I.E.S Infanta Elena

FAMILIA PROFESIONAL DE INFORMÁTICA Y COMUNICACIONES

CICLO FORMATIVO DE GRADO SUPERIOR DESARROLLO DE APLICACIONES WEB

TRABAJO FIN DE CICLO

TeacherCover Web App

Pablo Sainz y Marta Torres

Curso 2022-2023

TÍTULO: TeacherCover Web App

AUTOR: Pablo Sainz del Nogal y Marta Zahara Torres de la Casa

TUTOR DEL PROYECTO: Carlos Tessier Fernández

FECHA DE LECTURA: xx de Junio de 2023

CALIFICACIÓN:

Fdo: Pablo Sainz del Nogal & Marta Zahara Torres de la Casa

Tutor/a del Proyecto

Teacher Cover

Resumen

Este proyecto está enfocado en desarrollar una **aplicación web**, TeacherCover, para la gestión de las guardias de los docentes del IES Infanta Elena. Aplicando una metodología ágil basada en Scrum se llevó a cabo un análisis previo de las necesidades de los profesores mediante la realización de entrevistas. La aplicación creada empleando como framework Angular con TypeScript para el front-end y firebase para el Back-end busca conseguir mediante una **interfaz** intuitiva que permita a los usuarios un acceso fácil y cómodo a estos trámites. El objetivo de ésta es que el centro tenga una mejora significativa del proceso de guardias y ahorro de tiempo y recursos para el centro educativo en general. En conclusión debe ser una herramienta que permita la optimización de la gestión de guardias.

Abstract

This project is focused on developing a web application, TeacherCover, for the management of the standby teachers of the IES Infanta Elena. Applying an agile methodology based on Scrum, a prior analysis of the teachers' needs was carried through interviews. The application created using Angular framework with TypeScript for the front-end development and firebase for the back-end seeks to achieve through an intuitive interface that allows users easy and comfortable access to these procedures. The objective of this application is that the center achieve a significant improvement in the standby teachers management process, saving time and resources for the educational center in general. In conclusion, it must be a tool that allows the optimization of the on-call teachers management.

Índice

Índice de figuras e imágenes

1. Introducción: Objetivos y Motivación

El objetivo del proyecto es la realización de una aplicación web que permita a los profesores y jefatura del instituto IES Infanta Elena realizar la gestión de las guardias del profesorado.

Actualmente en el centro no está informatizado el sistema de gestión de guardias. Consta de 3 documentos en papel: Un horario en el que se especifica en cada franja horario que profesores se encuentran de guardia o de suplencia de guardia, un documento en el que se especifican las guardias del día que hay que cubrir con, si corresponde, información adicional, en el que los docentes de guardia y suplencia firman como que están presentes, y un tercer documento en el que el docente se apunta las guardias que realiza. Todo ésto dificulta que se lleve un correcto seguimiento, además que obliga a los docentes a tener que pasar por la sala de profesores en su hora de guardia para revisar todos los documentos indicados anteriormente.

El objetivo que mueve la realización de éste proyecto es poder proporcionar al centro una herramienta que simplifique la gestión de las guardias y por tanto la convivencia de los docentes en el centro. Facilitar la revisión de qué guardias corresponden al docente de guardia sin tener que desplazarse a la sala de profesores agilizando por tanto el proceso.

Para ello se pretende que desde un perfil de administración de la aplicación se permita a los departamentos de administración y jefatura incluir nuevas guardias o consultar el estado de las guardias actuales, así como comprobar las horas de guardias realizadas por los docentes y poder además realizar modificaciones de datos.

Desde el perfil de usuario el profesorado puede ser notificado de las guardias correspondientes junto con información importante relativa a las mismas, además de llevar un control de las guardias realizadas.

Para comprender mejor las necesidades y funcionalidades que ha de abarcar la aplicación, se ha profundizado en la problemática a resolver preguntando directamente a algunos de los usuarios a los que está enfocada la aplicación.
Se han realizado para ello entrevistas a distintos docentes del centro para recoger las funcionalidades,aspectos claves que debe cumplir la aplicación para que se adapte a las necesidades de los mismos, identificando posibles problemas que entorpecen la labor de realización de las guardias.

Entre las respuestas se han concluido distintos puntos a tratar. La necesidad de poder mantener un control de la disponibilidad de los profesores que se encuentran de guardia. Además de mantener un registro de las horas de guardia realizadas por cada profesor, poder equilibrar el número de horas de guardia realizadas entre los docentes, de tal manera que sea lo más equitativo posible, y que exista una persistencia de esos datos para poder realizar un registro anual de los mismos.

