



MEMORIA TRABAJO FINAL DE GRADO DE DESARROLLO DE APLICACIONES WEB

mementomori.io



2019/2020

ROMÁN PASTUSHENKO SLAUTSKIY
IES Campanillas

Índice

Analisis y diseño	
Analisis y diseño	12
Atributos.....	10
Bibliografía	19
Características del usuario	4
Conclusiones	
Conclusiones.....	14
Contexto	3
Dependencias	5
Descripción de la implementación de las tecnologías del proyecto.....	13
Descripción general	4
Despliegue	
Despliegue	15
Despliegue local.....	15
Despliegue remoto	16
Diagrama de autenticación	12
Diagrama de componentes React.....	12
Diagrama de diseño de la base de datos.....	13
Especificación de requisitos	
Especificación de requisitos	3
Implementación	
Implementación.....	13
Introducción	12
Introducción	3
Introducción	
Introducción	3
Obligaciones de HLC.....	11
Obligaciones de la presentación de la interfaz web	10
Obligaciones del desarrollo de la interfaz cliente.....	11
Obligaciones del despliegue	11
Obligaciones del diseño	10
Obligaciones del lado de desarrollo de servidor	11
Planteamiento del problema.....	3
Posibles ampliaciones	15
Presentación y objetivos	3
Requerimientos de eficiencia	10
Requerimientos de interfaces	9
Requisitos específicos	5
Restricciones.....	5
Resumen del trabajo realizado y timeframe	14
Tecnologías utilizadas	13
Validación personal.....	15

1. Introducción

- Presentación y objetivos

Este documento describe el trabajo realizado en el proyecto final del grado superior de desarrollo web. El proyecto consiste en el desarrollo de un sitio web interactivo para el usuario, con el fin de que este pueda usarlo para hacer análisis de su vida, y en base a ello tomar decisiones.

La web es accesible desde cualquier navegador por Internet y tiene información referente a la vida del usuario que la usa, esta información se puede modificar y se generaran datos en base a ello.

En cuando a usuarios la página tiene varios tipos. Por un lado, están los usuarios no registrados que únicamente tienen acceso a la página de inicio con información sobre la aplicación. Por otro lado, están los usuarios registrados que pueden acceder a la propia aplicación y usar sus características. Y por último está el usuario administrador que a diferencia del resto de los usuarios registrados tiene un acceso extra a un panel de administración.

- Contexto

El proyecto ha sido realizado para todo el público que tenga acceso a algún navegador web, es para todas las edades. La idea surge del propio concepto memento mori, que encontré en varios blogs y foros de filosofía. Es una adaptación propia.

- Planteamiento del problema

El problema que se planteaba era como convertir un calendario memento mori en algo usable digital.

La idea sale de aquí:

<https://scrawnytobrawny.com/28251>

<https://www.genyfinanceguy.com/2014/11/08/momento-mori-remember-your-mortality/>

Así que la funcionalidad de la aplicación a grandes rasgos tenía que ser, la posibilidad de crear un calendario dividido en semanas de vida del usuario de forma automática, y a su vez añadir opciones interactivas. Y también tenía que existir un administrador que gestionase a todos los usuarios, permitiendo hacer cambios en ellos (Actualizar contraseñas, cambiar nombres, ver usuarios existentes ...)

2. Especificación de requisitos

- Introducción:

○ Propósito:

El propósito de la especificación de requisitos es definir cuáles son los requerimientos que debe tener la aplicación que se va a desarrollar y describir la funcionalidad del usuario a lo largo de ella.

○ Ámbito:

El desarrollo del sitio web está orientado a ofrecer diversos contenidos y funcionalidad que ayuden al usuario a poder obtener un análisis de su vida, mediante su interacción con el calendario, se le irán generando datos visuales con los cuales el mismo podrá dedicarse a sacar

conclusiones. Por tanto, la aplicación distingue dos zonas de diferencias, la parte pública únicamente muestra el funcionamiento de la aplicación con una presentación en una página de inicio, mientras la parte privada es de acceso a la propia app por parte de usuarios registrados.

- Visión global:

A continuación, se realizará la descripción general del sistema desarrollado con sus funciones, características del usuario, restricciones y supuestos.

- **Descripción general:**

- Perspectiva del producto: La aplicación desarrollada pretende dar información general sobre la vida de una persona, y se puede acceder desde cualquier sistema operativo con conexión a internet y un navegador.
- Funciones del producto: Las funciones de la aplicación según el tipo de usuario son las siguientes:
 - Usuario anónimo
 - Autenticación: Dentro de la aplicación hay una página de inicio, donde se debe introducir email y contraseña, de esa forma el usuario podrá acceder a la parte privada de la aplicación.
 - Registro: El usuario puede crear un usuario registrado.
 - Acceso a página de inicio: El usuario puede acceder a la página principal de inicio.
 - Usuario registrado
 - Acceso a calendario: Acceso a la principal función de la aplicación que es el calendario. Esto se traduce en:
 - Generar calendario: Se muestra un panel durante el registro que permite generar un calendario. Es una acción que se puede realizar una única vez.
 - Modificar celdas del calendario: Esta acción permite interactuar con el calendario
 - Ver estadísticas del calendario: Tiene acceso al panel de estadísticas del calendario que surgen en base a nuestro input en el propio calendario.
 - Ver perfil: Podemos ver nuestro propio perfil.
 - Modificar perfil: Permite modificar los datos como la contraseña o el email.
 - Borrar perfil: Permite borrar nuestro perfil.
 - Usuario Administrador:
 - Mismas funciones que usuario registrado, pero con añadidos
 - Listar usuarios: Permite ver una lista de los usuarios disponibles en el sistema. Se podrán ver todos sus datos, eso incluye correo, id, nombres....
 - Modificar usuarios: Permite modificar a todos los usuarios del sistema, eso incluye modificar todos sus datos, eso incluye correo, contraseña, nombres...
 - Borrar usuarios: Permite borrar usuarios del sistema
 - Filtrar usuarios: Permite filtrar usuarios poniendo su correo electrónico.

