Creación de una herramienta web para gestión de un equipo de trabajo

Adrian Delgado Lopez

Resumen— En todos los trabajos se necesita una coordinación de equipo y para esto es necesario poder llevar a cabo una gestión del equipo y las tareas que se realizan de forma fácil y rápida para que la información esté actualizada al momento. La propuesta de este proyecto para solucionar el problema es una aplicación Web o WebApp accesible desde cualquier dispositivo que permita ver toda la información relevante de las tareas de un solo vistazo, además de integrar una gestión de los trabajadores de la empresa. La aplicación resultante del proyecto es una WebApp funcional con diferentes opciones para visualizar las tareas, además de incluir funcionalidades para la gestión de usuarios y personal de la empresa y un sistema de roles con distintas funcionalidades que varían según el usuario que accede a la aplicación.

Palabras clave— Gestión de tareas, Gestión de personal, Base de datos no relacional, Desarrollo web, WebApp, Angular

Abstract— In every workplace, team coordination is needed. To achieve this, the team and task management must be easily and quickly done so the information about them is always up to date. This project's approach to the mentioned problem is a web application or WebApp accessible from any device which allows to quickly see all the relevant information about the tasks at a glance, besides integrating an easy management of the company staff. The resulting app from this project is a completely functional WebApp with different views for task information and user/staff management, including a role system which changes the app functionality depending on which user logs in.

Index Terms— Task management, Staff management, Non-relational databases, Web development, WebApp, Angular

1 Introducción

1.1 Motivación

El problema a solucionar en este proyecto radica en la organización de un gran equipo de trabajo (más de 50 personas) y el seguimiento de las tareas realizadas por el mismo. Es necesario poder llevar un fácil control sobre las tareas a completar, completadas y en progreso, cuándo se estima terminarlas y qué personal las está llevando a cabo así como el personal que queda disponible para realizar otras tareas. Toda esta información debe ser consultable de forma rápida y sencilla.

La aplicación desarrollada paliará todos estos problemas facilitando una gestión del equipo de trabajo y las tareas llevadas a cabo, permitiendo tan sólo de un vistazo ver qué tareas se cierran, cuándo y permitiendo consultar quién las lleva a cabo y sus pertinentes detalles.

- *E-mail de contacte: 1528186@uab.cat*
- *Menció realitzada: Enginyeria del Software*
- *Treball tutoritzat per: Jorge Bernal del Nozal (Ciències de la Computació)*
- *Curs 2021/22*

1.2 Estado del arte

En el mercado existen diversas herramientas para la gestión y control de tareas, como Trello o Jira, que permiten en mayor o menor medida hacer un seguimiento de la gestión de tareas del equipo y quién las lleva a cabo, cada una con diversas opciones y funcionalidades.

Trello es una herramienta gratuita de gestión de tareas [1] de fácil uso que replica un tablero de Kanban, permitiendo crear y asignar tareas a diferentes miembros del equipo y clasificarlas en columnas personalizables, además de ofrecer algunas funcionalidades añadidas con el modo premium, mientras que Jira es un producto de pago y mucho más completo con una gran adaptabilidad a cada proyecto con un gran número de funcionalidades distintas para diferentes ámbitos de todas las fases del desarrollo software [2].

La aplicación a realizar es similar a estas dos mencionadas, pero ofrece algo más de control sobre el equipo de trabajo que Trello, pero sin llegar a la complejidad y gran

abanico de funcionalidades que proporciona Jira ya que no se necesitan tales funcionalidades, y que podría llegar a ser abrumador, haciendo más mal que bien, por lo que la simpleza es virtud. Por lo tanto, la aplicación a desarrollar ofrecerá un control de tareas similar al control que ofrece Trello pero sin ceñirse al tablero Kanban, y además ofrecerá un control completo sobre los trabajadores del equipo y su respectiva información.

1.3 Objetivos del proyecto

El objetivo principal del proyecto consiste en el desarrollo de una aplicación WebApp que cumpla los requisitos definidos en la sección de Requisitos y solucione los problemas presentados en la Motivación. Para esto, deberá permitir la creación y asignación de tareas, así como facilitar la consulta de información de estas mismas.
Como objetivos secundarios, la aplicación debe contar con:

- Un sistema de registro y log-in seguro
- Un sistema de exportación de datos de tareas
- Un sistema para visualizar tareas que cierran próximamente
- Mensajería por correo electrónico

1.4 Riesgos del proyecto

Riesgo	Probabilidad	Impacto	Contingencia
Falta de experiencia con las tecnologías a usar	Media	Alto	Documentarse sobre las tecnologías.
Retrasos respecto a la planificación	Baja	Alto	Reevaluación de la planificación y modificación de esta.
Problemas a la hora de implementar sistema de correo	Alta	Bajo	Búsqueda de servicios de correo externos o desistir de implementarlo.
Problemas a la hora de implementar las distintas funcionalidades de los roles de usuario	Baja	Media	Limitar el número de roles o las funcionalidades de estos
Complejidad mayor a la esperada al implementar la vista de calendario	Media	Bajo	Limitar la funcionalidad con tal de simplificar su implementación

1.5 Requisitos

1.5.1 Funcionales

- Gestión y asignación de tareas, que pueden tener distintos campos cada una.
- Visualización de las tareas en formato de lista o calendario
- Visualización de las tareas que cierran próximamente
- Exportación de datos de tareas en Excel o CSV
- Sistema de roles y usuarios
- Sistema de envío de correos a través de la app

