiones (8)

Id	Creación	Actualización	Finalización	Usuario
15	19:01:36 31/05/2020	19:02:25 31/05/2020	19:02:25 31/05/2020	Jaime Martínez Rincón
7	18:52:39 30/05/2020	19:21:38 30/05/2020	19:21:38 30/05/2020	Jaime Martínez Rincón
5	17:26:44 30/05/2020	18:50:52 30/05/2020	18:50:52 30/05/2020	Jaime Martínez Rincón
6	18:44:38 30/05/2020	18:50:50 30/05/2020	18:50:50 30/05/2020	Jaime Martínez Rincón
4	00:39:32 30/05/2020	02:27:06 30/05/2020	02:27:06 30/05/2020	Jaime Martínez Rincón
3	21:59:46 29/05/2020	00:39:31 30/05/2020	00:39:31 30/05/2020	Jaime Martínez Rincón
2	19:44:12 29/05/2020	21:59:45 29/05/2020	21:59:45 29/05/2020	Jaime Martínez Rincón

1.3.2.6 react-select (<https://react-select.com/>)

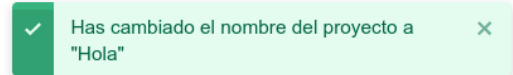
Esta librería nos permite crear desplegables donde se pueden seleccionar múltiples elementos de una lista.

Este componente se usa para elegir la lista de miembros para ver las estadísticas.



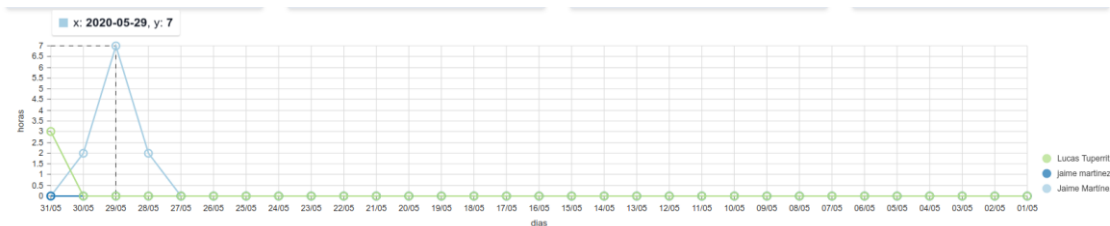
1.3.2.7 react-toast-notifications (<https://jossmac.github.io/react-toast-notifications/>)

Esta librería nos permite mostrar notificaciones al usuario de una forma muy intuitiva y en mi opinión, el estilo por defecto es muy bonito.



1.3.2.8 nivo (<https://nivo.rocks/>)

Esta librería nos permite crear todo tipo de gráficos. Está construida sobre d3 (una librería muy popular de visualización de datos) y es muy fácil de usar. Además, tiene buena apariencia.



1.3.2.9 yup (<https://github.com/jquense/yup>)

Esta librería nos permite validar datos creando un “schema” que designa la estructura de los datos y la estrategia de validación. Esta librería es usada en todos los formularios, para validar que son del tipo correcto o mostrar un aviso al usuario.

```
const schema = yup.object().shape({
  emailAddress: yup.string().email().required(),
  password: yup.string().required(),
});
```

1.3.2.10 emotion (<https://emotion.sh/docs/introduction>)

Esta librería nos permite usar CSS dentro de componentes React. De esta forma, podemos crear componentes que encapsulan el CSS. Al modificar el estilo de un componente, lo tenemos directamente ahí. Este CSS es optimizado y se une en un bundle para que cargue rápido.

```
const TableWrapper = styled.div`
  flex-grow: 1;

  .BaseTable__row {
    cursor: pointer;
  }
`;
```

1.3.3 Tecnologías específicas del backend

1.3.3.1 Typescript (<https://www.typescriptlang.org/>)

Para el backend, he decidido usar Typescript como lenguaje de programación.

Typescript es esencialmente JavaScript con tipos. Está ganando mucha popularidad por su robustez y la integración con IDEs.

Typescript tiene que ser compilado a JavaScript, ya sea con JIT (just in time – justo en el momento) o antes de ejecutarlo con Node.

```
export const accessTokenCookieName = 'timet_accessToken';

export interface TokenPayload {
  readonly tokenId: string;
  readonly userId: string;
}

export function authMiddleware(ignoreExpiration: boolean) {
  return async function (req: Request, res: Response, next: NextFunction) {
    const accessToken = req.cookies[accessTokenCookieName];

    if (!accessToken) {
      return noAccessTokenError(req, res);
    }

    try {
      const decodedPayload = (await jwt.verify(accessToken, process.env.TIMEIT_JWT_SECRET, {
        ignoreExpiration: ignoreExpiration,
      })) as TokenPayload;

      const tokenInfo = await AuthToken.findOneOrFail(decodedPayload.tokenId);

      if (tokenInfo.status !== AuthTokenStatus.ACTIVE) {
        await tokenInfo.remove();
        return inactiveTokenError(req, res);
      }
    }
  };
}
```

1.3.3.2 TypeORM (<https://typeorm.io/>)

Esta librería es un ORM basado que funciona sobre Node, está pensado para ser usado junto a Typescript. Es un ORM muy fácil de utilizar y muy potente si sabes utilizarlo.

Nos permite crear clases para los modelos, relaciones entre modelos, joins, migraciones, uso de patrón de ActiveRecord, crear nuestros propios DAOs, consultas directas, etc.

```
import {Entity, PrimaryGeneratedColumn, Column} from "typeorm";

@Entity()
export class User {

  @PrimaryGeneratedColumn()
  id: number;

  @Column()
  firstName: string;

  @Column()
  lastName: string;

  @Column()
  age: number;
}
```

En la página web oficial dicen que se han basado en frameworks como Hibernate.

```
const user = new User();
user.firstName = "Timber";
user.lastName = "Saw";
user.age = 25;
await user.save();

const allUsers = await User.find();
const firstUser = await User.findOne(1);
const timber = await User.findOne({ firstName: "Timber", lastName: "Saw" });

await timber.remove();
```

1.3.3.3 Express (<https://expressjs.com/es/>)

Express es una librería para crear un servidor web fácilmente y listo para producción.

```
export default function mountRoutes(app: express.Application, connection: Connection, mailer: Mail) {
  const apiRouter = express.Router();
  apiRouter.post('/authenticate', wrapAsync(authController.authenticate));

  apiRouter.post('/create-account', wrapAsync(authController.createAccount(mailer)));
  apiRouter.post('/confirm-account/:token', wrapAsync(authController.confirmAccount));

  apiRouter.post('/request-password-reset', wrapAsync(authController.requestPasswordReset(mailer)));
  apiRouter.post('/perform-password-reset/:token', wrapAsync(authController.performPasswordReset(mailer)));
}
```

Junto a Express he usado varios middlewares que sirven para extender la funcionalidad del servidor. Junto a algunos que mencionaré más tarde, están:

- “cookie-parser”: Este es un middleware que comprueba la cabecera Cookie de las peticiones HTTP y las hace accesibles a través del objeto de la petición.
- “cors”: Este es un middleware que nos permite usar CORS (Cross-origin resource sharing) que es un mecanismo de seguridad que permite/restringe el acceso a recursos dependiendo del dominio desde donde se hace la petición.

1.3.3.4 PostgreSQL (<https://www.postgresql.org/>)

Esta es la base de datos que he elegido para mi plataforma. Es completamente libre y gratuita, muy rápida, con funcionalidad comparable a bases de datos de pago como Oracle o MSSQL, fácil de usar, fácil de configurar y con una gran comunidad y soporte detrás.

Soporta una amplia gama de tipos que otras bases de datos no soportan, como puede ser el tipo JSON, JSONB (binario), direcciones IP, arrays, etc.

Es sensible a las mayúsculas y a las minúsculas, al principio fue a lo que más me costó adaptarme.

```
query:
SELECT period_date AS day, (
  SELECT COALESCE(SUM(
    EXTRACT(EPOCH FROM (
      COALESCE(
        session."endedAt",
        session."updatedAt"
      ) - session."createdAt"
    ) / 60
  ), 0)
FROM session
LEFT JOIN project_member
  ON project_member.id = session."projectMemberId"
LEFT JOIN project
  ON project.id = project_member."projectId"
WHERE DATE(session."createdAt") = period_date
AND project.id = $3
AND project_member.id = $4
) AS "minuteSum"
FROM (
  SELECT DATE(GENERATE_SERIES(
    $1,
    $2,
    interval '1 day'
  )) AS period_date
) AS period_date_series
ORDER BY period_date DESC;
```

1.3.3.4.1 Adminer (<https://www.adminer.org/>)

Para visualizar la base de datos y realizar pruebas sobre la base de datos no lo he hecho a través de la CLI, he preferido hacerlo a través de esta interfaz basada en PHP.

De forma similar a PhpMyAdmin, me permite visualizar toda la base de datos, realizar consultas, ver la estructura, crear nuevos registros, etc.

Idioma: Español | PostgreSQL » db » timeit_dev » public » Mostrar: project_member

Adminer 4.7.6 4.7.7

DB: timeit_dev | Esquema: public

Comando SQL | Importar | Exportar | Crear tabla

registros auth_token
registros mail_token
registros project
registros project_member
registros session
registros session_app_event
registros session_note
registros user

Mostrar: project_member

Visualizar contenido | Mostrar estructura | Modificar tabla | Nuevo Registro

Mostrar | Condición | Ordenar | Límite: 50 | Longitud de texto: 100 | Acción: Mostrar

	id	createdAt	role	status	projectId	userId
Modificar	1	2020-05-28 18:42:36.89893+00	admin	active	1	1
Modificar	2	2020-05-30 14:30:43.909759+00	admin	active	2	1
Modificar	7	2020-05-30 17:29:57.743714+00	employee	active	2	14
Modificar	8	2020-05-30 20:18:30.361774+00	admin	active	3	15
Modificar	11	2020-05-31 20:16:40.699571+00	admin	active	4	1
Modificar	12	2020-05-31 20:17:16.638327+00	admin	active	5	1
Modificar	9	2020-05-30 22:06:03.705026+00	employee	active	1	15
Modificar	21	2020-06-04 11:57:55.925843+00	admin	active	6	1
Modificar	20	2020-06-03 19:19:38.859067+00	employee	invited	1	14

Resultado completo | 9 registros | Modificar | Guardar | Modificar | Clonar | Eliminar | Exportar (9)

Importar

1.3.3.5 JSON Web Token (<https://github.com/auth0/node-jsonwebtoken>)

Para crear un token para las sesiones de usuario, he usado JSON Web Tokens. Este tipo de token se valida por sí solo, sin saber la clave con la que se cifraron.