- Características del usuario:

Se puede diferenciar entre los usuarios registrados y no registrados, dentro de los registrados también hay diferenciaciones.

- Usuarios no registrados: Solo tiene acceso al panel de registro, y a la página de inicio informativa.
- Usuarios registrados: Distinguimos entre:
 - Usuarios simples: Tienen acceso al calendario y a todas las funciones de la app relativas al mismo.
 - Usuarios administradores: Tienen acceso a toda la app, que viene a ser lo mismo que los usuarios simples, y también un panel para controlar al resto de usuarios.

- Restricciones:

Es requerido un navegador y conexión a internet para utilizar el programa.

- Dependencias:

La aplicación precisa para su despliegue de un servidor web que sirva ficheros estáticos, un servidor web que soporte node, y una base de datos postgresql. Y para su uso es necesario un navegador web.

- Requisitos específicos

- Requerimientos funcionales.

A continuación, se describen las diversas funciones que ofrece la aplicación web, clasificadas según el tipo de usuario que accede al servicio.

Usuarios no registrados

APARTADO	DESCRIPCIÓN
TÍTULO	Autenticación
PROPÓSITO	Acceder a la interfaz de la aplicación
ENTRADA	Email y contraseña
PROCESO	Se comprueba que el usuario existe
SALIDA	Se redirige al usuario a la aplicación, y si es su primer login, el usuario será redirigido a un panel en el que tiene que generar su calendario. Si el usuario no existe, se dará un aviso de que el usuario no es correcto.

APARTADO	DESCRIPCIÓN
TÍTULO	Registro
PROPOSITO	Crear un nuevo usuario
ENTRADA	Correo electrónico, nombre 1, nombre 2, contraseña, fecha de nacimiento.
PROCESO	Se comprueba que los datos introducidos sean correctos, que la contraseña sea segura, que la fecha de nacimiento sea correcta, que los nombres estén definidos y que el email sea válido. También se comprueba que el usuario no sea duplicado (email).

SALIDA	Si el registro es correcto, el usuario hace login automático. Si hay algún error, nos avisara el sistema.
--------	---

Usuarios registrados comunes

APARTADO	DESCRIPCION
TITULO	Generar calendario
PROPOSITO	Crear un nuevo calendario para el usuario
ENTRADA	El usuario debe introducir cuantos años espera vivir, este valor debe estar entre 0 y 100. Los valores decimales serán parseados a un entero automáticamente.
SALIDA	Si la creación es correcta, el usuario será redirigido a su calendario.

APARTADO	DESCRIPCION
TITULO	Ver Calendario
PROPOSITO	Cargar el calendario del usuario
ENTRADA	El usuario envía su token de acceso
SALIDA	El usuario recibe su calendario

APARTADO	DESCRIPCION
TITULO	Editar celdas del calendario
PROPOSITO	El usuario puede modificar cada celda disponible del calendario
ENTRADA	El usuario envía una descripción de la semana, y su puntuación del 1 al 5.
SALIDA	La celda cambia de color en base a la puntuación establecida

APARTADO	DESCRIPCION
TITULO	Ver estadísticas del calendario
PROPOSITO	El usuario puede ver graficas relacionadas con el calendario. Hay 4 tipos, todas en el mismo panel.
ENTRADA	El usuario envía su token
SALIDA	El usuario recibe 4 graficas que son mostrados por la interfaz.

APARTADO	DESCRIPCION
TITULO	Modificar perfil
PROPOSITO	El usuario puede modificar sus datos de perfil
ENTRADA	El usuario puede introducir sus datos que pueden ser, nombre1, nombre2, email, y contraseña, para que sean modificados.
SALIDA	Si la salida es satisfactoria se avisará al usuario, si las contraseñas no coinciden, si falta algún nombre, o si el email es incorrecto, el usuario también recibirá un aviso sin que se modifiquen sus datos.

APARTADO	DESCRIPCION
TITULO	Borrar perfil
PROPOSITO	El usuario puede borrar su perfil
ENTRADA	El usuario debe hacer click en borrar su perfil.
SALIDA	El usuario es redirigido a la pagina de inicio si su perfil se borra, si no se avisa al usuario de que ha ocurrido un error.

APARTADO	DESCRIPCION
TITULO	Ver semana actual
PROPOSITO	El usuario puede ver en el calendario la semana actual para orientarse.
SALIDA	La semana actual aparecerá parpadeando en el calendario.

APARTADO	DESCRIPCION
TITULO	SALIR DEL SISTEMA
PROPOSITO	El usuario podrá realizar un logout
ENTRADA	El usuario deberá hacer click en el botón de logout
SALIDA	El usuario invalidara la futura generación de tokens hasta un nuevo login.

Usuarios registrados administradores

Aparte de las funcionalidades de los usuarios registrados comunes dispondrán también de las siguientes.

APARTADO	DESCRIPCION
TITULO	Ver lista de usuarios
PROPOSITO	El usuario puede ver todos los usuarios disponibles en el sistema, y sus datos.
SALIDA	El usuario recibe una lista con todos los usuarios del sistema y sus datos.