1.5.2 Hardware

Debido a que el proyecto consiste en una aplicación Web desarrollada con NodeJS, los requisitos Hardware para correrla son mínimos. Se puede ejecutar hasta en dispositivos IoT tales como una Raspberry Pi, por lo que los requisitos son:

	Mínimo	Recomendado	Personal
CPU	Cualquier CPU de 2 núcleos	CPU de 4 núcleos	Ryzen 5 3600X
RAM	1GB	4GB o más	16GB
SO	Windows 7 o posterior, Linux o MacOS	Windows 10, Linux o MacOS	Windows 10 64-bit

1.5.3 Software

- Como base del desarrollo NodeJS [3] es esencial ya que será el gestor de paquetes a través del cual instalar el resto de programas. Existen alternativas como Python [4] que también cuenta con frameworks para el desarrollo web, pero se opta por NodeJS debido a experiencia previa con este software y la facilidad de uso.
- Un navegador web como Google Chrome [5] o Firefox [6]. Personalmente usaré Chrome debido a que es mi navegador habitual.
- Para el back-end y base de datos se usará Firebase v9 SDK [7] debido a que incorpora la base de datos no relacional cloud Firestore [8], está integrado en Node y es fácil de usar con una extensa documentación al ser propiedad de Google, además de incluir herramientas como Firebase Auth [9]. Como alternativa se ha considerado utilizar Express [10] con MongoDB Atlas [11] como BD.
  Cabe destacar que ambas alternativas usarían BD no relacional debido a que su flexibilidad con las estructuras de datos facilita la implementación de tareas con diferentes campos cada una, frente a la rigidez que presentan las relacionales.
- Para desarrollar el front-end se ha escogido Angular 12 [12] por tener experiencia previa con el

framework y por las ventajas que trae usar Typescript frente a otros frameworks que usan Javascript como sería la alternativa React [13].

1.6 Metodología

Como metodología se ha escogido una que permita un desarrollo incremental, con un ciclo de vida iterativo que a su vez permita reuniones de seguimiento y control del proyecto en un pequeño equipo de dos personas. Estos requerimientos nos llevan a escoger la metodología Kanban [14], basada en el desarrollo incremental y flexible ante cambios que puedan surgir durante el proyecto.

1.7 Planificación temporal

El proyecto se estructura en ciertas etapas que comprenden un número de pasos definidos. Cada etapa genera un entregable al final de esta misma. Las etapas del proyecto consisten en las siguientes:

a) Estudio del problema: se realiza un estudio previo del problema planteado para extraer los requisitos de la aplicación que permitan solucionar satisfactoriamente el problema. Como resultado se obtendrá un documento de requisitos.
b) Diseño de la aplicación: se documenta el diseño de las diferentes partes de la aplicación: el diseño de los modelos de la base de datos no relacional, el diseño de interfaz del front-end. Como resultado se obtendrán los diferentes documentos de diseño de cada parte de la aplicación.
c) Diseño e implementación de la primera versión: se desarrollará la primera versión de la aplicación que incluirá las funcionalidades básicas de gestión y visualización de tareas, así como log-in y registro. Como resultado se obtendrá el código ejecutable de esta primera versión.
d) Integración de funcionalidades completas: se desarrollarán las funcionalidades restantes y se completarán las básicas. Como resultado se obtendrá el código completo y ejecutable de la aplicación.
e) Pruebas de la aplicación en entorno controlado: se correrá una fase de testeo utilizando herramientas de testing en busca de errores indeseados o fallos en el sistema. Como resultado se obtendrá un código depurado con el mínimo de bugs posibles.

Se puede observar un diagrama de Gantt con todas estas fases y sus tareas pertinentes en la figura 15 en el anexo.

2. DISEÑO DE LA APLICACIÓN

2.1 Diseño de BD

En este proyecto se utiliza una BD no relacional, concretamente Firestore, debido al requisito de que las tareas puedan contener información en campos completamente distintos y únicos. Esto se traduce en columnas en una BD relacional las cuales tendrán información en algunas filas y en otras no, pero sobre todo habrá que alterar el esquema de la tabla de tareas siempre que se añada un campo nuevo. En una BD no relacional esto no es necesario gracias a la flexibilidad que ofrecen, permitiendo guardar información de tareas con campos distintos sin necesidad de alterar nada. La base de datos estará formada por estas cuatro colecciones:

2.1.1 Colección Tareas

Esta colección guarda la información propia de las tareas, tanto la información básica (nombre, fecha de entrega, trabajadores asignados, si ha sido verificada o el tipo de tarea) como toda la información adicional que se pueda añadir durante la creación de esta a través de campos adicionales añadidos en forma de array.

2.1.2 Colección Personal

Esta colección contiene la información de los trabajadores a gestionar: sus datos personales como nombre y apellidos, datos profesionales como su especialidad de tarea (en qué tipo de tarea está especializado), su rol, disponibilidad, correo de contacto y plataformas que tiene disponibles.

2.1.3 Colección Roles

Una pequeña colección que guarda la información de cada rol: el nombre del rol (administrador, verificador o usuario) y los permisos asociados a este rol que dictarán qué acciones puede tomar en la aplicación.