2. Tecnologías y Herramientas utilizadas

Para el desarrollo de TeacherCover se han utilizado varias tecnologías y herramientas.

2.1. Control de versiones y desarrollo de aplicación

Git

En primer lugar, este es un proyecto realizado por dos personas, para facilitar esta tarea y tener una comunicación más sencilla y trabajar de forma conjunta, se ha usado GitHub / Github Desktop. Esta es una plataforma de desarrollo colaborativo, donde se pueden alojar proyectos en repositorios utilizando el sistema de **control de versiones** Git. Con esto podemos tener un log organizado de las nuevas versiones, de las actualizaciones y poder unificar el código de cada integrante de manera cómoda y fácil.

IDE

Para desarrollar la aplicación web y codificar, hemos usado Visual Studio como entorno de desarrollo integrado (**IDE**). Visual Studio es una poderosa herramienta de desarrollo creada por Microsoft que ofrece un conjunto completo de características y herramientas para desarrollar, depurar y mantener aplicaciones. Con Visual Studio, existen varias características como la depuración en tiempo real, el resultado de sintaxis y la integración con Git (ya mencionada antes) para trabajar de manera más eficiente.

2.2. Front-end

Angular

TeacherCover usa Angular como **framework** de desarrollo **front-end**. Angular es una plataforma de desarrollo web de código abierto creada por Google que se utiliza para construir aplicaciones web de una sola página complejas y escalables.
Angular ayuda a crear una aplicación web rápida y eficiente que proporciona una experiencia de usuario fluida y dinámica.
Angular cuenta con un amplio conjunto de características y herramientas, incluyendo el **enlace de datos bidireccional**, la **inyección de dependencias**, las directivas personalizadas y los servicios. Además, Angular dispone de una gran cantidad de documentación y recursos en línea muy actualizados.

TypeScript

TypeScript es un lenguaje de programación que se basa en JavaScript. El framework Angular trabaja con TypeScript, este es un superset de JavaScript, lo que significa que añade nuevas características y mejoras a la sintaxis. Este permite utilizar características de programación orientada a objetos (POO), como clases y herencia. Otro punto muy

importante es que se puede compilar a JavaScript, eso significa que cualquier navegador web o plataforma que soporte JavaScript también soportará TypeScript.

Bootstrap

Bootstrap es un framework para la realización del front-end con varias bibliotecas multiplataforma y un conjunto de herramientas de código abierto para el diseño de sitios web y aplicaciones web. Éste se emplea en TeacherCover para crear **interfaces de usuario (UI)** y **experiencias de usuario (UX)** consistentes. El framework se combina con CSS y JavaScript para estilizar los elementos de una página HTML. Esta es una herramienta que proporciona interactividad en la página, por lo que ofrece una serie de componentes que facilitan la comunicación con el usuario, como menús de navegación, controles de página, barras de proceso y más. Uno de sus principales objetivos en esta aplicación es permitir la construcción del sitio web responsive adaptable a distintos dispositivos..

HTML5

HTML5 (Hypertext Markup Language) es el lenguaje de marcas utilizado para crear y diseñar páginas web. Esta es una herramienta indispensable y esencial para el diseño y desarrollo de la aplicación web. Es altamente compatible con otras tecnologías que también se usan en conjunto en la aplicación como CSS y JavaScript, será además esencial su combinación con Bootstrap.

CSS3

CSS3 es la última versión de CSS (Cascading Style Sheets), un lenguaje utilizado para definir la presentación y diseño de páginas/aplicaciones web. Este proporciona la posibilidad de crear estilos y diseños únicos altamente personalizados modificando todos los elementos del html para crear la imagen de marca del sitio o aplicación.

2.3. Back-end

Firebase

Firebase será la plataforma de desarrollo de **back-end**. Firebase es una plataforma en la nube creada por Google que proporciona una variedad de servicios para el desarrollo de aplicaciones, incluyendo almacenamiento en la nube, autenticación de usuarios, bases de datos en tiempo real y notificaciones push. Utilizando Firebase se logra desarrollar la aplicación y escalarla de manera rápida y sencilla, sin preocuparse por la infraestructura subyacente, centrándose en la lógica e interfaz de ésta teniendo la parte del back con Firebase.

Hosting

Firebase Hosting es un servicio de alojamiento de sitios web y aplicaciones web que forma parte de la plataforma de Firebase. Firebase Hosting ofrece un alojamiento seguro, rápido y escalable para sitios web y aplicaciones web, lo que significa que no hay necesidad de configurar un servidor web ni preocuparse por la infraestructura del alojamiento.
Firebase Hosting ofrece una amplia variedad de características y funcionalidades, incluyendo soporte para SSL (Secure Sockets Layer), alojamiento de dominios personalizados, direccionamiento y control de versiones.