APARTADO	DESCRIPCION
TITULO	Filtrar usuarios por su email
PROPOSITO	Obtener usuarios específicos conociendo su email
SALIDA	El administrador recibe los datos del usuario cuyo email se corresponda al introducido
ENTRADA	El usuario debe introducir el correo del usuario a buscar en un input

APARTADO	DESCRIPCION
TITULO	Ordenar usuarios alfabéticamente por su correo
PROPOSITO	El administrador puede ordenar el listado de usuarios alfabéticamente mediante su correo electrónico.
SALIDA	Recibe una lista de usuarios ordenada alfabéticamente
ENTRADA	El usuario debe hacer click en el nombre de la columna

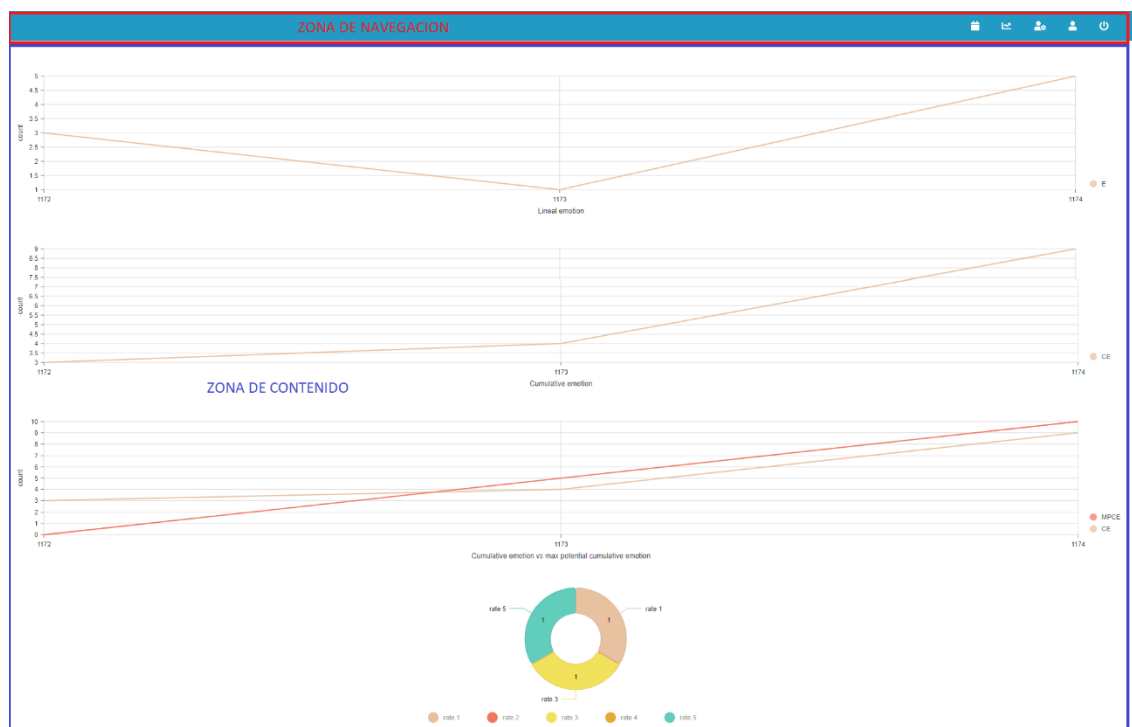
APARTADO	DESCRIPCION
TITULO	Borrar usuario
PROPOSITO	El usuario puede borrar a cualquier usuario del sistema
SALIDA	Mensaje de éxito o fallo en el borrado.
ENTRADA	El usuario debe de hacer click en el botón de borrar al lado del usuario.

APARTADO	DESCRIPCION
TITULO	Modificar usuario
PROPOSITO	El usuario puede modificar todos los datos de cualquier usuario
ENTRADA	El usuario puede introducir nombres nuevos, correos nuevos, y contraseñas nuevas para el resto de los usuarios
SALIDA	Si los datos introducidos fueron correctos el sistema nos avisara, si hubo algún fallo en los datos introducidos, el sistema nos avisara del fallo.

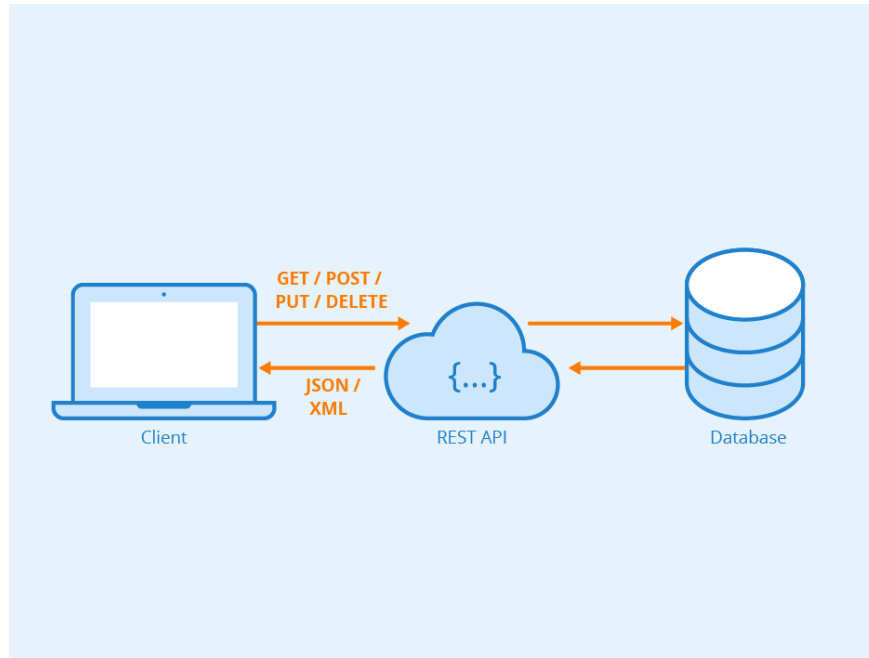
APARTADO	DESCRIPCION
TITULO	Usar paginación de la tabla
PROPOSITO	Para mantener el listado de usuarios conciso, hay un limite de 10 usuarios por página, el usuario podrá desplazarse entre estas páginas para ver a más usuarios.
ENTRADA	El usuario deberá hacer click en el número de página
SALIDA	Se abrirá una nueva página con más usuarios.