2.1.4 Colección Especialidades

Una pequeña colección que tan guarda el nombre de todas las especialidades definidas y los campos adicionales que debe tener cada especialización a modo de plantilla. Esta colección existe con el objetivo de facilitar las querys necesarias desde la aplicación cuando se necesita recuperar las diferentes especialidades que hay.

2.2 Arquitectura del código

Para este proyecto se ha optado por aplicar una arquitectura basada en tres capas: la capa más exterior contiene todos los componentes visuales que conforman las páginas y elementos con los que puede interactuar el usuario, la capa más interna contendrá *services* que interactúan con el back-end y la base de datos y son inyectables (por inyección de dependencia) en la capa intermedia, formada por casos de uso que comunican la capa externa con la capa interna, añadiendo una capa de abstracción de por medio.

El objetivo de aplicar esta arquitectura es buscar reducir el acoplamiento en el código de la aplicación, separando todo lo relacionado con la GUI visible e interactivo para el usuario de las llamadas a la base de datos y back-end.

2.3 Roles de usuario

2.3.1 Administrador

El rol de Administrador corresponde al rol más alto en la jerarquía de roles establecida, pudiendo acceder a todas las funcionalidades de la aplicación al completo. Tendrá todos los permisos para consultar, crear, eliminar, verificar o modificar tareas, consultar, modificar, eliminar y contactar usuarios, además de asignar a usuarios a tareas y ver todas las tareas registradas. Consultar figura 16 del anexo para ver el diagrama de casos de usos del administrador.

2.3.2 Verificador

El rol de Verificador corresponde al rol intermedio en la jerarquía de roles establecida, pudiendo acceder a todas las pantallas de la aplicación pero sin permiso para acceder a todas sus funcionalidades completas. Contará con permisos para consultar todas las tareas registradas y contactar usuarios, además de verificar las tareas completadas. No podrá modificar, eliminar ni crear tareas ni usuarios. Consultar la figura 17 del anexo para ver el diagrama de casos de uso del Verificador

2.3.3 User

El rol de User corresponde al rol más limitado y bajo en la jerarquía de roles establecida, pudiendo acceder sólo a información asociada al mismo usuario. Tan sólo podrá consultar tareas, concretamente las tareas que tenga asignadas la persona, en caso de que tenga asignadas. No podrá modificar, eliminar ni crear tareas ni usuarios, ni tampoco contactar con otros usuarios. En la siguiente figura se puede ver el diagrama de casos de uso del usuario.

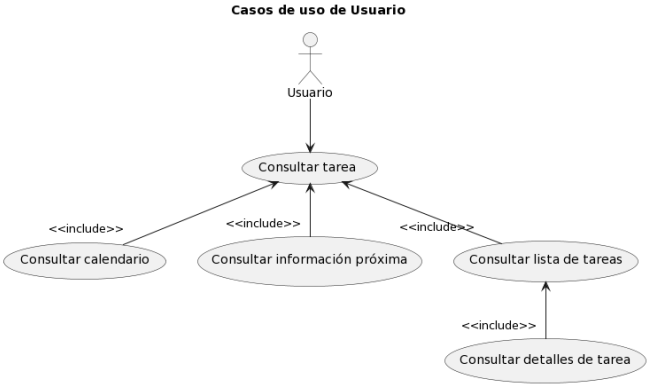

Fig 1. Casos de uso de usuario

2.4 Prototipo visual

Durante la fase del diseño se ha desarrollado un prototipo visual de la aplicación con el objetivo de confirmar que alumno y tutor están de acuerdo en lo que se quiere desarrollar, además de tener una referencia visual en la que basarse al desarrollar la aplicación. Este prototipo tiene versiones adaptadas para cada rol ya que las acciones que pueden tomar en la aplicación son diferentes. A continuación se muestran algunas de las pantallas del prototipo