Se puede guardar información dentro del JSON Web Token, esta información puede estar cifrada o no, lo importante es que es imposible modificarla ya que el checksum guardado dentro del token sería diferente al que se calcularía al calcularlo basado en la información guardada.

Soporta expiración automática, asignación a dueños, etc. He usado esta tecnología por probar y aprender algo nuevo, pero realmente para el uso de mi aplicación no me hubiese hecho falta.

Una gran virtud de estos tokens es que permiten tener sesiones sin tener que guardarlas en el servidor, llamado en la comunidad “stateless authentication” – autenticación sin estado. El usar este tipo de autenticación hace que muchas funcionalidades como saber que tokens tienes, poder revocar ciertos tokens, etcétera, sea imposible sin tener estado en el servidor.

Por lo que al final he tenido que usar un híbrido, un id interno del token se guarda en el servidor, y cuando un usuario intenta autenticarse, se valida el id del token que el usuario ha enviado con el token guardado en la base de datos.

Realmente, este sistema no se diferencia en mucho a las sesiones tradicionales.

1.3.3.6 Nodemailer (<https://nodemailer.com/>)

Esta librería nos permite enviar correos electrónicos desde JavaScript, solo le tenemos que dar ciertos detalles de nuestro proveedor / servidor de correo electrónico y podremos enviar correos electrónicos fácilmente.

La aplicación envía correos electrónicos al usuario para confirmar la cuenta, principalmente, tras registrarse, tras pedir un cambio de contraseña y al recibir una invitación a un proyecto.

1.3.3.7 Helmet (<https://github.com/helmetjs/helmet>)

Esta librería es un middleware de Express que nos permite hacer nuestra aplicación más segura al enforzar ciertas cabeceras en las peticiones de HTTP.

Hay algunas funcionalidades que vienen activadas por defecto y otras que hay que activarlas manualmente y que requieren más configuración. Yo he usado las que vienen por defecto, que son las siguientes.

Modulo	Acción
Dns Prefetch Control	Evita que el navegador resuelva las IPs del contenido antes de tiempo.
Frameguard	Evita que el usuario haga clic en iframes que se pueden poner encima de nuestra página web.

Hide Powered By	Oculto el hecho que usamos Express y su versión.
HSTS	Evita que la aplicación sea forzada a usar HTTP usando ataques de MITM.
IE No Open	Evita que Internet Explorer permita el ejecutar o abrir archivos sin descargarlos (descarga temporal).
No Sniff	Evita que otras aplicaciones alteren el tipo de archivo (mimetype).
XSS Filter	Evita ataques XSS, para que otras páginas web no ejecuten código en la nuestra.

1.3.3.8 Dotenv (<https://github.com/motdotla/dotenv>)

Esta librería nos permite crear un archivo .env que guarda variables que no queremos que se compartan entre entornos de desarrollo (desarrollo, pruebas, producción, etc.)

Además, de esta forma puedo guardar credenciales que solo tengo yo en mi máquina y ya que este archivo está ignorado por Git, no puedo cometer el fallo de hacer un commit con las contraseñas, de, por ejemplo, mi correo electrónico.

```
TIMEIT_EMAIL_HOST=smtp.yandex.com
TIMEIT_EMAIL_PORT=465
TIMEIT_EMAIL_USER=example@example.com
TIMEIT_EMAIL_PASS=somepass
TIMEIT_JWT_SECRET=somesecret
TIMEIT_CORS_ORIGIN=http://localhost:3000
TIMEIT_COOKIE_DOMAIN=localhost
TIMEIT_CRYPT_ROUNDS=10
TIMEIT_FRONTEND_URL=http://192.168.0.4:3000
```

El contenido de estos archivos se carga y es accesible en el objeto *process.env* de Node.js.

1.3.3.9 Bcrypt (<https://github.com/kelektiv/node.bcrypt.js>)

BCrypt es un algoritmo de hash diseñado para hacer que las contraseñas sean extremadamente seguras. Es la mejor valorada en la comunidad de criptografía.

```
const correctPassword = await bcrypt.compare(password, user.passwordHash);

export const hashPassword = (rawPassword: string): Promise<string> =>
  bcrypt.hash(rawPassword, parseInt(process.env.TIMEIT_CRYPT_ROUNDS));
```

1.3.3.10 Nanoid (<https://github.com/ai/nanoid>)

Para enviar los correos electrónicos, en la URL he tenido que incluir un token el cual luego se utiliza para validar y confirmar la acción del correo electrónico. Para eso he utilizado esta librería, que me permite generar IDs aleatorios que se pueden introducir en la URL sin escapar y que funciona más rápido que UUIDv4 siendo igual de seguro.

1.3.3.11 Morgan (<https://github.com/expressjs/morgan>)

Este middleware de Express nos permite ver información al recibir una petición HTTP. Tiene muchas opciones para configurar el formato del log y es muy útil, ya que por defecto Express no tiene un sistema de logs.

```

::ffff:192.168.0.4 - - [06/Jun/2020:14:28:46 +0000] "GET /api/data_query/history_statistics/1?memberId=1&startDate=K222020-06-01T14:28:45.828Z&endDate=K222020-06-06T14:28:45.828Z HTTP/1.1" 200 304 "http://localhost:3000/project/1" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.97 Safari/537.36"

```

1.2.3.12 Nodemon (<https://nodemon.io/>)

En el desarrollo de una aplicación, es muy útil tener una herramienta que automáticamente reinicie la aplicación cuando se detecte un cambio en el archivo.

Nodemon hace exactamente eso, ejecuta un archivo usando Node.js y cuando detecta un cambio simplemente reinicia la aplicación.

Esto junto al compilador de typescript en modo de “watch” hace que la experiencia al programar sea mucho mejor que si tuviésemos que estar reiniciando la aplicación manualmente cada vez que quisiésemos probar los cambios que hemos realizado.

1.3.4 Tecnologías específicas del cliente de escritorio

1.3.4.1 Qt

Qt es una framework para desarrollar aplicaciones para múltiples plataformas. Entre las plataformas que soporta están Windows, Linux, Mac OS, Android, sistemas embebidos, etc.

Para crear aplicaciones en Qt se puede usar C++, el lenguaje oficial de Qt. También se pueden usar bindings en otros lenguajes como Python, Go, etc.

Al usar Qt, usamos la abstracción que se hace sobre las APIs nativas de cada sistema operativo, por lo que no te tienes que preocupar de la plataforma a la que vas a compilar. Al compilar se genera un ejecutable para cada plataforma.

Para crear las vistas se usa un lenguaje llamado QML. Este lenguaje nos permite crear la estructura de la vista con controles incluidos con Qt.

QML tiene un motor de JavaScript integrado, por lo que la lógica de interfaz se puede programar sin tocar C++, simplemente con JS.

```

Button {
    id: startCounterButton
    text: qsml("Pulsar contador")
    font.pointSize: 20
    anchors.top: parent.top
    anchors.topMargin: 50
    anchors.bottom: parent.bottom
    anchors.bottomMargin: 20
    anchors.right: parent.right
    anchors.rightMargin: 45
    anchors.left: parent.left
    anchors.leftMargin: 45
    enabled: selectedProjectId != -1

    onClicked: {
        backend.js.createSession(selectedProjectId, function(res, err) {
            if (err) {
                const session = JSON.parse(res);
                watchView.push("Session qml", {
                    sessionId: session.id
                });
            } else {
                console.log('Unexpected error occurred', err);
            }
        });
    }

    contentItem: Text {
        text: parent.text
        font: parent.font
        color: Qt.rgba(0, 0, 0, 0.5)
        horizontalAlignment: Text.AlignCenter
        verticalAlignment: Text.AlignVcenter
        elide: Text.ElideRight
    }

    background: Rectangle {
        fill: background
        color: enabled ? styles.greenColor : styles.textColor
        radius: 10
    }

    states: [
        state {
            name: "Hovering"
            PropertyChanges {
                target: backgroundColor
                color: styles.lightGreenColor
            }
        },
        state {
            name: "Pressed"

```

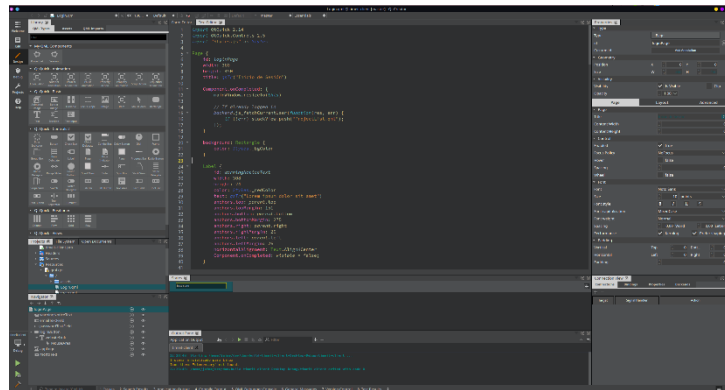
Es una framework muy utilizada hoy en día, algunas de las aplicaciones que uso y están desarrolladas con Qt son KDE Plasma, VirtualBox, TeamViewer, Audacity, entre otras.