- Requerimientos de interfaces

- Interfaces de usuario: Se muestra el formato que conforma la interfaz gráfica en la aplicación, la cuales es similar en toda la aplicación.



- Interfaces hardware: Es una aplicación web, por tanto, se puede visualizar en cualquier dispositivo capaz de ejecutar un navegador web.
- Interfaces software: Es necesario un navegador web y conexión a internet.
- Interfaces de comunicaciones: Las comunicaciones se realizan a través de un protocolo rest api por HTTPS / SSL.



- **Requerimientos de eficiencia.**

La eficiencia dependerá del lado del servidor, pero la influencia en el cliente puede ser por parte del render del calendario. Al ser un elemento pesado, dependerá la capacidad de renderizar el DOM del navegador del usuario, y el estado de conexión de internet del usuario.

La aplicación está adaptada a móviles y pantallas de ordenador de escritorio.

- **Obligaciones del diseño:**

- Estándares cumplidos: Es una web con acceso seguro, con autenticación cuya comunicación entre cliente y servidor se hace encriptada. El idioma para la presentación es el inglés.
- Limitaciones hardware: El servidor debe estar online las 24 horas para atender las peticiones de los usuarios.

- **Atributos**

- Seguridad: La seguridad en la web ha sido correctamente implementada mediante un sistema de JWT, la comunicación entre cliente y servidor siempre se hace de forma encriptada por https. Las claves secretas del servidor se guardan en un proceso .env, para que sea más difícil su localización en caso de ser comprometido. Para acceder a la aplicación se precisa de usuario y contraseña, y en función del tipo de usuario se mostrarán unos u otros elementos en los distintos paneles.

- Mantenimiento: El administrador puede modificar los datos de los usuarios y borrarlos. Pero para todo lo demás es necesario la intervención del desarrollador web.
- **Obligaciones de la presentación de la interfaz web:**
 - Se ha utilizado SASS para realizar las hojas de estilo.
 - La interfaz es responsive, y contiene media queries, flexbox y grid layout
 - Se ha utilizado nivo.rocks para crear graficas.
- **Obligaciones del desarrollo de la interfaz cliente:**
 - Se ha utilizado REACT + REDUX
 - Se ha utilizado javascript ES6
 - Se ha utilizado paginación
 - Se ha creado un CRUD en JS
- **Obligación de HLC:**
 - Se ha comentado el código en ingles
 - La presentación del repositorio público del proyecto está en inglés
- **Obligaciones del despliegue:**
 - Se ha realizado un despliegue remoto en NGINX
 - Se ha documentado todo el código del proyecto con ficheros autogenerados, JSDOC para el cliente, y swagger para la api.
 - Se ha utilizado SCV git, y se ha hosteado el repositorio en github.
 - Se han utilizado ramas como por ejemplo:
<https://github.com/romanpastu/MementoMori/commit/c7304226544849e78fcabf99420e234c69407c5f>
 - El despliegue es parcialmente automatizado:
 - Se accede al directorio de nuestra web, se realiza un “git pull”, y posteriormente se cambia algunas líneas de código contenidas en index.js en la parte del backend, y en constants.js en la parte del front end. Esta todo debidamente comentado en el código fuente.
 - Se ha utilizado tanto ssh como ftp, ssh para conectarse al servidor, y ftp para mover algunos ficheros entre maquinas.
 - La documentación del proyecto está disponible en el repositorio de la página web.
 - Los ficheros de configuración acreditativos del servidor web, están incluidos en el repositorio del proyecto.
- **Obligación del lado de desarrollo de servidor:**
 - Se ha implementado una REST API con node.js en el lado servidor. Y se ha implementado un diseño basado en componentes con gestión de estados globales FLUX en el lado cliente, con REACT.
 - Existen tanto usuarios anónimos, como registrados, como administradores.
 - Se ha utilizado una base de datos relacional de 4 tablas.
 - Existe un servicio de api, hosteado con swagger.