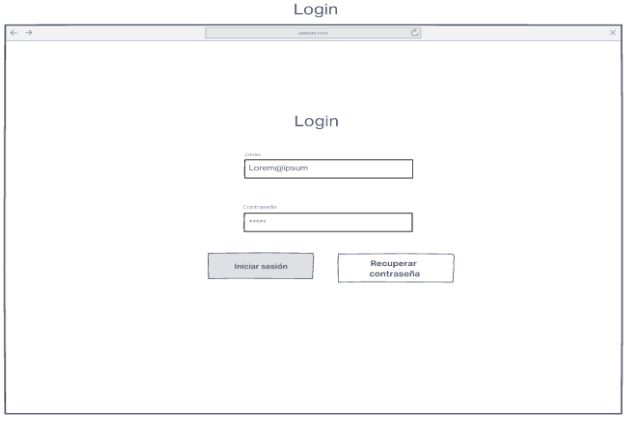

Fig. 2 Pantalla de login del prototipo

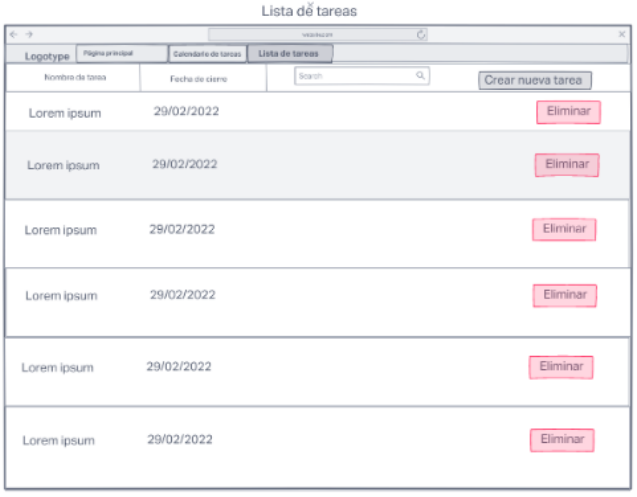

Fig. 3 Pantalla de listado de tareas del prototipo

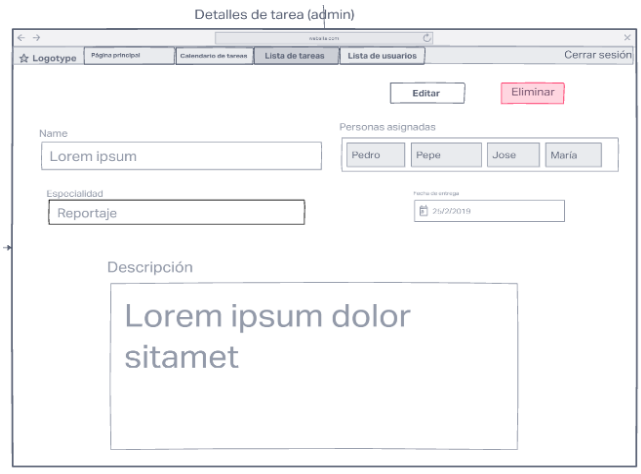

Fig 4. Pantalla de creación de tarea del prototipo

Fig 6. Listado de tareas

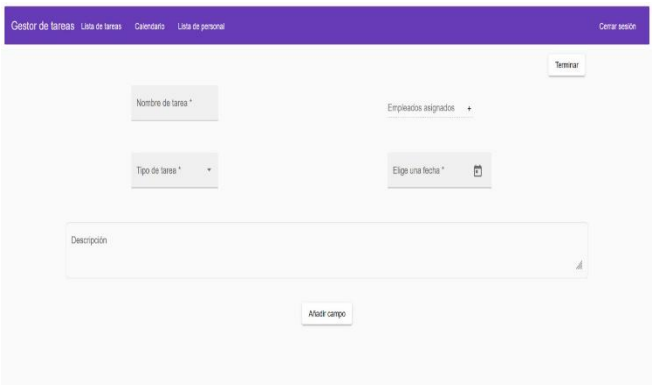

Fig. 7 Pantalla de creación de tareas

3. DESARROLLO

3.1 Primera versión de la aplicación

Para la primera versión de la aplicación se han desarrollado los módulos definidos en la planificación de esta fase, incluyendo: módulo de inicio de sesión y sistema de autenticación de usuario que hace uso de Firebase Auth [9], un completo listado de tareas en curso, filtrables por nombres y ordenables por distintos campos, el módulo de creación de tareas y la navegación por la aplicación a través del uso de una barra de navegación (navbar [15]).

Todo el desarrollo de estos módulos se ha hecho siguiendo la arquitectura propuesta de tres capas y se ha hecho una documentación paralela de estos módulos, la estructura de carpetas del proyecto y todas las funcionalidades añadidas. A continuación se muestra el resultado visual de las pantallas prototipadas mostradas anteriormente

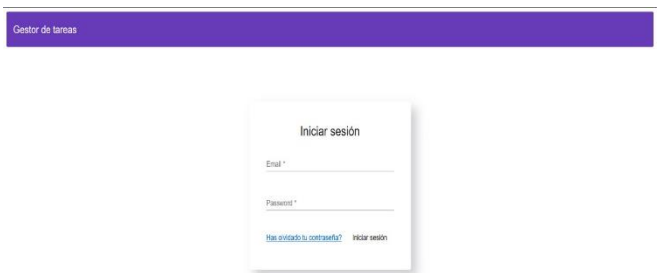

Fig. 5 Pantalla de login

3.2 Desarrollo de la versión final

Para la versión final de la aplicación se han desarrollado los módulos definidos en la planificación de esta fase, incluyendo pero no limitado a: completar funcionalidades del módulo de tareas, implementar el módulo de usuarios, similar al de tareas, además del sistema de roles explicado en el apartado 2.3. Además de esto, se ha hecho una revisión de la seguridad de la base de datos con tal de mejorar su robustez, y se han añadido algunas pequeñas funcionalidades para mejorar la experiencia de usuario pedidas por el tutor.

Tal y como se define en la planificación, durante el desarrollo de esta fase se ha elaborado documentación sobre detalles y estructura del proyecto así como sus funcionalidades separadas por módulos.

3.2.1 Módulo de tareas

En esta versión el módulo de tareas se ha visto completado con las funcionalidades restantes, tales como la descarga de datos, diferentes vistas de tareas, historial de tareas, filtro de trabajadores según el tipo de tarea en el que se especializan y plantillas basadas en el tipo de tarea, además de campos extra definidos por el usuario si lo encuentra necesario.