Base de Datos

Firebase Firestore es una base de datos en tiempo real de NoSQL que forma parte de la plataforma de Firebase. Firestore permite a los desarrolladores almacenar y sincronizar datos en tiempo real entre los clientes y el servidor, lo que lo convierte en una opción ideal para aplicaciones web y móviles que requieren datos actualizados en tiempo real.
Firestore utiliza documentos y colecciones para almacenar los datos, lo que hace que la estructura de la base de datos sea más flexible y escalable que otras bases de datos en tiempo real. Firestore también ofrece una amplia variedad de características y funcionalidades, como consultas y filtros avanzados.

3. Estimación de Recursos y Planificación

Para la realización del proyecto se trabajará con una metodología Agile basada en Scrum en la que se harán pequeñas revisiones semanales del trabajo realizado y de aquello que quede por realizar desgranando el mismo en pequeñas tareas asumibles. Al tratarse de una metodología ágil todo puede sufrir modificaciones durante el proceso y el objetivo principal es obtener un producto mínimo viable al que ir añadiendo mejoras y modificaciones según sea necesario.

Los recursos software que se requieren para desarrollar la aplicación son los siguientes:

- Visual Studio Code como entorno de desarrollo
- Angular y TypeScript para la integración del Front
- Firebase para Back-end, hosting y base de datos en tiempo real
- HTML5, CSS3 y Bootstrap para Front-end
- Git y GitHub para control de versiones y repositorio en la nube
- Canva para realización de logo y recursos visuales
- Trello como herramienta de repartición de tareas
- Figma para realización de prototipos

El trabajo será realizado por un equipo de 2 integrantes.

Para la planificación temporal se ha realizado un cronograma (Figura 1) en la que se indican de forma orientativa un reparto de tareas por cada semana del proyecto.

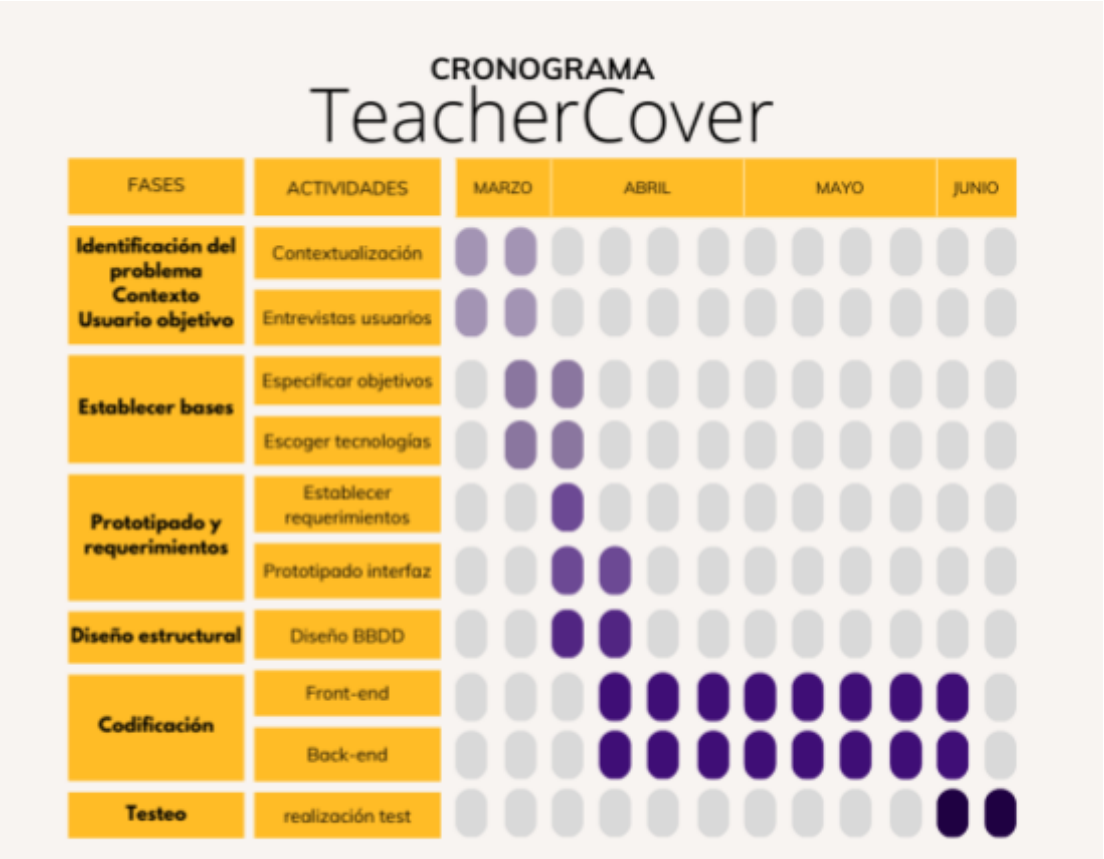


Figura 1. Cronograma proyecto TeacherCover.