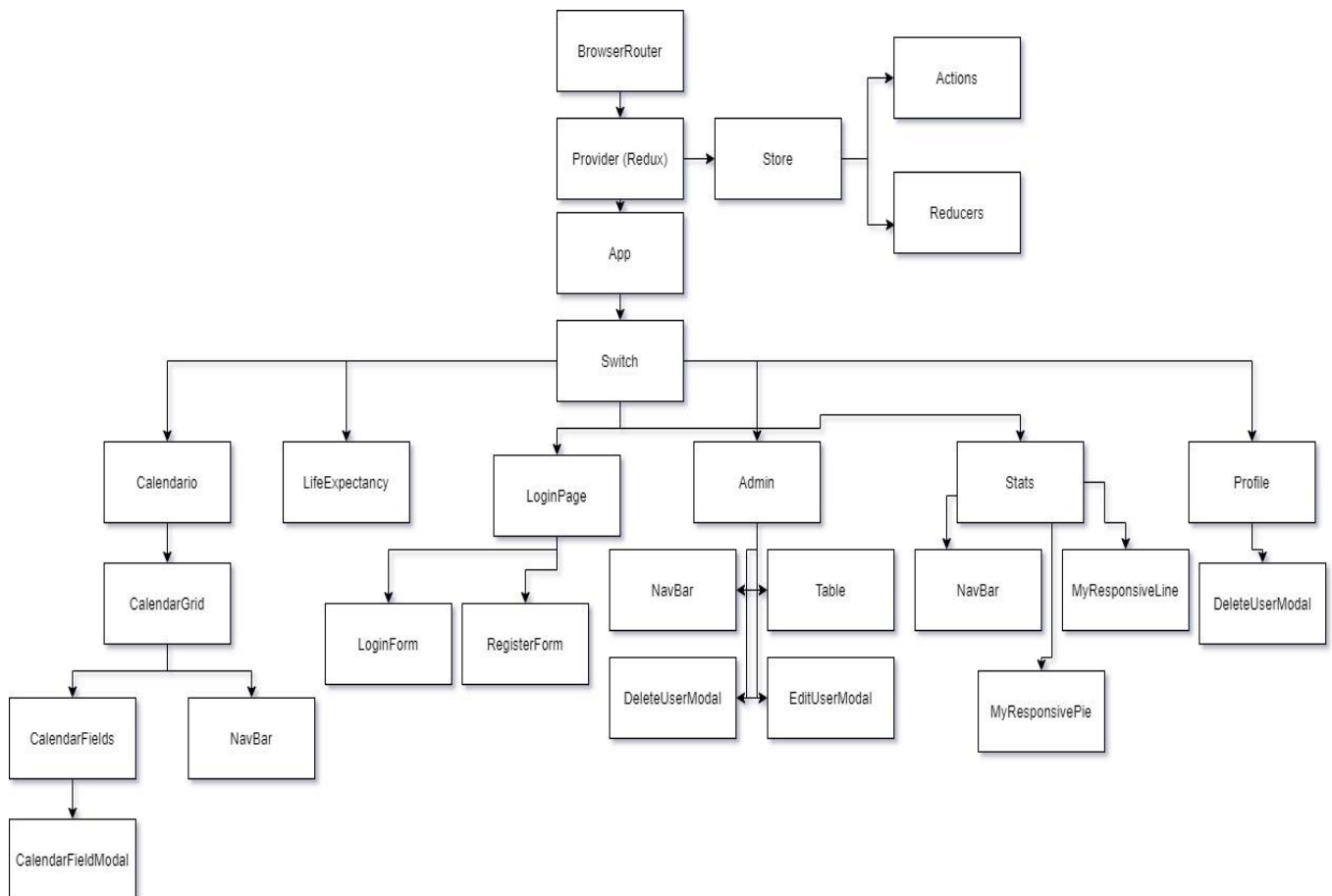
- Se ha utilizado un hosting externo, un vps de digitalocean.

3. Análisis y diseño

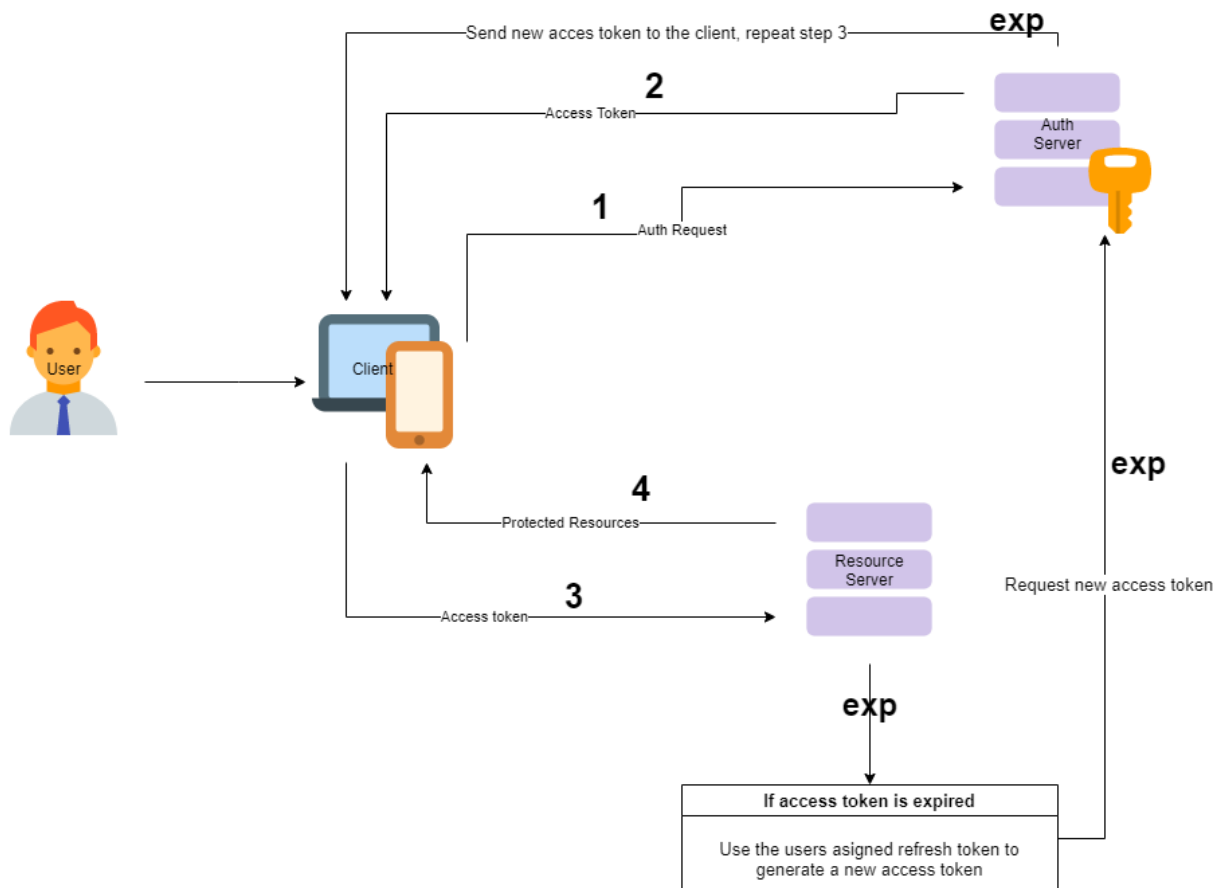
- Introducción

Para realizar el análisis de esta aplicación web, se enseñará diagramas que muestren el diseño de la aplicación.