A continuación se muestran las pantallas del módulo de tareas con sus funcionalidades completas:

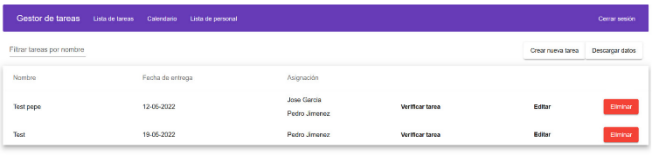

Fig. 8 Lista de tareas completa

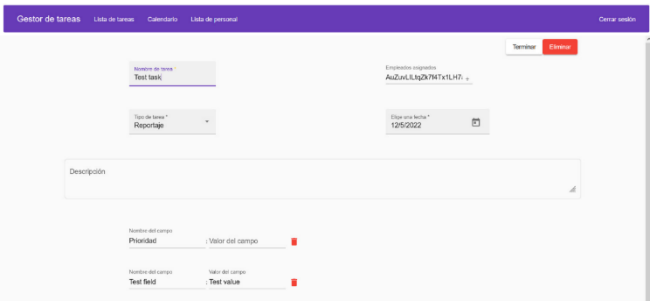

Fig. 9 Detalles de tarea de tipo Reportaje con un campo añadido

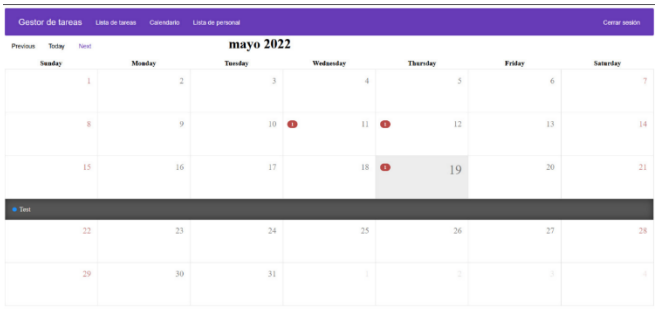

Fig. 10 Vista de calendario

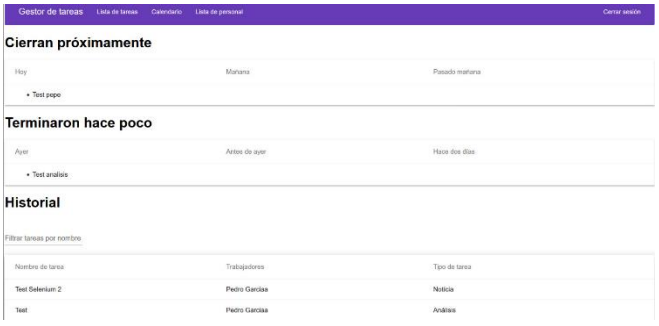

Fig. 11 Vista resumen

3.2.2 Módulo de usuarios

En esta versión se ha implementado el módulo de usuarios, el cual permite crear, modificar y desactivar usuarios. Este módulo es muy similar a la primera versión del módulo de tareas, ya que está compuesto por una lista de usuarios que muestra algunos detalles de cada usuario además de ser ordenable y filtrable, y pantallas de creación y edición de usuarios con todos los datos necesarios para crear sus respectivas cuentas de usuario (correo y contraseña para la autenticación, rol para los permisos de usuario, e información de trabajador).

A continuación se muestran las pantallas mencionadas del módulo de usuarios:

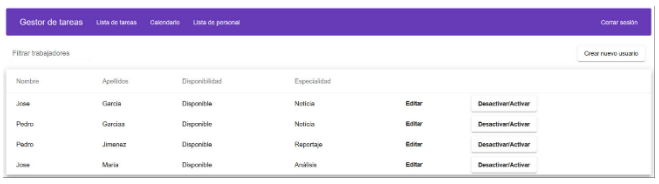

Fig. 12 Lista de usuarios

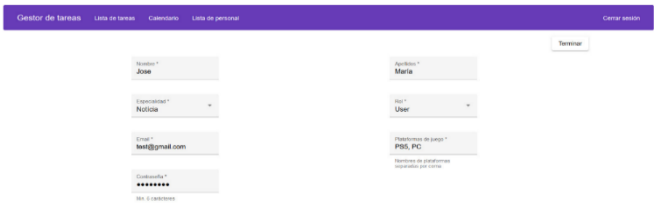

Fig. 13 Creación de usuario

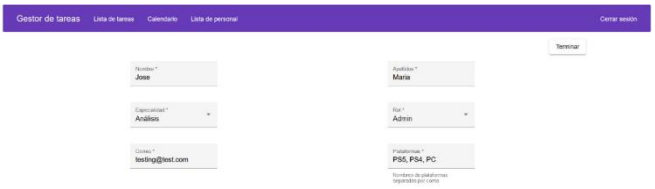

Fig. 14 Modificación de usuario

3.2.3 Seguridad de la base de datos

Durante el desarrollo de esta última versión se ha puesto énfasis en la seguridad de los datos guardados en la base de datos. Pese a existir un sistema jerárquico de roles que solo debería permitir modificar datos desde la aplicación a los usuarios administradores, hay que tener en cuenta la posibilidad de que exista un fallo de seguridad que permita saltarse (o modificar) los permisos de usuario, de forma que permitiría a un usuario malintencionado acceder y/o modificar datos que no debería.

Para evitar que esto ocurra y añadir más seguridad al sistema, se ha hecho uso de las reglas de seguridad de Firestore, que permiten definir qué peticiones de lectura o escritura rechazar basado en los permisos del usuario que está realizando la petición (usando sólo el token de autenticación recibido) y si es un usuario autenticado. De esta forma, aunque un usuario malicioso consiga saltarse los permisos desde la aplicación, si su cuenta no está registrada con los permisos necesarios sus peticiones no serán aprobadas.