He decidido usar Qt y C++ ya que la aplicación de escritorio tiene que poder estar cerca de las APIs nativas de cada sistema operativo, y de esa manera no tendría que estar jugando con funcionalidades para acceder las APIs de C/C++ desde otro lenguaje de más alto nivel, como podría ser JavaScript. Así podía usar un lenguaje para toda la aplicación de escritorio.

Me ha gustado mucho trabajar con Qt, pensaba que iba a ser más complicado, pero me he llevado una gran sorpresa. Es una muy buena plataforma para realizar aplicaciones de escritorio.

1.3.4.2 Qt Creator

Para crear la aplicación cliente usando Qt, he decidido usar el IDE oficial. La verdad es que es un IDE muy completo, pero he de decir que es un poco confuso como distribuir los archivos al principio.



1.4.4.3 QML

QML es el lenguaje de programación usado en Qt Quick.

Es un lenguaje un poco raro al principio, pero una vez empiezas a conocerlo descubres que es una herramienta muy potente y no tan difícil de usar.

Este lenguaje se usa para crear la interfaz de las aplicaciones, también se pueden controlar eventos de la interfaz de usuario y programar la lógica usando su propio motor de JavaScript integrado completamente en el lenguaje.

Ofrece interoperabilidad con el código C++, por lo que podemos programar el código para mejorar la interfaz en JS y cuando haya que realizar operaciones más costosas o que acceden a APIs de menor nivel podemos llamar a funciones de C++ desde QML.

```
font.pointSize: 14
selectByMouse: true
selectionColor: Styles._lightBlueColor
background: Rectangle {
    color: Styles._whiteColor
    border.width: parent.focus && 2
    border.color: parent.focus
        ? Styles._blueColor
        : Styles._whiteColor
    radius: 5
}

Keys.enabled: true
Keys.onReturnPressed: executeLoginAction();

function executeLoginAction() {
    backend.js_authenticateUser(emailTextField.text,
        passwordTextField.text,
        function(res, err) {
            if (err) {
                if (err === "INVALID_CREDENTIALS") {
                    warningNoticeText.text = "Tu correo o contraseña son incorrectos";
                } else if (err === "INACTIVE_ACCOUNT") {
                    warningNoticeText.text = "Tu cuenta todavía no ha sido confirmada";
                } else {
                    warningNoticeText.text = "Error desconocido: " + err;
                }
                warningNoticeText.visible = true;
            } else {
                warningNoticeText.visible = false;
                stackView.push("ProjectList.qml");
            }
        });
}

Button {
    label: loginButton
```


2. Análisis del funcionamiento de la aplicación

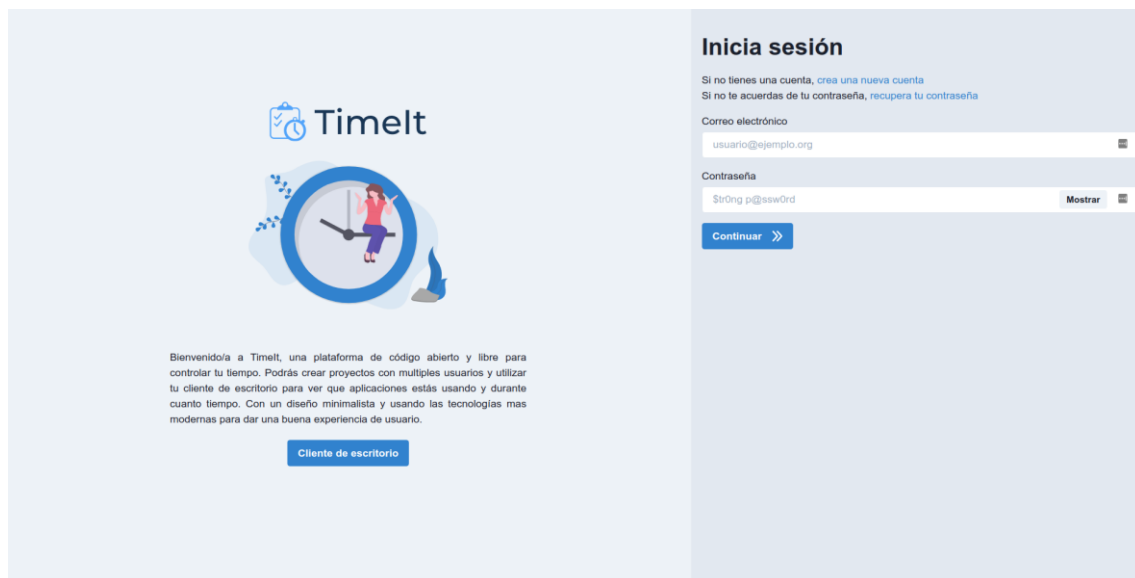
En este apartado analizaremos cómo funciona la plataforma desde el punto de vista de un usuario.

2.1 Plataforma web

La aplicación está diseñada para pantallas de ordenador, pero igualmente se adapta a pantallas más pequeñas, de, por ejemplo; un smartphone, mientras sea posible.

2.1.1 Página de inicio de sesión

Lo primero con lo que nos encontraremos al abrir la página web donde está nuestro “frontend” es la página de inicio de sesión (si es que no tenemos una sesión iniciada)



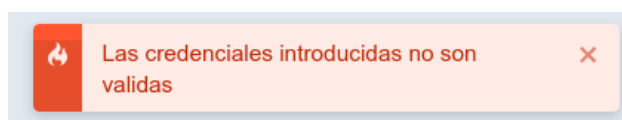
A la izquierda tendremos un logo junto con una descripción de nuestra aplicación, y un enlace para descargarnos el cliente de escritorio.

A la derecha tendremos el formulario para iniciar sesión, junto enlaces para crear una cuenta si no tenemos una cuenta además de un enlace para recuperar la contraseña si es que no nos acordamos.

En el campo de la contraseña tenemos un botón que muestra la contraseña en texto plano para confirmar que la contraseña que hemos introducido es correcta.

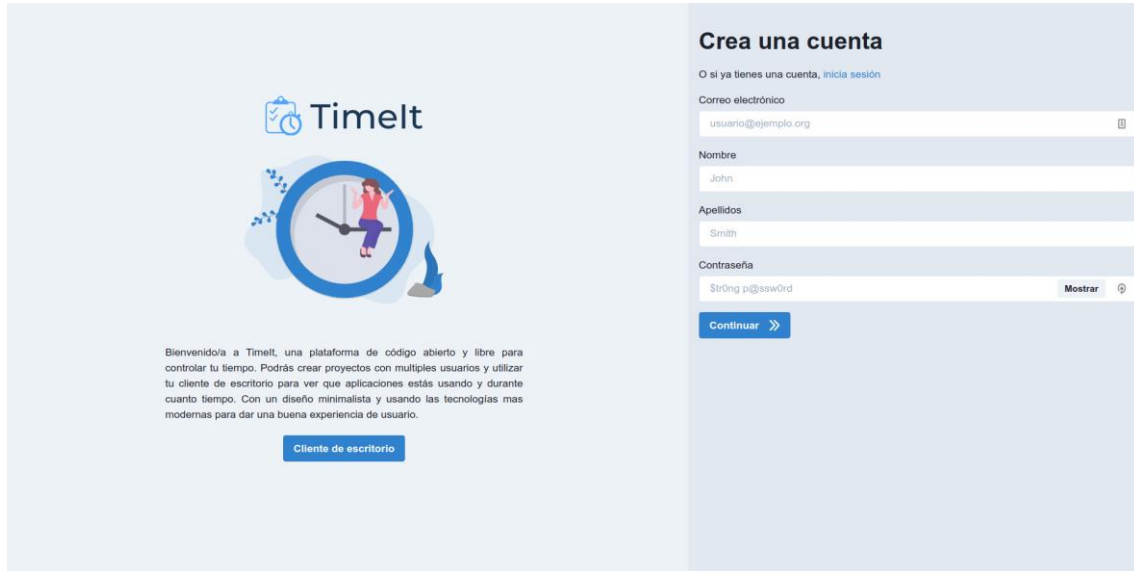
Los campos se validan, el correo tiene que ser válido y ningún campo puede estar vacío.

Al iniciar sesión correctamente iremos a la página de proyectos. Si las credenciales no son correctas recibiremos una notificación toast.



2.1.2 Página de registro

Esta página tiene la misma apariencia, lo único que cambia es el formulario y los enlaces a otras páginas.



Timelt

Bienvenido/a a Timelt, una plataforma de código abierto y libre para controlar tu tiempo. Podrás crear proyectos con múltiples usuarios y utilizar tu cliente de escritorio para ver que aplicaciones estás usando y durante cuanto tiempo. Con un diseño minimalista y usando las tecnologías más modernas para dar una buena experiencia de usuario.

[Cliente de escritorio](#)

Crea una cuenta

O si ya tienes una cuenta, [inicia sesión](#)

Correo electrónico
usuario@ejemplo.org

Nombre
John

Apellidos
Smith

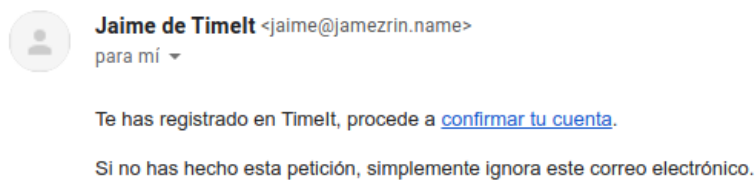
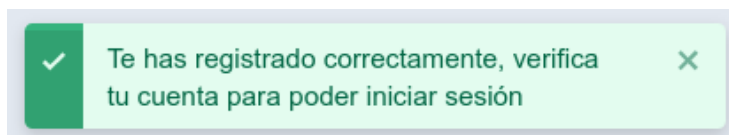
Contraseña
\$tr0ng p!@ssw0rd [Mostrar](#)

[Continuar >>](#)

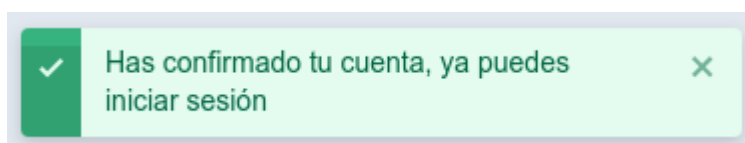
Los enlaces del formulario nos permiten volver a la página de inicio de sesión.