Creado con Canva

La planificación se establece en 60 horas totales de realización de proyecto (30 horas por cada integrante del equipo de proyecto) repartidas en un total de 12 semanas.

La repartición de las tareas entre los integrantes del equipo se realizará con Trello, herramienta que permite visualmente llevar un control de las tareas que se están realizando y aquellas que quedan por realizar.

4. Desarrollo: Análisis (Requisitos)

Los requisitos de la aplicación se dividen en 2 tipos de perfiles de **login**, cada uno de ellos tendrá unas funcionalidades distintas:

4.1. Perfil Administrador

Los usuarios con perfil de Administración podrán incluir horarios semanales de guardia en el que se especifican las guardias de todos los docentes, moralmente se realizará una vez al inicio del curso salvo que existan modificaciones.

Registrar una guardia en el día. Cuando un docente notifique su ausencia se podrá incluir para que aparezca en el parte de guardias del día, se podrá incluir una guardia de una única hora o añadir todas las de un docente en el día.

Se podrá dar de alta y de baja a usuarios.

4.2. Perfil Usuario

Los usuarios con Perfil de Usuario podrán realizar acciones desde el login en la aplicación.

Acceso a su perfil de usuario en la aplicación, el cual permite:

- Consulta de su horario de guardia
- Consulta de sus guardias realizadas
- Adjuntar una observación o incidencia a una guardia realizada
- Consultar listado de las guardias del día por horas
- Fichar como que está presente en el centro en su hora de guardia
- Aceptar la guardia que se le notifica (cambia estado de pendiente a adjudicada)
- Confirmar la realización de la guardia (cambia estado de adjudicada a en progreso)
- Confirmar la guardia realizada (cambia estado de en progreso a realizada)
- Consultar el número de horas de guardia realizada (semanal, mensual, curso lectivo)

5. Análisis del modelo

5.1. Diagrama de Casos de Uso

En el diagrama de casos de uso (Figura 2) se indican todas las acciones identificadas en el apartado de requisitos de una forma visual. Se han identificado relaciones entre algunas de las acciones posibles. En el caso del perfil de usuario se observa que la acción de “Confirmar guardia” extiende de la acción de “Aceptar guardia”, ésto quiere decir que las primera sólo podrá realizarse como condición de haber realizado la segunda. Lo mismo ocurre con “Finalizar guardia” que sólo será una opción disponible trás haber confirmado la guardia.
Se ha identificado también una relación de inclusión en el caso de “Añadir incidencia” ya que para poder realizarla será necesario realizar una búsqueda de las guardias para seleccionar sobre qué guardia se indica la incidencia.
En el perfil de Administrador se indican tres relaciones de inclusión. Para poder borrar un usuario será necesario buscarlo anteriormente. Lo mismo ocurre con las guardias, para poder “Modificar guardia” o “Eliminar guardia” será necesario realizar una búsqueda de la misma.