4. Resultados

Al estar trabajando en una WebApp no se pueden obtener resultados cuantitativos, pero sí se puede buscar asegurar la calidad del producto a través de testing de la aplicación.

Con el objetivo de asegurarse de que la experiencia de usuario es la deseada y funciona como se espera, se ha ejecutado testing end-to-end (E2E) [16] desde el punto de vista de los 3 usuarios de la aplicación (Administrador, Verificador, Usuario). El objetivo de hacer este test consiste en buscar fallos en el flujo de la aplicación. En el contexto de la aplicación desarrollada se busca si el flujo de navegación y acciones del usuario puede ser abruptamente interrumpido ya sea por bugs en el programa o fallos en intrínsecos en el diseño de la aplicación. El resultado

de llevar a cabo este test ha sido positivo, ya que no se han detectado fallos en dicho flujo de navegación y la experiencia de usuario no se ha visto afectada en ningún momento.

También se ha llevado a cabo testing unitario de los componentes de la aplicación, buscando llevar más al límite cada componente con casos extremos que no suelen ocurrir, buscando si el componente reacciona como se espera y si la experiencia de usuario se ve afectada. Como resultado de llevar a cabo este testing se han encontrado varios bugs menores, que no rompen la aplicación, pero pueden ser un inconveniente para el usuario (como ejemplo, en la creación de tareas se encontró un bug que podía borrar campos que no debían borrarse al cambiar el tipo de tarea), los cuales fueron solucionados y, de nuevo, testeadas las funcionalidades para poner a prueba las soluciones, que resultaron ser efectivas.

Cabe comentar que por la parte de Firebase se ha llevado a cabo test de la seguridad de la base de datos y las reglas de seguridad que regulan qué colecciones y cuándo se pueden leer o modificar, a través de la herramienta de simulación que incluye el propio módulo de Firestore. Esta herramienta permite simular peticiones CRUD (Create, Read, Update and Delete) con diferentes parámetros (como el documento objetivo, los datos que se quieren modificar o datos de autenticación) con el objetivo de comprobar si las reglas establecidas funcionan como es esperado o existe algún caso en el que se aprueben peticiones ilegales o, por lo contrario, se rechacen peticiones legales. Como resultado del test de seguridad se han obtenido diversos casos en los que las reglas de seguridad no funcionaban correctamente, los cuales se han usado para mejorarlas y repetir el proceso de forma iterativa, hasta obtener un set de normas de seguridad que funcionan correctamente basándose en los roles definidos.

Para finalizar, es necesario comentar que la gran mayoría del testing realizado a la aplicación ha sido de forma manual. La principal razón es que al ser una aplicación prácticamente solo front-end no hay tanta lógica compleja como podría haber en la API de un backend, por lo que no ha sido necesario hacer un testing complejo y no ha habido la necesidad de automatizar el testing.

A continuación se adjunta un vídeo de demostración de las funcionalidades de la aplicación desde el punto de vista de cada tipo de usuario, con tal de ofrecer una visión completa del funcionamiento de la aplicación de forma fluida:

https://youtu.be/NWMgc4N6HIs

5. CONCLUSIONES

5.1 Experiencia personal

El proyecto ha sido una gran experiencia, y a la vez un reto, ya que ha resultado una oportunidad de poner a prueba habilidades y conocimientos adquiridos durante la carrera. Además de eso, he podido usar la oportunidad del proyecto para aprender o profundizar en tecnologías como Angular o Firebase y obtener experiencia en estas, cosa que puede ser útil de cara al futuro. También he puesto a prueba competencias más concretas de la mención, como la captación de requisitos, el diseño de aplicación y bases de datos o el testing durante las diferentes fases del desarrollo del trabajo. Finalmente, cabe destacar que el hecho de que la aplicación pretenda tener un uso y propósito más allá de simplemente ser un trabajo académico añade motivación a la hora de trabajar en ella y satisfacción al superar los diferentes retos encontrados durante el desarrollo.

5.2 Conclusiones del proyecto

Respecto al proyecto como tal, pese a encontrar algunos problemas durante el desarrollo, principalmente limitaciones impuestas por la versión gratuita de Firebase, se han completado todos los módulos principales y sus funcionalidades acordados durante la fase de captación de requisitos de la aplicación, además de algunos ajustes y pequeñas mejoras sugeridas durante el periodo de desarrollo. Para ello se han tenido que superar algunos retos encontrados durante el desarrollo, los cuales se pueden agrupar principalmente en problemas por desconocimiento de las tecnologías usadas y limitaciones impuestas por la versión gratuita de Firebase.

El primer grupo de retos se ha solucionado gracias a la documentación de Google de las diferentes funcionalidades proporcionadas por Firebase además del uso de foros externos como StackOverflow y la documentación de Angular y las librerías usadas (como AngularFire), mientras que el segundo grupo ha resultado ser algo más complicado de solucionar. Principalmente, para solucionar estos problemas se han aplicado soluciones alternativas a la originalmente propuesta, siempre consultando al tutor primero y con su consentimiento para modificar algunos requisitos iniciales. Por ejemplo, en un principio el usuario administrador podría eliminar usuarios de la aplicación, pero debido a las limitaciones impuestas por Firebase, tras consultar al tutor se optó por la desactivación de estos para facilitar el desarrollo y no sobre complicar la solución.