Todos los campos son validados y como la página de inicio, tenemos un botón para mostrar la contraseña.

Al crear la cuenta recibiremos una notificación toast la cual nos dirá que es necesario confirmar la cuenta. La aplicación nos envía un correo electrónico con un enlace para confirmar la cuenta.

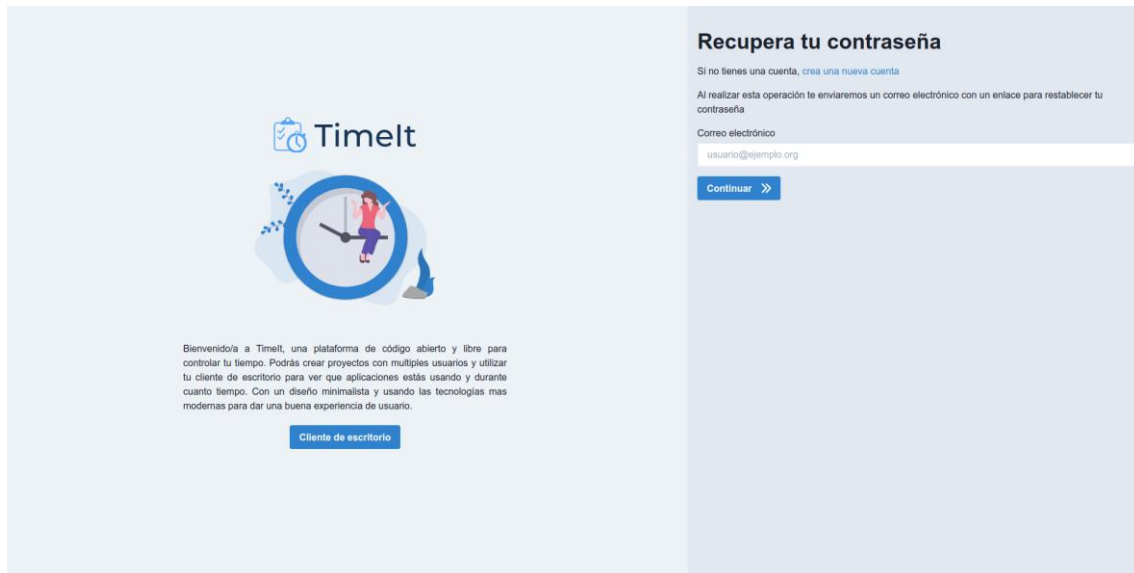


Tras hacer clic en ese enlace, iremos a la página de inicio de sesión recibiendo otra notificación toast confirmando que ya podemos iniciar sesión.



2.1.3 Página de recuperación de contraseña

Esta página es prácticamente lo mismo que la de inicio de sesión, pero nos permite enviarnos un correo electrónico con un enlace para restablecer nuestra contraseña en caso de que nos la hayamos olvidado.



Timelt

Bienvenido/a a Timelt, una plataforma de código abierto y libre para controlar tu tiempo. Podrás crear proyectos con múltiples usuarios y utilizar tu cliente de escritorio para ver que aplicaciones estás usando y durante cuanto tiempo. Con un diseño minimalista y usando las tecnologías mas modernas para dar una buena experiencia de usuario.

[Cliente de escritorio](#)

Recupera tu contraseña

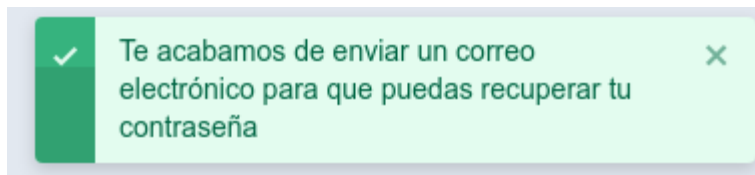
Si no tienes una cuenta, [crea una nueva cuenta](#)

Al realizar esta operación te enviaremos un correo electrónico con un enlace para restablecer tu contraseña

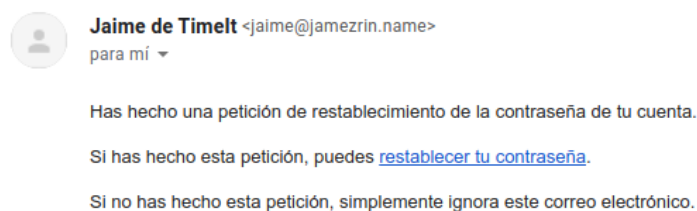
Correo electrónico

[Continuar >>](#)

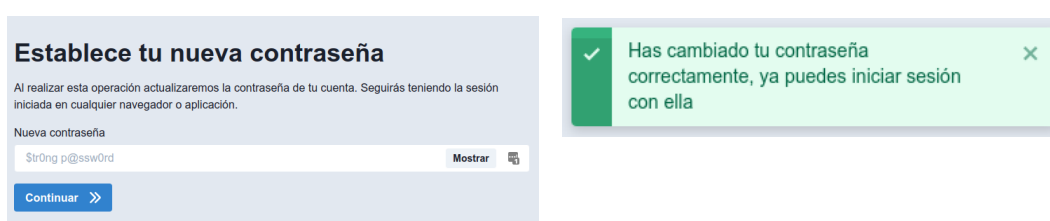
Al pedir el restablecimiento de contraseña, recibiremos otra notificación toast indicándonos que se ha realizado la solicitud correctamente.



Y como podemos esperar, recibiremos el correo electrónico con un enlace para restablecer la contraseña.



Tras hacer clic en ese enlace, iremos a una página similar a la anteriores, con un formulario para introducir la nueva contraseña. Al completar este formulario la contraseña cambiará a la que hemos elegido y recibiremos una notificación toast.



Establece tu nueva contraseña

Al realizar esta operación actualizaremos la contraseña de tu cuenta. Seguirás teniendo la sesión iniciada en cualquier navegador o aplicación.

Nueva contraseña

[Mostrar](#)

[Continuar >>](#)

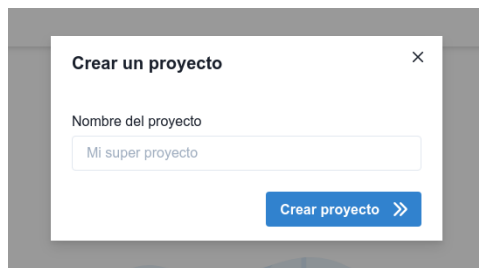
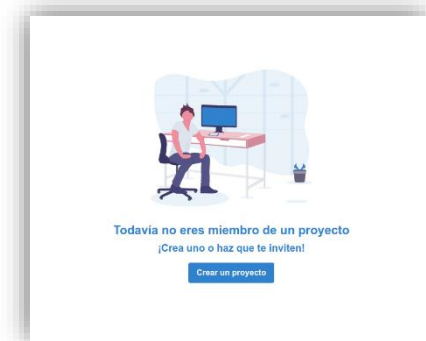
Has cambiado tu contraseña correctamente, ya puedes iniciar sesión con ella

2.1.4 Página de lista de proyectos

Tras iniciar sesión, esta es la página con la que nos encontraremos. Aquí podremos encontrar una lista de los proyectos de los que formamos parte. Al hacer clic en uno de los proyectos iremos a la página de proyecto.

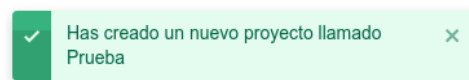



Si no formamos parte de ningún proyecto, veremos esta llamada a acción en lugar de la lista que hemos visto antes.




Si hacemos clic en el botón para crear un proyecto veremos este modal donde nos pedirá el nombre.

Al crearlo recibiremos una notificación toast confirmando la acción.



En la barra de navegación de la página nos encontraremos con el botón  el cual nos permitirá cambiar entre el tema oscuro y el tema claro.

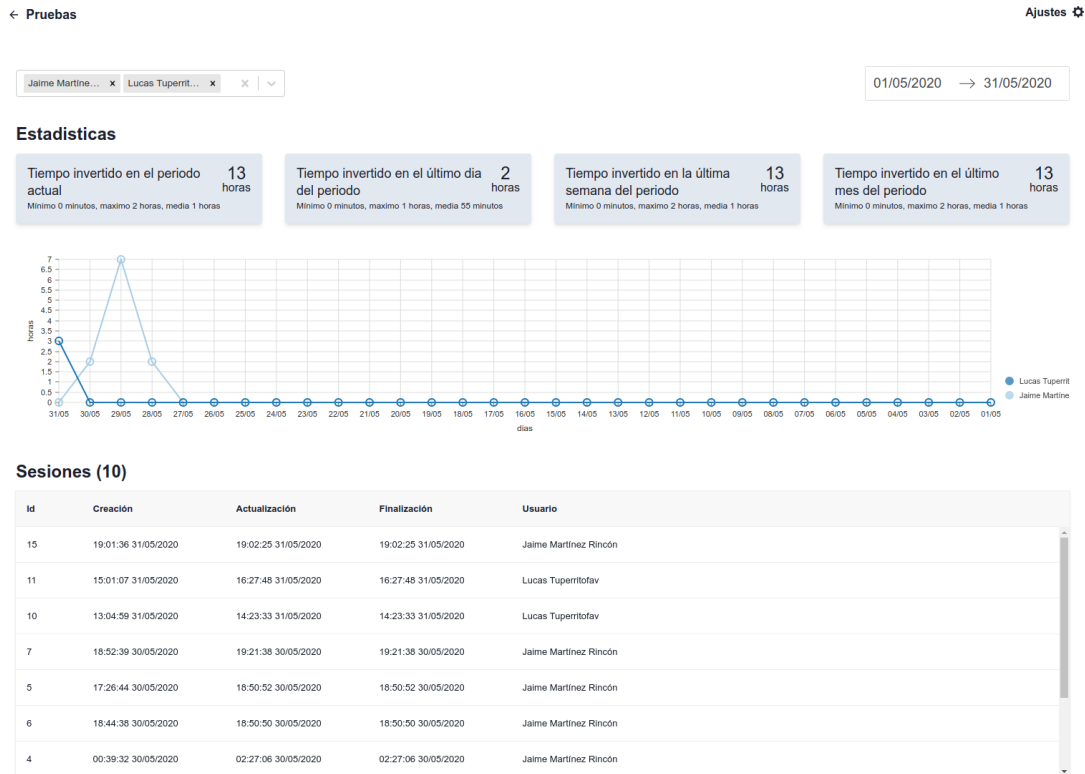


Y también nos encontraremos con el botón  el cual nos permitirá cerrar nuestra sesión.