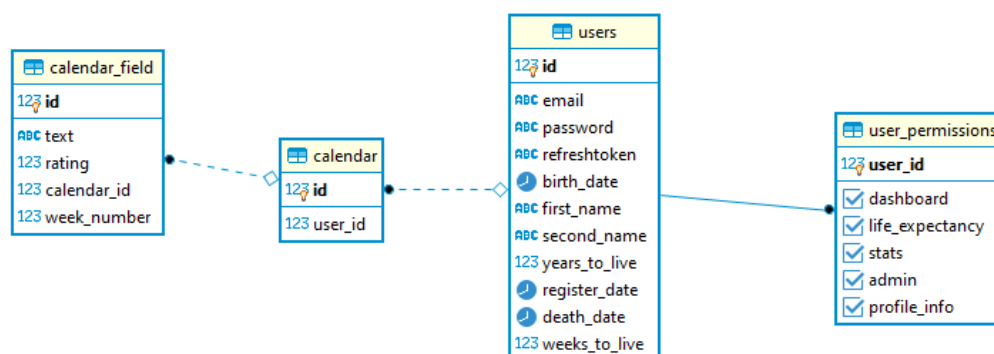
- Diagrama de componentes React.



- Diagrama de autenticación



- Diagrama de diseño de la base de datos



4. Implementación

- Tecnologías utilizadas en el desarrollo del proyecto.

- Backend: Node.js + Express.js + JWT + Postgresql

- App Front End: React – Create React App Boilerplate
- Landing Page: React – Gatsby React Boilerplate (Prerendered)
- Despliegue en servidor: Ubuntu + Nginx
- Docs:
 - Api: Swagger
 - Componentes: JsDocs

- **Descripcion de la implementación de las tecnologías del proyecto.**

- **BackEnd:**
 - Se ha creado un servidor REST con node.js , para ello se ha utilizado el framework express. Se han creado modulos middleware para la autenficación de las distintas rutas API con bearer tokens.
 - Se ha usado una base de datos en postgresql, con 4 tablas. Se ha elegido postgresql debido a que dispone funciones especiales como generateSeries() las cuales eran necesarias para el proyecto y otras BD como mysql no traen integración nativa.
- **App Front End:** Se ha utilizado React, el cual posee una arquitectura de componentes, pero también se ha implementado Redux, para seguir un patron de diseño FLUX. Hay que destacar que no se ha usado para todos los componentes, pero si se ha usado para aquellos casos en los que gestionar estados globales era mas sencillo que usar estados locales de componentes.
- **Landing Page:** Se ha utilizado React, pero una variante especial, esta se llama GATSBY es un boilerplate especial para react cuya diferencia es que permite hacer un prerender del código fuente, y por tanto facilita el posicionamiento SEO. Se ha utilizado porque esta es la parte publica de la aplicación, y la que indexaran los robots de Google.
- **Despliegue en el servidor:** Se ha usado un vps Ubuntu, de DigitalOcean. En este vps se ha instalado una base de datos postgresql, en la cual se ha permitido el acceso externo con contraseña, para poder gestionarla desde cualquier equipo con PgAdmin4. Aparte se ha utilizado un servidor nginx como servidor de contenido, pero nginx se ha utilizado para hostear 3 aplicaciones distintas.
Por un lado, sirve la api, que es el backend, en <https://api.mementomori.io>
Por otro lado, sirve la landing page en <https://mementomori.io>
Y por último sirve la app en <https://app.mementomori.io>

Todo se sirve con certificados ssl generados por let's encrypt.

- La **documentación** de los componentes React se ha generado con jsdoc, y se encuentra en.
<https://github.com/romanpastu/MementoMori/tree/master/FrontEnd/mementomori/docs>
Mientras la documentación de la api se ha generado con swagger y se encuentra en <https://api.mementomori.io/api-docs/>

5. Conclusiones

- Resumen del trabajo realizado y timeframe.

El proyecto comienza a desarrollarse el 9 de enero de 2020. Se comienza a desarrollar la parte de Frontend con datos predefinidos, sin usar base de datos ni manejar sesiones.

El 23 de marzo se diseña la base de datos y se comienza el desarrollo del backend y el sistema de auth.

El 15 de mayo se comienza a desarrollar la landing page.

Las semanas del 1 y 8 de junio se dedica a desplegar el proyecto y documentar.

Por tanto, vemos que es un proyecto que se inicio a principios de año, y ha estado casi medio año en desarrollo.

Lo primero que hice fue plasmar la idea del calendario de forma gráfica, la cual ha tenido varios cambios de diseño a lo largo del desarrollo del proyecto. Una vez estuvo desarrollada la parte gráfica y visual, se procedió a implementar una base de datos que persistiese datos. A su vez se comenzó a implementar el sistema de auth.

Tras ello, ya diseñé el resto de páginas como administración de usuarios, página de perfil etc...

Y por último realice la pagina de inicio por tener alguna especie de presentación, y aproveche para probar una tecnología nueva como React Gatsby.

Tras todo ello, me dedique a hacer pruebas, y buscar fallos que pudiese haber cometido, y por último documente el proyecto.