5.3 Planificación del proyecto

Echando la vista atrás, la planificación del proyecto se ha cumplido en gran medida. El desarrollo de la primera versión se atrasó aproximadamente una semana debido a que se dedicó más tiempo del esperado a investigar las tecnologías y se subestimó el desconocimiento de las mismas, pero una vez asentada la base de conocimiento sobre estas el desarrollo fue mucho más fluido para la siguiente y última versión, por lo que se recuperó el tiempo perdido y se finalizó sobre la fecha esperada, llegando a la fase de testing sin retrasos, la cual también fue llevada a cabo sin problemas alguno. En líneas generales,

la planificación del proyecto establecida ha sido acertada y con casi sin ningún retraso, por lo que no ha sido necesario llevar a cabo una replanificación de las fases de proyecto.

5.4 Trabajo futuro

Pese a que la aplicación se encuentra en un estado perfectamente funcional y usable para el propósito que se diseñó, aún podría mejorarse para incluir más funcionalidades y pulir las ya existentes. Para conseguir esto primero hace falta afrontar las limitaciones que impone Firebase y bien desarrollar un back-end propio que incluya Firebase Admin o bien usar el plan de pago de Firebase con tal de tener acceso a Firebase Functions, que a grandes rasgos actuarían como la API de un back-end habitual, pero integrado en el propio Firebase.

Una vez solucionado el problema de las limitaciones de Firebase, se podrían introducir mejoras de calidad de vida para el usuario como la mensajería directa por correo desde dentro de la propia aplicación o la posibilidad de descargar datos de los usuarios tal y como se hace con las tareas, además de mejorar la seguridad de la aplicación añadiendo una capa adicional de seguridad en el back-end.

Aprovechar la estructura de la base de datos para añadir más plantillas de tareas a través de un sistema que permita al administrador guardar, modificar o eliminar plantillas con los campos que crea correspondientes es algo a tener en cuenta, ya que actualmente es algo que se hace de forma manual introduciendo los campos directamente a la BD. Esto permitiría flexibilizar y facilitar aún más la creación de tareas para el usuario. Con ello se reduce la necesidad de crear campos nuevos recurrentes en diferentes tareas, eliminando el tedio y mejorando la calidad de vida del usuario.

Finalmente, según las necesidades del equipo se podría investigar si hay algún tipo de vista adicional que pueda ayudar a la organización del equipo de trabajo. Por ejemplo, si es un equipo que trabaja mucho con Kanban, implementar una vista en forma de tablero de forma similar a como lo hace Trello podría facilitar la gestión y visualización de las tareas y el estado en el que están.

Agradecimientos

Quiero expresar mis agradecimientos a mi tutor Jorge Bernal por prestarse a resolver mis dudas durante el proyecto, a mi hermano Oscar Delgado por prestarse a ayudarme y resolver mis dudas sobre desarrollo web gracias a su experiencia en el campo, además de familia y amigos que me han dado su apoyo durante el desarrollo de este proyecto.

Bibliografia

[1] *Trello facilita a Los equipos La Gestión de Proyectos y tareas*. Trello. Retrieved June 5, 2022, from https://trello.com/es/tour

[2] *¿Para qué se utiliza jira software?* Atlassian. Retrieved June 5, 2022, from https://www.atlassian.com/es/software/jira/guides/use-cases/what-is-jira-used-for

[3] Node.js. Retrieved March 3, 2022, from https://nodejs.org/es/

[4] Python. *Python.org*. Retrieved March 3, 2022, from https://www.python.org/

[5] Google. *Navegador Web Google Chrome*. Google, Retrieved March 3, 2022, from https://www.google.com/intl/es/chrome/

[6] *Download the fastest Firefox ever*. Mozilla. Retrieved March 3, 2022, from https://www.mozilla.org/en-US/firefox/new/

[7] Google. *Add firebase to your JavaScript project | firebase documentation*. Google. Retrieved March 3, 2022, from https://firebase.google.com/docs/web/setup

[8] Google. Get started with Cloud Firestore | firebase documentation. Google. Retrieved March 3, 2022, from https://firebase.google.com/docs/firestore/quickstart

[9] Google. *Firebase authentication | firebase documentation*. Google. Retrieved March 3, 2022, from https://firebase.google.com/docs/auth?hl=es&authuser=0

[10] *Infraestructura de Aplicaciones Web Node.js*. Express. Retrieved March 3, 2022, from https://expressjs.com/es/

[11] *MongoDB atlas: Multi-cloud developer data platform*. MongoDB. Retrieved March 3, 2022, from https://www.mongodb.com/atlas

[12] *Angular – The modern web developer's platform*. Retrieved March 3, 2022, from https://angular.io/

[13] *React – una biblioteca de javascript para construir interfaces de usuario.* Retrieved March 3, 2022, from https://es.reactjs.org/

[14] Ahmad, M. O., Markkula, J., Oivo, M. *Kanban in software development: A systematic literature review*. IEEE Xplore. Retrieved March 3, 2022, from https://ieeexplore.ieee.org/abstract/document/6619482

[15] Wikimedia Foundation. (2022, May 20). *Navigation Bar*. Wikipedia. Retrieved April 7, 2022, from https://en.wikipedia.org/wiki/Navigation_bar