2.1.5 Página de información de proyecto

En esta página podremos ver información de cada proyecto.

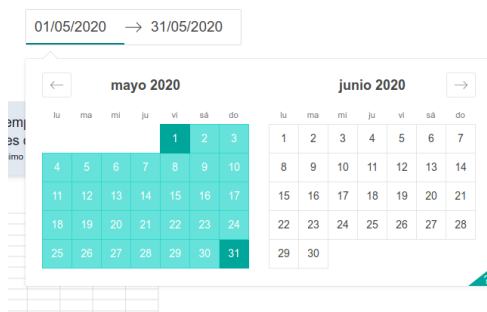
Como se ve, podemos ver estadísticas sobre las horas invertidas por los miembros seleccionados resumidas, en suma, media, mínimo y máximo además de un gráfico de líneas con la suma de horas por miembro y por día.



Podemos seleccionar los miembros de los que queramos ver la información.

Jaime Martine... x Lucas Tuperrit... x x v

jaime martinez



También podemos seleccionar el rango o periodo de fechas.

Si somos manager o admin, podremos ver y acceder a la página de configuración de proyecto, donde podremos ver los miembros y realizar acciones sobre ellos. También podremos hacer otras acciones sobre el proyecto.

2.1.6 Página de sesión

Si hacemos clic en una fila de la tabla de sesiones en anterior página, veremos esta página. En ella podremos ver otra tabla con todas las actividades que han ocurrido en la sesión.

← Pruebas

Lista de eventos

Tipo de evento	Fecha de creación	Ocurrencias	Contenido
App	01:14:32 30/05/2020	1	node.js - NodeMailer - Transport option must be a transport instance or configuration object - Stack Overflow - Google Chrome (google-chrome 5206)
App	01:14:27 30/05/2020	9	datetime - PostgreSQL data() with timezone - Stack Overflow - Google Chrome (google-chrome 5206)
App	01:11:17 30/05/2020	5	What's the best approach to confirm user email address: sending an email confirmation link or sending a verification code in email? - User Experience Stack Exchange - Google Chrome (google-chrome 5206)
App	01:10:52 30/05/2020	1	https://www.google.com/search?q=email+confirmation+flow&log=email+confirmation+flow&ag=chrome.6957.5215934&sourceid=chrome&ie=UTF-8 - Google Chrome (google-chrome 5206)
App	01:10:47 30/05/2020	1	ruby on rails - Devise - create user account with confirmed without sending out an email? - Stack Overflow - Google Chrome (google-chrome 5206)
App	01:10:42 30/05/2020	1	https://stackoverflow.com/questions/7465467/devise-create-user-account-with-confirmed-without-sending-out-an-email - Google Chrome (google-chrome 5206)
App	01:10:37 30/05/2020	4	create user on register or on confirmation email - Buscar con Google - Google Chrome (google-chrome 5206)
App	01:10:17 30/05/2020	1	create user on register or on confirmation - Buscar con Google - Google Chrome (google-chrome 5206)
App	01:10:12 30/05/2020	1	create user on register? - Buscar con Google - Google Chrome (google-chrome 5206)
App	01:08:47 30/05/2020	9	postgres - postgres default timezone - Stack Overflow - Google Chrome (google-chrome 5206)
App	01:08:42 30/05/2020	1	nodemailer/nodemailer: (x) Send e-mails with Node.js - easy as cake! - Google Chrome (google-chrome 5206)
App	01:08:37 30/05/2020	1	RG8Boys/express-mailer: Send Emails from your application and response object. - Google Chrome (google-chrome 5206)
App	01:08:32 30/05/2020	4	express mailer - Buscar con Google - Google Chrome (google-chrome 5206)
App	01:08:22 30/05/2020	2	javascript - Sending email to multiple recipients via express-mailer issue - Stack Overflow - Google Chrome (google-chrome 5206)
App	01:08:12 30/05/2020	1	javascript - How to send email in html form using express.js - Stack Overflow - Google Chrome (google-chrome 5206)
App	01:07:52 30/05/2020	3	node.js - sent an email with express.js - Stack Overflow - Google Chrome (google-chrome 5206)
App	01:07:32 30/05/2020	4	express-mailer - npm - Google Chrome (google-chrome 5206)
App	01:06:22 30/05/2020	6	mailer.js - timeit-webapp - Visual Studio Code (code 592983)

Las ocurrencias son las veces que el cliente de escritorio ha detectado que hemos estado en esa ventana. En esta ventana también se ven notas, que son simplemente eso, notas, las cuales se crean desde el cliente de escritorio.

Esta tabla se ordena por fecha de creación de forma descendente, de forma que veremos los eventos más recientes arriba.

El contenido del evento depende de si es una nota o un evento de aplicación. Si es una nota el contenido del evento es simplemente el texto que ha escrito el usuario. Si es un evento de aplicación, veremos el título de la ventana junto al nombre del proceso y su id entre paréntesis.

Para que un evento se una e incremente el número de ocurrencias, debe tener el mismo título de ventana, el mismo nombre de proceso y el mismo id.

2.1.7 Página de configuración de proyecto

Esta página, la cual es solo accesible por miembros privilegiados (admin o manager) nos permite realizar ciertas acciones sobre el proyecto y sus miembros.

← Pruebas

Miembros del proyecto

Id	Nombre	Apellidos	Rol	Estado	Fecha entrada	Correo	Acciones
20	jaime	martinez	employee	invited	03/06/2020	mrjaime1990@gmail.com	▲ Ⓢ
1	Jaime	Martínez Rin...	admin	active	28/05/2020	jaime@jamezrin.name	▼ Ⓢ
9	Lucas	Tuperritofav	employee	active	31/05/2020	dasb7db128312v31283@gmail.com	▲ Ⓢ

Cambiar nombre de proyecto

Aquí puedes cambiar el nombre del proyecto.

Nombre de proyecto **Cambiar nombre**

Invitar a usuario

Invita a un usuario para monitorizar su trabajo. Podrás cambiar su rol en la lista de miembros. Aparecerá como un miembro cuando acepte la invitación.

Correo electrónico **Invitar usuario**

Borrar proyecto

Confirma que quieres borrar este proyecto escribiendo su nombre debajo. Una vez borres el proyecto, se borrarán todas las sesiones, eventos de aplicación y notas relacionados permanentemente.

Nombre de proyecto **Borrar proyecto**

Como vemos, tenemos una lista de los miembros que están dentro del proyecto, junto con botones para subir de rol (employee -> manager -> admin) y otro para expulsar.

Solo el administrador puede cambiar el rol de otro usuario. Para expulsar a un miembro el miembro que realiza la acción tiene que estar un nivel por encima en la jerarquía de permisos, si no, no podrá realizar esta acción.

Si hacemos clic en el botón para expulsar, recibiremos un dialogo de confirmación. Si aceptamos, se borrará toda la información sobre ese miembro de proyecto.

192.168.0.4:3000 says

¿Está seguro de que quiere expulsar a este miembro? Se borrarán todos los eventos e información almacenada.

Cancel **OK**

También disponemos de un formulario para cambiar el nombre del proyecto, otro para invitar a un usuario y otro para borrar el proyecto.

Los campos de estos formularios también se validan para que no estén vacíos. En caso del formulario para borrar el proyecto hace falta escribir el nombre del proyecto para que se habilite el botón y podamos borrar el proyecto.

Al invitar a un usuario, este recibirá un correo electrónico con la invitación. Al hacer clic, ya podrá ver el proyecto y usarlo.

Jaime de Timelt
para mí ▼

Te han invitado al proyecto Pruebas, puedes [aceptar la invitación](#).

Si no quieres aceptar esta invitación, simplemente ignora este correo electrónico

2.2 Aplicación de escritorio

La aplicación de escritorio cuenta con la funcionalidad básica para usar nuestra cuenta creada en la plataforma y enviar los eventos de aplicación y notas.

La aplicación es compatible con Windows y Linux (solo si usa X11).

2.2.1 Vista de inicio de sesión

Es la primera vista con la que nos encontraremos nada más iniciar la aplicación (si no hemos iniciado sesión antes).



Contaremos con el formulario para iniciar sesión, además de un enlace para ir a la página web.