- Validación personal:

La mayoría de los conocimientos adquiridos para la realización de este trabajo, los he adquirido durante mi estancia en empresa en el periodo DUAL. He utilizado React y node porque personalmente prefiero javascript a PHP, y es algo con lo que me manejo más.

Era un proyecto que ya tenía pensado hacer algún día, pero como se dio la oportunidad aproveche para hacerlo como TFG. Técnicamente la aplicación web es 100% funcional, y lista para producción. Bajo mi punto de vista es una app que se puede usar perfectamente. Pero si que faltaría implementar una verificación de activación de cuentas por correo.

- Posibles ampliaciones:

Como futura ampliación me gustaría migrar el BackEnd a algún framework que no fuese express.js. Como por ejemplo nestjs, y me gustaría añadir funcionalidades en la pagina principal, como un blog, y una tienda. Aparte de añadir verificación por correo.

6. Despliegue

- Despliegue local:

El despliegue local se realizará en localhost, y consta de los siguientes pasos:

1. Clonamos el repositorio del proyecto
2. Descargamos postgresSQL, y lo instalamos junto a pgAdmin para gestionar las bases de datos.
3. Una vez instalado, nos conectamos a la base de datos local con pgAdmin, y creamos una base de datos nueva con los scripts que se encuentran aquí: <https://github.com/romanpastu/MementoMori/blob/master/Database/script.sql>
4. Tras ello, entramos al repositorio que habíamos descargado, y para empezar vamos a lanzar nuestro BackEnd, para ello entramos en la carpeta BackEnd por línea de comandos.
5. Creamos un fichero en el mismo directorio que se llame .env , que contendrá nuestras claves secretas.
6. En .env añadimos las siguientes líneas
 - a. ACCESS_TOKEN_SECRET=una_contraseña_fuerte
 - b. REFRESH_TOKEN_SECRET=una_contraseña_fuerte
 - c. DATABASE_URL=la_url_de_nuestra_base_de_datos
 - Esta url tendrá el siguiente formato:
postgres://usuario:contraseña@localhost:puerto_bd/nombre_bd
7. Tras crear y guardar el archivo, en esa misma carpeta, ejecutamos “npm i” , y esperamos a que se instalen todos los elementos.
8. Ejecutamos “npm run start”, y ya tenemos nuestro servidor corriendo en localhost:1234 , el puerto se puede cambiar al final del fichero index.js del BackEnd.
9. Tras tener corriendo el Backend, nos movemos al directorio FrontEnd, y ejecutamos npm i, para instalar las dependencias, y despues abrimos el fichero src/constants.js , y lo rellenamos con los datos de nuestro BackEnd (url, y puerto).
10. Posteriormente, si ejecutamos en el directorio MementoMori/FrontEnd/mementomori npm run start, tendremos nuestro proyecto corriendo en modo desarrollador, y podremos acceder a el
11. Si interesa correrlo en modo producción tenemos que instalar el modulo serve
12. Escribimos npm install -g serve, y tras instalarse serve -s build
13. Se nos abrirá la versión de producción de la aplicación React en un puerto por defecto
14. Tras tener el BackEnd y el FrontEnd corriendo, pasamos a ejecutar nuestra landing page, nos movemos a su correspondiente directorio, y escribimos “npm i”, y posteriormnete “gatsby build” y tras ello escribimos “gatsby serve” y se nos abrirá la pagina en producción.
15. Con todos los servicios iniciados, damos por concluido el despliegue local. Los comandos de “serve” se pueden reemplazar por “npm run start”(front) o “gatsby develop”(lading) para entrar en modo desarrollo.

- **Despliegue remoto:**

1. En este proceso se utilizara un servidor Ubuntu, damos por supuesto que se han instalado programas como npm y node, necesarios para desplegar cualquier aplicación de node/React.
2. Entramos en el servidor por ssh , y nos desplazamos a /var/www
3. Clonamos nuestro repositorio, y accedemos a él, entramos en cada una de las distintas carpetas de las 3 partes del proyecto (frontend, backend, landingpage) , y ejecutamos npm i en los directorios que contienen package.json para instalar las distintas dependencias.
4. Ahora iremos a la carpeta de BackEnd, y crearemos un fichero .env en la raíz, con los mismos parámetros que habíamos puesto en el despliegue local, pero el parámetro de

DATABASE_URL= lo rellenamos igual que en el despliegue local pero con los datos de nuestro servidor. La contraseña se la pondremos posteriormente, así que pon la que te interese, y el puerto por defecto es 5432 , la bd del proyecto se llamara MementoMori