[16] *End to end testing: A detailed guide*. BrowserStack. (2021, July 20). Retrieved May 20, 2022, from https://www.browserstack.com/guide/end-to-end-testing

ANEXO

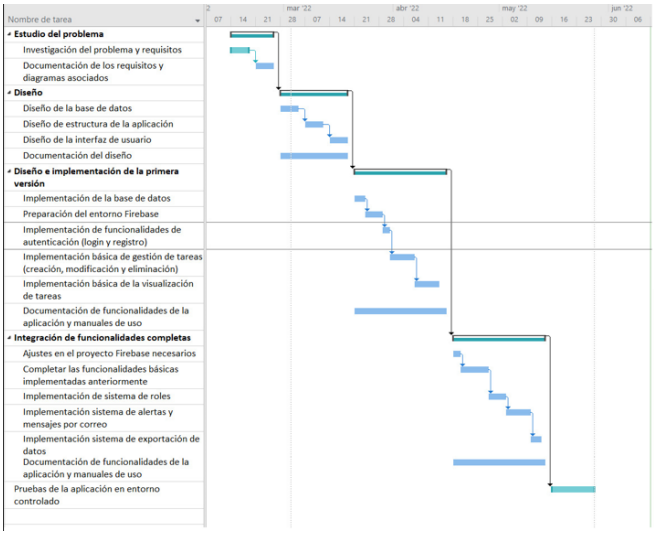

Fig. 15 Diagrama de Gantt de planificación temporal.

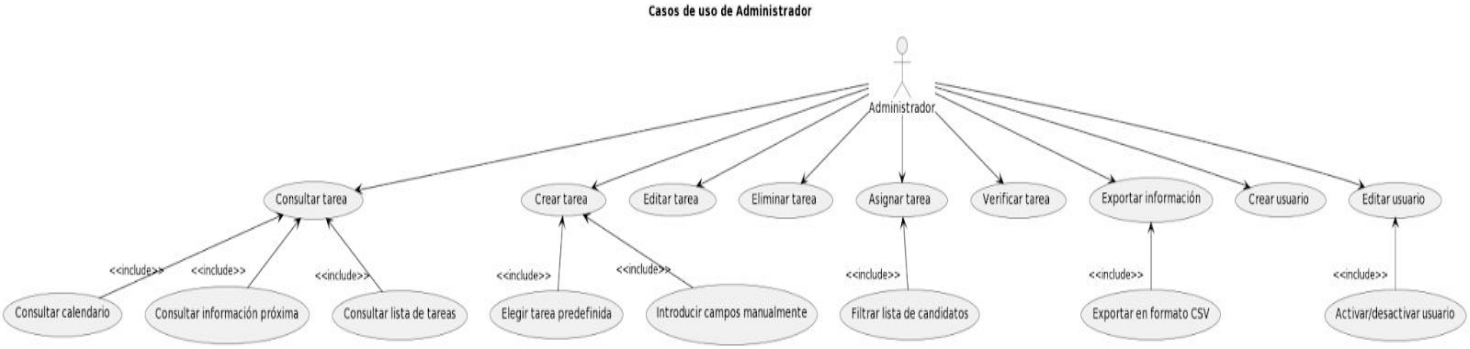

Fig. 16 Casos de uso de administrador

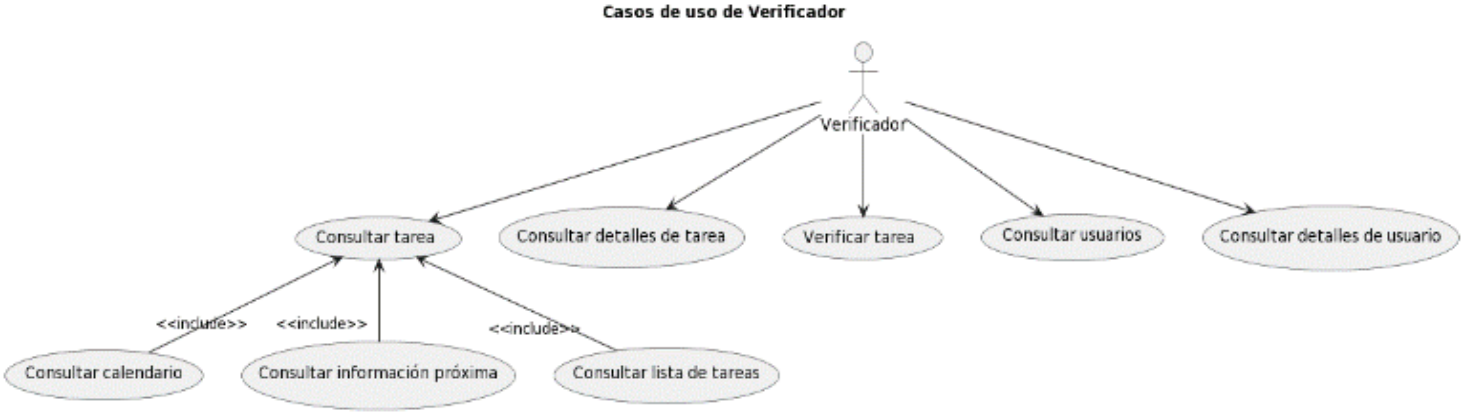

Fig 17. Casos de uso de verificador