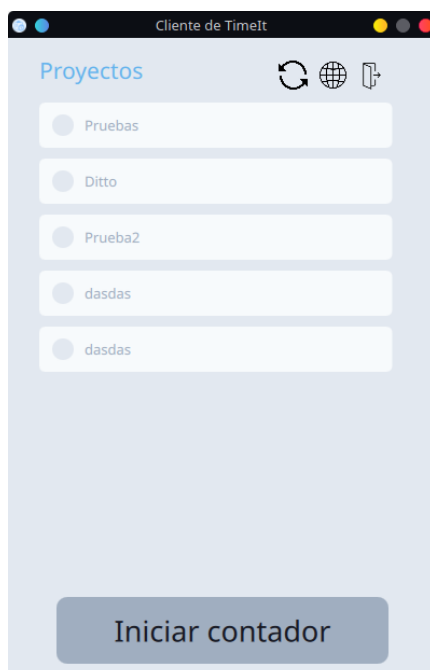
No hay forma de registrarse desde el cliente de escritorio, esa acción debe realizarse desde la página web.

Si las credenciales no son correctas, recibiremos un mensaje de alerta encima del formulario indicándonos que el correo o la contraseña no son correctos.

Tu correo o contraseña son incorrectos

Tras iniciar sesión iremos a la vista de lista de proyectos, la cual veremos a continuación.

2.2.2 Vista de lista de proyectos



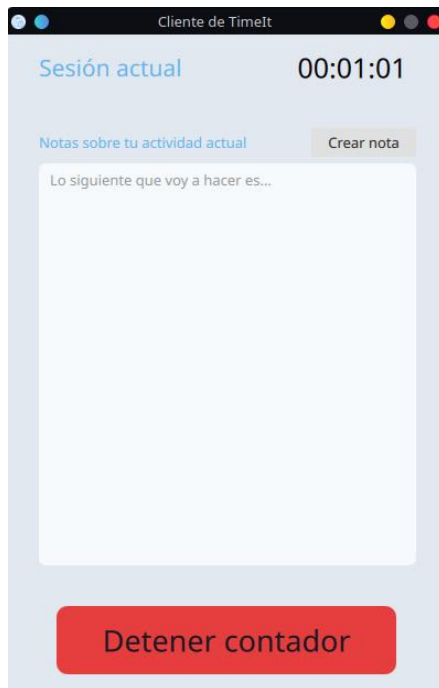
Esta vista nos la encontraremos tras haber iniciado sesión, veremos una lista de los proyectos de los que somos miembros. Podemos cerrar sesión, ir a la página web o recargar la lista.

Al seleccionar un proyecto de la lista el botón de iniciar contador se habilitará.

Iniciar contador

Al hacer clic en este botón iremos a la vista de sesión.

2.2.3 Vista de sesión



En esta vista podremos ver un contador del tiempo desde que iniciamos la sesión. También veremos una caja de texto para crear una nota y enviarla a la plataforma web.

Mientras que tenemos una sesión de trabajo iniciada, se enviarán los eventos de aplicación automáticamente, de forma que podemos iniciar esta aplicación y dejarla minimizada mientras trabajamos y en la plataforma web podremos ver las actividades que hemos realizado.

El botón de detener contador finalizará la sesión y nos llevará de nuevo a la vista de lista de proyectos.

3. Pruebas

Prueba 1

Elegir un rango de fechas o/y seleccionar los miembros y después ir a otra página (sesión, ajustes, etc.) y volver atrás.

Expectación: El rango de fechas y los miembros seleccionados son los mismos.

Resultado: Se reestablecen ambos el rango de fechas y los miembros seleccionados a los valores por defecto.

Solución: Guardar el estado en fuera del componente de la página, guardarlo en la URL o persistirlo en sessionStorage.

Prueba 2

Perder la conexión a Internet o el servidor se cae mientras tenemos una sesión iniciada.

Expectación: Nos notifica de que no tenemos conexión.

Resultado: No ocurre nada, simplemente se tiran errores por stderr.

Solución: Detectar esta situación y notificar al usuario de alguna manera. Guardando los eventos de alguna forma para que cuando volvamos a tener conexión se envíen. Si paramos la sesión se debería parar también.

Prueba 3

Cambiar el tema de la aplicación cambia el color de todos los componentes.

Expectación: Todos los componentes (selector de miembros, selector de fechas, tablas, grafico de líneas, etc.) cambian de estilo adaptándose al modo oscuro para que el color sea uniforme en toda la aplicación.

Resultado: Los componentes de selector de fechas, las tablas y el grafico de líneas mantienen el tema claro y no se ve bien en contraste con el resto de la aplicación.

Solución: Cambiar el estilo de los componentes de las respectivas librerías.

Prueba 4

Perder la conexión a la base de datos una vez el servidor ya está iniciado.

Expectación: Cuando un usuario pide un recurso de la API, se devuelve un error interno del servidor indicando que no hay conexión a la base de datos.

Resultado: Se devuelve un error genérico con el cuerpo indicando que es un error "ECONNREFUSED" encapsulado en un error interno del servidor (código 500).

Solución: Crear un middleware de express que comprueba que tenemos acceso a la base de datos antes de proceder con otras peticiones.

Prueba 5

Cerrar el cliente de escritorio mientras tenemos una sesión de trabajo iniciada.

Expectación: La sesión se considera terminada tras no recibir actualizaciones, no sigue el contador de tiempo.

Resultado: La sesión se mantiene abierta, hasta que el usuario que la inició abra una nueva sesión. El tiempo no sigue corriendo, por lo que tampoco es un gran problema.

Solución: Pasado un cierto tiempo sin recibir una actualización, se debería actualizar la sesión estableciendo la fecha de fin a la fecha de última actualización.

Prueba 6

Seleccionar un rango de fechas muy extenso (por ejemplo, de 10 años).

Expectación: Nos avisa de que no podemos seleccionar un periodo tan largo.

Resultado: Realiza una petición a la base de datos con ese periodo e intenta renderizar el gráfico de líneas con más de 3000 puntos de datos en el eje X, el navegador se ralentiza usando prácticamente toda la CPU durante al menos 10 segundos.

Solución: Comprobar el periodo seleccionado y si es demasiado largo, avisar al usuario con un mensaje y no seguir con la actualización de las estadísticas. Alternativamente se podría seleccionar un número de días de diferencia máximo para que ni siquiera pudiese seleccionar tal periodo.

Prueba 7

Cambiar el nombre de proyecto a espacios en blanco.

Expectación: Nos avisa que tenemos que elegir un nombre con caracteres alfanuméricos, no solo espacios u otros caracteres no visibles.

Resultado: Se establece un nombre de proyecto de solo espacios.

Solución: Usar una expresión regular para asegurarnos que introduce algún carácter alfanumérico.

Prueba 8

Registrarnos (o reestablecer la contraseña) usando una contraseña corta.

Expectación: No nos deja y nos avisa que tenemos que elegir una contraseña con un mínimo de caracteres para incrementar la seguridad.

Resultado: Nos deja continuar poniéndonos contraseñas demasiado cortas, de, por ejemplo, 1 carácter.

Solución: Establecer un mínimo de caracteres para la contraseña al registrarnos y al pedir una nueva contraseña.

Prueba 9

Registrarnos (o reestablecer la contraseña) usando solo espacios o caracteres raros.

Expectación: No nos deja y nos avisa que tenemos que elegir una contraseña con caracteres apropiados.

Resultado: Nos deja registrarnos con contraseñas de solo espacios, por ejemplo.

Solución: Usar una expresión regular para asegurarnos que los caracteres usados en la contraseña son correctos.

Prueba 10

Enviar formularios al servidor usando peticiones HTTP desde herramientas como Postman (sin pasar por la validación del frontend).

Expectación: Nos devuelve un error, usando las mismas técnicas de validación que el frontend.

Resultado: Nos permite, por ejemplo, registrarnos sin establecer un nombre, o con un correo electrónico vacío, etc.

Solución: Externalizar los schema de validación del frontend y usarlos en el backend para validar las peticiones recibidas.

Prueba 11

Intentar acceder a un proyecto que no existe o del que no somos miembros cambiando el id dentro de la URL. También aplica para otros recursos como sesiones, ajustes de proyecto, etc.

Expectación: Nos alerta que no existe ese proyecto y nos devuelve a la página anterior.

Resultado: Se queda una pantalla de carga que nunca terminará.

Solución: Comprobar que la respuesta del servidor sobre el proyecto que estamos intentando acceder es y si no es válida, mostrar una notificación toast y devolvernos a la página anterior.

Prueba 12

En cualquier ruta del servidor donde se pide un id como parámetro por la URL, intentar usar un número superior a un número entero de 32 bits.

Expectación: Funciona correctamente o nos devuelve un error alertándonos que el parámetro no es válido.

Resultado: En error directo de la base de datos donde indica que no es un numero valido.

Solución: Envolver al error devuelto por la base de datos o intentar soportar ids superiores a enteros de 32 bits (por ejemplo, tipos long).

4. Conclusiones

En mi opinión, creo que este proyecto es el mayor proyecto que he realizado yo solo hasta la fecha. Al principio no sabía con qué idea empezar el desarrollo del proyecto, pero me decanté por esta ya que tenía bastante claros los requisitos de la aplicación y lo que yo quería que hiciese la aplicación.

Gracias a los objetivos que me planteé, he aprendido muchas tecnologías las cuales creo que me ayudarán mucho en mi futuro y he puesto en práctica mis conocimientos de programación y de diseño.

4.1 Grado de logro de objetivos iniciales

Pienso que he logrado todos los objetivos iniciales que me planteé, e incluso he conseguido realizar objetivos que pensé para un futuro, si seguía desarrollando la aplicación.

Durante el desarrollo de la aplicación he descubierto funcionalidades extras y posibles mejoras a la implementación actual, pero igualmente, creo que los requisitos iniciales de la aplicación se han cumplido con creces.

Esta fase del desarrollo del proyecto se ha considerado como finalizado al finalizar todas las tareas con prioridad alta, media o baja listadas en el tablero Kanban donde distribuía las tareas a realizar.

He de decir que han quedado ciertas tareas por realizar, pero son mejoras al código actual de la aplicación y ni forman parte del plan inicial del proyecto ni afectan de una gran forma al resultado final del proyecto.