postgres://usuario:contraseña@ip_servidor:puerto_bd/nombre_bd

5. En la carpeta de BackEnd también tenemos que ir a index.js, y descomentar la línea 35-50 , también tenemos que ir al final del archivo y sustituir app, por httpsServer, tal como indican los comentarios.
6. También tenemos que ir al archivo /src/constants.js de frontEnd , y comentar la línea de const urlBackend, dejando en activo, ósea descomentando la línea de abajo, de api.mementomori.io
7. Una vez instalados todo, y aplicados los cambios, procedemos a ejecutar “npm run build” en la carpeta del frontEnd, y gatsby build en la carpeta de la landing page.
8. Eso creará las carpetas con la versión de producción que deberemos servir con un servidor web.
9. Tras tener los builds realizados, procedemos a instalar postgresql.
“sudo apt update
sudo apt install postgresql postgresql-contrib”
10. Entramos como usuario postgres con “sudo -i -u postgres” , y ejecutamos psql , donde con \password , estableceremos nuestra contraseña (la que metimos en el .env de backend).
11. Tras ello tenemos que permitir el acceso remoto a la base de datos.
12. Entramos en /etc/postgresql/versión/main y editamos dos ficheros
 - a. Postgresql.conf , buscamos la línea listen_addresses = ‘localhost’ , y cambiamos localhost por ‘*’ , para entrar remotamente.
 - b. Pg_hba.conf , bajamos al final del archivo y añadimos las siguientes líneas
“host all all 0.0.0.0/0 md5” y “host all all ::/0 md5” . La opción md5 nos pedirá contraseña, y ::/0 nos permitirá acceder por ipv6, mientras la primera es ipv4”
13. Guardamos los ficheros, y reiniciamos postgresql “sudo service postgresql restart”
14. Ahora volvemos a un equipo con pgAdmin instalado, y nos conectamos a un servidor nuevo , indicando la ip de nuestro servidor, el puerto 5234 (default) , y la contraseña que hayamos puesto. Así tendremos acceso al servidor.
15. Ejecutamos los scripts de creación de bd, igual que en el despliegue local, y ya tendremos la BD lista.
16. Ahora tenemos que ir de vuelta a la carpeta de nuestro proyecto, a la parte de BackEnd, porque vamos a dejar el servidor corriendo en segundo plano.
17. Instalamos forever , “npm i forever” , y tras ello escribimos “forever start index.js” , esto hará que nuestro servidor se quede corriendo en segundo plano.
18. Tras ello es hora de configurar nuestro servidor web, usaremos nginx.
19. Lo instalamos “sudo apt-get install nginx” , y procedemos a configurarlo, para ello nos vamos a su carpeta /etc/nginx/sites-available y creamos 3 ficheros.
 - a. api.mementomori.io (servirá el backend)
 - b. mementomori.io (Servirá la landing page)
 - c. app.mementomori.io (Servirá la app)

20. Los vhost (lo que acabamos de crear) de mementomori.io , y de app.mementomori.io, serán similares, y tendrán el siguiente formato.

```
server {
    server_name 134.122.65.31 app.mementomori.io www.app.mementomori.io;
    root /var/www/cra-memento-mori/MementoMori/FrontEnd/mementomori/build;
    index index.html index.htm;
    location / {
        try_files $uri /index.html =404;
    }
}
```

En server_name tenemos la ip del servidor, y la dirección donde vamos a desplegarlo, en caso de la app es app.mementomori.io, y en el caso de la landing es mementomori.io (Seguido de su consiguiente versión www en ambas).

Root será el directorio del build de nuestros proyectos, que es lo que se creo al hacer “gatsby build” y “npm run build” , serviremos ese directorio.

Una vez creados esos dos ficheros, procedemos a pasarlo por https para tener una conexión encriptada

21. Añadimos certbot con “add-apt-repository ppa:certbot/certbot” , ejecutamos “apt-get update” , y posteriormente “apt-get install python-certbot-nginx”
22. Nos vamos al directorio /etc/nginx/sites-available , y ejecutamos
23. “sudo certbot –nginx -d mementomori.io -d app.mementomori.io”
24. Esto nos creara los certificados ssl, y nos creara registros en el vhost, recuerda indicar la opción 2 de redirect url para que redirija todo el trafico a https cuando el proceso lo pida.
25. Tras ello abrimos crontab con “crontab -e” , y añadimos esta línea
26. " 0 12 * * * /usr/bin/certbot renew –quiet” con el fin de autorenovar los certificados ssl.
27. Ahora generamos el vhost de la api, api.mementomori.io con los siguientes datos

```
server
{
    listen 443;
    listen [::]:443;
    server_name api.mementomori.io;

    location /
    {
        proxy_pass https://127.0.0.1:1111;
    }
}
```






28.

Donde el puerto 1111 equivale al puerto en el que estes corriendo el backend.

29. Tras ello ejecutamos “sudo certbot –nginx -d api.mementomori.io” , y aceptamos el proceso (dando a autoredirect).
30. Entramos en el vhost mementomori.io y modificamos una línea, cambiamos return 400, por “return 301 https://mementomori.io\$request_uri; # managed by Certbot”
31. Reiniciamos nginx con “sudo service nginx restart” , y ya tenemos acceso a todas las paginas web

32. La api estará en <https://api.mementomori.io> , la landing page en <https://mementomori.io> , y la app en “https://app.mementomori.io”

33. Pero antes de que funcione tenemos que apuntar nuestro dominio modificando los records de dns tal que asi.

<input type="checkbox"/>	Type	Host	Value	TTL	
<input type="checkbox"/>	A Record	app	134.122.65.31	Automatic	
<input type="checkbox"/>	A Record	@	134.122.65.31	Automatic	
<input type="checkbox"/>	A Record	www	134.122.65.31	Automatic	
<input type="checkbox"/>	A Record	api	134.122.65.31	Automatic	
 ADD NEW RECORD					

7. Bibliografia

- Stackoverflow (Dudas variadas)
- <https://www.nginx.com/blog/using-free-ssl-tls-certificates-from-lets-encrypt-with-nginx/>
- <https://blog.bigbinary.com/2016/01/23/configure-postgresql-to-allow-remote-connection.html>
- <https://www.evermoretechnologies.com/blog/2017/07/a2ensite-for-nginx-on-ubuntu/#:~:text=a2ensite%20is%20a%20nice%20command,create%20the%20symlink%20through%20bash.&text=If%20you%20want%20to%20turn,off%2C%20just%20remove%20the%20symlink.>
- <https://itnext.io/node-express-letsencrypt-generate-a-free-ssl-certificate-and-run-an-https-server-in-5-minutes-a730fbe528ca>
- <https://stackoverflow.com/questions/12701259/how-to-make-a-node-js-application-run-permanently>
- <https://itsfoss.com/install-postgresql-ubuntu/>