Hay una gran parte de las tareas restantes que son arreglos de las pruebas que he enumerado anteriormente.

4.2 Futuras líneas de investigación

Este proyecto deja muchas posibilidades para seguir desarrollando, al inicio del proyecto me planteé futuras funcionalidades que estaría bien incluir en una segunda etapa de desarrollo.

Entre esas posibles funcionalidades puedo mencionar las siguientes

- Hacer capturas de pantalla automáticamente desde el cliente de escritorio y hacerlas visibles desde la plataforma web.
- Detectar el tiempo inactivo o AFK (away from the keyboard) mediante la detección de movimientos del ratón o/y del teclado
- Soporte para un cliente de escritorio funcional desde dispositivos móviles

A lo largo de este periodo de desarrollo del proyecto, también he descubierto una gran lista de posibles funcionalidades, entre ellas están las siguientes

- Creación de sesiones manualmente (por si, por ejemplo, en un momento dado el usuario se olvida de iniciar el cliente)
- Desplegar la aplicación web de forma que esté disponible desde su propio dominio

- Implementación de publicidad usando Carbon Ads o Google Adwords
- Implementación de un sistema de facturación automática dependiendo de las horas registradas
- Implementación de un sistema de generación de reportes mensuales o semanales en formato PDF conteniendo un resumen de las actividades realizadas
- Implementación básica de un cliente web mediante una extensión de navegador
- Implementación de borrado “soft”, para no borrar permanentemente registros
- Implementación de un sistema de “garbage collection” de tokens de correo electrónico o de sesión expirados
- Implementación de una página para cambiar los detalles personales de cada usuario registrado en la plataforma
- Implementación de un sistema para subir avatares o fotos de perfiles para cada usuario, los cuales serían mostrados junto a los miembros
- Implementación de un sistema de localización para hacer disponible el proyecto en otros idiomas, como el inglés, por ejemplo
- Implementación de las sesiones de usuarios mediante una base de datos NoSQL como puede ser Redis, para agilizar los inicios y comprobaciones de sesiones
- Implementación de un sistema de mensajería básico entre los diferentes miembros del proyecto
- Implementación de un sistema para iniciar sesión con cuentas de Google, Twitter, Github, etc.
- Implementación de una “landing page” para descargar el cliente de escritorio para las diferentes plataformas compatibles, ver información sobre la aplicación, incluyendo los posibles casos de uso, capturas de pantalla de la aplicación, desglose de funcionalidades, etc.

Y si tenemos en cuenta los arreglos de las pruebas realizadas y otros puntos que “estarían bien incluirlos” podríamos incluir los siguientes

- Añadir typescript al frontend
- Reimplementar la autenticación usando sesiones en vez de JWT
- Implementar un sistema para olvidar la sesión al cerrar el navegador
- Añadir paginación a los resultados de la API
- Rate limiting de la API, para no sobrecargar el servicio
- Escapar nombres de proyecto y otros detalles en los mailers
- Usar hbs o mustache u otro sistema de plantillas para los mailers
- Desplegar a una plataforma como puede ser AWS, Netlify o Vercel
- Implementar el tema oscuro de la aplicación al 100%
- Revisar la implementación de ciertos componentes, en especial ProjectCreationModal
- Reemplazar el dialogo de confirmación al expulsar un miembro por un componente más bonito
- Revisar la implementación de los controles para acciones en las tablas
- Añadir un botón para actualizar los proyectos y la lista de sesiones
- Validación de campos en el backend al 100%
- Persistencia de los filtros de sesiones en la página de proyecto
- Prevención de elección de rangos de fechas extremadamente largos
- La implementación de otros arreglos encontrados en las pruebas

4.3 Dificultades resueltas más destacables

El desarrollo de este proyecto no se ha realizado de una forma perfecta, ha habido varios momentos en el desarrollo donde he tenido que parar e investigar durante un tiempo antes de poder continuar.

Estos lapsos en los que he tenido que investigar y realizar pruebas de conceptos, han contribuido a momentos en los que no estaba seguro de que decisión tomar, causando periodos sin ningún avance perceptible.

Dicho esto, puedo mencionar las siguientes situaciones las cuales han sido considerablemente más difíciles de entender o llevar a cabo que las demás

Uso de React Hooks

Siempre he seguido de cerca el desarrollo de React, de vez en cuando he hecho pruebas para tener una idea general actualizada pero nunca había desarrollado una aplicación completa. Al desarrollar una aplicación en React, se puede hacer usando componentes de clases o componentes con funciones. Yo elegí la segunda, los componentes con funciones.

Usar componentes con funciones introduce el paradigma de programación funcional el cual puede costar bastante entender y ser proficiente al principio.

Viendo el vídeo original de ReactConf sobre la introducción a los Hooks y leyendo el tutorial oficial de React, aprendí como funcionaban hasta un punto en el que entiendo bien usar esta funcionalidad.

Diseño de la API REST

Al diseñar la API del backend, me topé con un montón de dudas sobre como diseñar esta API. Ayudándome de StackOverflow y de blogs de ingeniería de compañías como Microsoft, decidí el formato para mi API.

Código Multiplataforma en C++

Nunca había desarrollado código multiplataforma en C++ antes, y tuve que aprender como programar código OOP desde cero. Hay muchos conceptos en C++ que no existen en otros lenguajes de más alto nivel como Java, JavaScript, Go, etc. Al principio me costó mucho crear una estructura para separar el código para Linux y para Windows de una forma apropiada.

Autenticación

Nunca había diseñado una aplicación web con múltiples usuarios mediante un sistema de inicio de sesión.

Al principio tuve muchas dudas sobre como diseñar este sistema y tras probar muchas cosas me decanté por JWT (JSON Web Tokens).

También tuve bastantes problemas con como almacenar este token en el navegador del usuario, ya que nunca había trabajado con cookies personalmente, siempre había utilizado alguna framework que realizaba esto por mí.

Tras haber diseñado este sistema, he llegado a la conclusión que posiblemente no fue la mejor decisión desde el punto de vista técnico, aunque tampoco empeora o afecta gravemente a la aplicación.

Se podría decir que simplemente usé algo que no está explícitamente diseñado para nuestro caso de uso, pero que tras adaptarlo es como cualquier otra solución, completamente valida.

Pienso que haber realizado la autenticación de esta forma me ha dotado de conocimientos que no hubiese aprendido si hubiese realizado la autenticación de otra forma más sencilla.

Permisos

Ya que la aplicación tiene diferentes roles de miembros de proyecto, hay ciertos recursos o acciones que son solo accesibles para ciertos miembros, al principio no tenía claro cómo crear consultas que tuviesen estos factores en cuenta, pero tras varias iteraciones y varias revisiones de código, he conseguido un resultado bastante bueno.

Diseño de la interfaz

Al iniciar el proyecto, me propuse utilizar una librería llamada TailwindCSS, la cual te da clases de CSS que puedes usar para construir componentes desde cero de una forma rápida y sencilla.

Lo que no pensé es como se iba a integrar con React, y el nivel de limpieza de código o dificultad para mantener este código.

Al empezar a diseñar la aplicación con esta librería, me di cuenta de que no iba a ser eficiente diseñando la interfaz de la aplicación y que no iba a resultar en un buen trabajo, por lo que empecé a buscar alternativas específicas al mundo de React.

Tras una larga investigación, viendo alternativas como React Bootstrap, Material UI, Antd, Grommet, etc; me decidí por Chakra UI, una librería parecida a Bootstrap con componentes bastante bien diseñados por defecto y con un gran número de opciones de configuración.

Creo que el usar esta librería ha hecho que el desarrollo de la aplicación haya sido múltiples veces más rápido que si tuviese que haber hecho yo todos los componentes desde cero.

Consultas SQL de estadísticas

Al usar TypeORM, la librería te da una capa de abstracción sobre la base de datos para simplificar las consultas y no tener que escribir consultas SQL a mano. Estas técnicas se usan en prácticamente todas las partes de la aplicación.

Pero al realizar las consultas SQL para el gráfico de líneas y los diferentes indicadores de estadísticas, tuve que recurrir a utilizar formas para realizar consultas a la base de datos diferentes a las que había realizado para el resto de la aplicación. Esto es porque estas consultas requieren de funciones y sintaxis que no está disponible a través de TypeORM.

Para ponernos en contexto, estas dos formas son las que generalmente se usan en la aplicación para consultar datos.

1. Uso del patrón Active Record para realizar una consulta simple.

```
const mailToken = await MailToken.findOne({
  where: {
    id: token,
    type: MailRequestType.ACCOUNT_CONFIRMATION,
  },
});
```

2. Uso del constructor de consultas (QueryBuilder) para consultas más complejas.

```
// Current user as a member of the project that has this session
const currentProjectMember = await ProjectMember.createQueryBuilder( 'alias: 'projectMember' )
  .where( where: 'projectMember.user = :currentUserId', parameters: { currentUserId } )
  .leftJoin( property: 'projectMember.project', alias: 'project' )
  .leftJoin( property: 'project.members', alias: 'otherMembers' )
  .leftJoin( property: 'otherMembers.sessions', alias: 'otherMemberSessions' )
  .andWhere( where: 'otherMemberSessions.id = :sessionId', parameters: { sessionId } )
  .getOne();

if (!currentProjectMember) {
  return resourceNotFoundError(req, res);
}

// prettier-ignore
const sessionQueryBuilder = Session.createQueryBuilder( 'alias: 'session' )
  .where( where: 'session.id = :sessionId', parameters: { sessionId } )
  .loadRelationCountAndMap( mapToProperty: 'session.appEventCount', relationName: 'session.sessionAppEvents' )
  .loadRelationCountAndMap( mapToProperty: 'session.noteCount', relationName: 'session.sessionNotes' )
  .loadRelationIdAndMap( mapToProperty: 'session.projectMemberId', relationName: 'session.projectMember' );

// Ensure the session being looked up is owned by the current member
if (!isMemberPrivileged(currentProjectMember)) {
  sessionQueryBuilder.andWhere( where: 'session.projectMember = :currentProjectMemberId', parameters: {
    currentProjectMemberId: currentProjectMember.id,
  } );
}

const session = await sessionQueryBuilder.getOne();

if (!session) {
  return resourceNotFoundError(req, res);
}
```

Y en el caso de las consultas para las estadísticas, estamos hablando de realizar consultas mucho más complejas y largas, como se puede ver.

Al no tener mucha experiencia con PostgreSQL, tuve que probar mucho y leer sobre conceptos como CTEs, series, casteo de tipos, extracción de fechas, etc.

```
const allEvents = await conn.query(
  query: `
    SELECT id, type, "createdAt", data FROM (
      SELECT MAX(id) as id,
        'app_event' AS type,
        MAX("createdAt") AS "createdAt",
        "sessionId",
        json_build_object(
          'windowName', "windowName",
          'windowClass', "windowClass",
          'windowPid', "windowPid",
          'eventCount', COUNT(id),
          'firstEventId', MIN(id),
          'firstEventTime', MIN("createdAt")
        ) AS data
      FROM session_app_event
      GROUP BY "windowName", "windowClass", "windowPid", "sessionId"
    ) UNION ALL
    SELECT id,
      'note' AS type,
      "createdAt",
      "sessionId",
      json_build_object(
        'text', "noteText"
      ) AS data
    FROM session_note
  ) AS session_update
  WHERE "sessionId" = $1
  ORDER BY "createdAt" DESC
  ,
  parameters: [sessionId],
);
```

```
const allHistory = await conn.query(
  query: `
    SELECT period_date AS day, (
      SELECT COALESCE(SUM(
        EXTRACT(EPOCH FROM (
          COALESCE(
            session."endedAt",
            session."updatedAt"
          ) - session."createdAt"
        )) / 60
      ), 0)
    FROM session
    LEFT JOIN project_member
      ON project_member.id = session."projectMemberId"
    LEFT JOIN project
      ON project.id = project_member."projectId"
    WHERE DATE(session."createdAt") = period_date
      AND project.id = $3
      AND project_member.id = $4
    ) AS "minuteSum"
  FROM (
    SELECT DATE(GENERATE_SERIES(
      $1,
      $2,
      interval '1 day'
    )) AS period_date
  ) AS period_date_series
  ORDER BY period_date DESC;
  ,
  parameters: [startDate, endDate, projectId, memberId],
);
```

```
const allStats = await conn.query(
  query: `
    WITH allSessionsCte AS (
      SELECT session.*,
        FLOOR(EXTRACT(EPOCH FROM (
          COALESCE(
            session."endedAt",
            session."updatedAt"
          ) - session."createdAt"
        )) / 60) AS "durationMinutes"
      FROM session
      LEFT JOIN project_member
        ON project_member.id = session."projectMemberId"
      LEFT JOIN project
        ON project.id = project_member."projectId"
      WHERE session."createdAt" BETWEEN $1 AND DATE($2) + interval '1 day'
        AND project.id = $3
        AND project_member.id = ANY($4::int[])
    )
    SELECT currentPeriodStats.*,
      lastDayStats.*,
      lastWeekStats.*,
      lastMonthStats.*
    FROM (
      SELECT
        COALESCE(SUM("durationMinutes"), 0)
          AS "currentPeriodStatsMinuteSum",
        COALESCE(ROUND(AVG("durationMinutes")), 0)
          AS "currentPeriodStatsMinuteAvg",
        COALESCE(MIN("durationMinutes"), 0)
          AS "currentPeriodStatsMinuteMin",
        COALESCE(MAX("durationMinutes"), 0)
          AS "currentPeriodStatsMinuteMax",
        COUNT(allSessionsCte.id)::integer
          AS "currentPeriodSessionCount"
      FROM allSessionsCte
    ) AS currentPeriodStats, (
      SELECT
        COALESCE(SUM("durationMinutes"), 0)
          AS "lastDayStatsMinuteSum",
        COALESCE(ROUND(AVG("durationMinutes")), 0)
          AS "lastDayStatsMinuteAvg",
        COALESCE(MIN("durationMinutes"), 0)
          AS "lastDayStatsMinuteMin",
        COALESCE(MAX("durationMinutes"), 0)
          AS "lastDayStatsMinuteMax"
      FROM allSessionsCte
      WHERE DATE("createdAt") = DATE($2)
    ) AS lastDayStats, (
      SELECT
        COALESCE(SUM("durationMinutes"), 0)
          AS "lastWeekStatsMinuteSum",
        COALESCE(ROUND(AVG("durationMinutes")), 0)
          AS "lastWeekStatsMinuteAvg",
        COALESCE(MIN("durationMinutes"), 0)
          AS "lastWeekStatsMinuteMin",
        COALESCE(MAX("durationMinutes"), 0)
          AS "lastWeekStatsMinuteMax"
      FROM allSessionsCte
      WHERE EXTRACT(WEEK FROM "createdAt") =
        EXTRACT(WEEK FROM $2)
      AND EXTRACT(YEAR FROM "createdAt") =
        EXTRACT(YEAR FROM $2)
    ) AS lastWeekStats, (
      SELECT
        COALESCE(SUM("durationMinutes"), 0)
          AS "lastMonthStatsMinuteSum",
        COALESCE(ROUND(AVG("durationMinutes")), 0)
          AS "lastMonthStatsMinuteAvg",
        COALESCE(MIN("durationMinutes"), 0)
          AS "lastMonthStatsMinuteMin",
        COALESCE(MAX("durationMinutes"), 0)
          AS "lastMonthStatsMinuteMax"
      FROM allSessionsCte
      WHERE EXTRACT(MONTH FROM "createdAt") =
        EXTRACT(MONTH FROM $2)
      AND EXTRACT(YEAR FROM "createdAt") =
        EXTRACT(YEAR FROM $2)
    ) AS lastMonthStats
  ,
  parameters: [startDate, endDate, projectId, filteredMemberIds],
);
```

Tema oscuro de componentes

Al usar el tema oscuro que Chakra UI te da por defecto, tienes que adaptar muchos colores manualmente. Esto si es código de la aplicación es bastante fácil, pero si hay que hacerlo sobre componentes externos, la cosa se complica mucho.

Hay casos en los que ni siquiera está soportado el cambiar los estilos sobre la marcha o la framework de CSS in JS no es compatible con la framework de la librería de componentes que queremos adaptar al tema oscuro, etc.

Es por eso por lo que hay componentes que he dejado sin estilos, ya que pienso que no merece la pena invertir tanto tiempo en reescribir esta lógica de estilos y es algo que los contribuidores de las librerías conflictivas están trabajando en arreglar.

Los componentes conflictivos en este caso son especialmente el de los gráficos (nivo), el selector de rango de fechas (react-dates) y las tablas (react-base-table)

Rechazos del servidor de correo

Al probar la aplicación, ya sea durante desarrollo o después, he tenido que probar funciones que te envían correos electrónicos para poder realizarse. El servicio de correo electrónico tiene un límite por hora de los correos que puedes enviar desde fuera del cliente oficial.

En el caso de la aplicación, usa ese servicio para enviar correos electrónicos y si enviamos el mismo correo electrónico varias veces en poco tiempo, nos pueden bloquear temporalmente.

No podía hacer nada sobre esta situación, excepto esperar o pagar por una cuenta de empresa, lo cual obviamente no iba a hacer.

5. Menciones especiales

Hay ciertas páginas a las que quiero dar crédito, ya que pienso que es una parte muy obvia de la interfaz gráfica de la aplicación.

<https://undraw.co/illustrations>

Todas las ilustraciones usadas en la aplicación vienen de esta página. Me declaro un fan del trabajo de este equipo. Todas las ilustraciones son completamente libres.

<https://www.namecheap.com/logo-maker/app/>

El logo de la aplicación ha sido creado con esta aplicación. Está basada en inteligencia artificial y es completamente libre y gratuita. En mi opinión, da muy buenos resultados.



6. Bibliografía

Lo que más me ha ayudado a desarrollar este proyecto es la documentación de las propias herramientas que he necesitado y respuestas encontradas en Google.

Entre lo que he usado puedo destacar los siguientes enlaces, su contenido me ha ayudado mucho para desarrollar esta aplicación.

React <https://es.reactjs.org/> <https://reactjs.org/tutorial/tutorial.html>

Conferencia en ReactConf sobre hooks: <https://youtu.be/dpw9EHDh2bM>

Qt <https://www.qt.io/>

TypeORM <http://typeorm.io/>

ExpressJS <https://expressjs.com/es/>

Undraw: <https://undraw.co/illustrations>

Node.js <https://nodejs.org/es/>

TypeScript <https://www.typescriptlang.org/>

C++ <https://www.learncpp.com/>

PostgreSQL <https://www.postgresql.org/>

ChakraUI <https://chakra-ui.com/>

Nivo <https://nivo.rocks/>

Emotion <https://emotion.sh/>

React Router <https://reacttraining.com/react-router/>

React Base Table <https://autodesk.github.io/react-base-table/>

StackOverflow <https://stackoverflow.com/>

Ventana actual Linux <https://github.com/UltimateHackingKeyboard/current-window-linux/blob/master/get-current-window.c>

Ventana actual Windows <https://docs.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-getforegroundwindow>

JWT vs Sesiones

- <https://stackoverflow.com/a/45214431/4673065>
- <http://crypto.net/~joepie91/blog/2016/06/13/stop-using-jwt-for-sessions/>

Diseño de APIs REST correctamente

- <https://www.moesif.com/blog/technical/api-design/REST-API-Design-Best-Practices-for-Sub-and-Nested-Resources/>
- <https://stackoverflow.com/a/26388209/4673065>
- <https://stackoverflow.com/a/7500415/4673065>