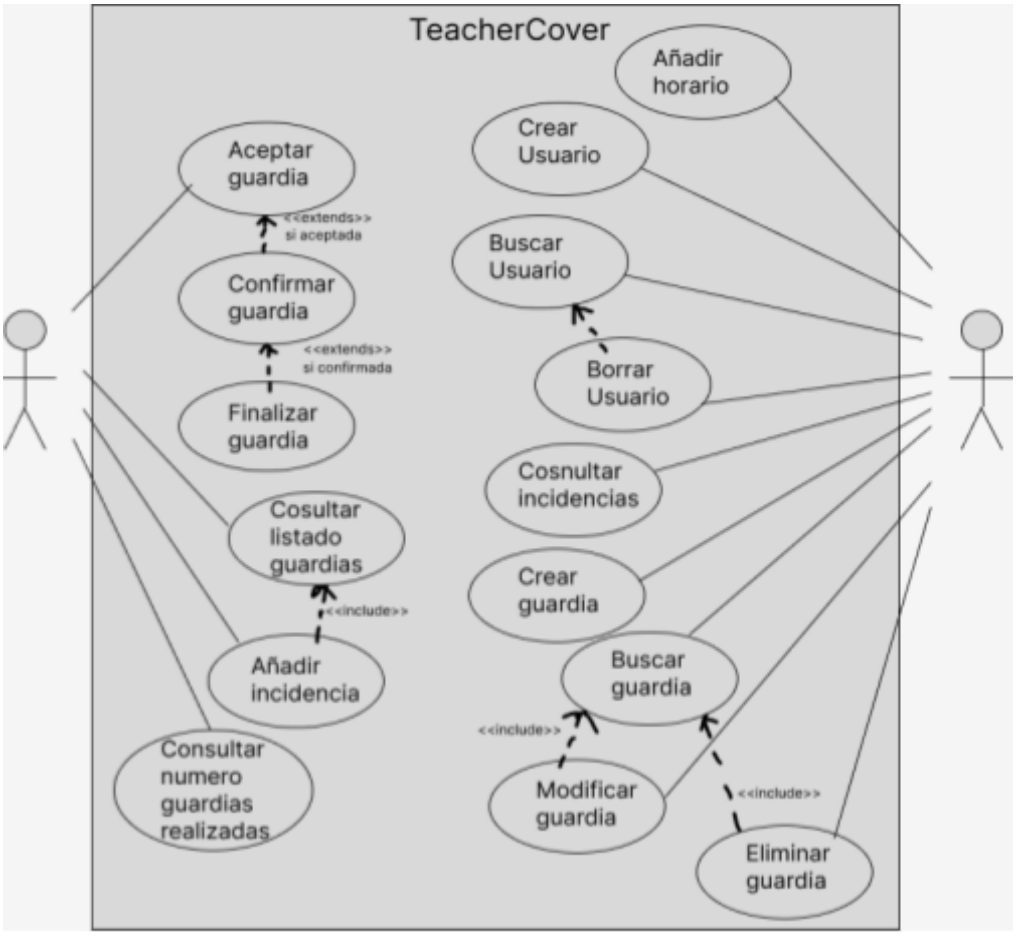


Figura 2. Diagrama de Casos de Uso de TeacherCover

Creado con Figma

Nombre: Aceptar guardia
ID: CU-U01

Descripción:

Un usuario profesor en su hora de guardia o hora de guardia de apoyo recibirá el aviso de las guardias disponibles y podrá aceptarla

Actores: Usuario profesor

Precondiciones:

- El usuario profesor deberá estar logueado en la aplicación
- El usuario profesor debe tener dadas de alta en la base de datos sus horas de guardia
- Debe coincidir la franja horaria en la que el profesor tenga guardia y que existan guardias a realizar en dicha franja horaria para que el profesor pueda ver las disponibles y seleccionar una guardia

Curso normal del caso de uso:

Al llegar a la franja horaria el profesor visualizará las guardias que hay que realizar y podrá seleccionar una

Postcondiciones:

Para poder asignar una guardia como realizada a un profesor éste deberá posteriormente marcar el estado de la guardia como confirmada y finalizada

Nombre: Confirmar Guardia
ID: CU-U02

Descripción:

Un usuario profesor al aceptar una guardia y llegar a esta, tendrá la opción de marcarla como confirmada, para confirmar que se está realizando y se encuentra en el aula.

Actores: Usuario profesor

Precondiciones:

- El usuario profesor deberá estar logueado en la aplicación
- El profesor debe de haber aceptado una guardia anteriormente.
- El profesor debe encontrarse en el aula para confirmar que se encuentra realizando la guardia.

Curso normal del caso de uso:

1 -Al llegar al aula el profesor tendrá la opción de confirmar que está realizando la guardia que acepto.

Postcondiciones:

Para poder asignar una guardia como realizada a un profesor éste deberá posteriormente marcar el estado de la guardia como finalizada

Nombre: Finalizar guardia
ID: CU-U03

Descripción:

Un usuario profesor al finalizar su hora de guardia podrá marcarla como finalizada

Actores: Usuario profesor

Precondiciones:

- El usuario profesor deberá estar logueado en la aplicación
- El usuario profesor deberá previamente haber aceptado y confirmado una guardia
- El tiempo de la hora de guardia deberá haber finalizado

Curso normal del caso de uso:

Al finalizar la franja horaria en la que el profesor ha realizado una guardia aparecerá una opción de marcar como finalizada una guardia

Postcondiciones:

Ninguna

Nombre: Consultar listado de guardias
ID: CU-U04

Descripción:

Un usuario podrá consultar sus horas de guardia realizadas

Actores: Usuario profesor

Precondiciones:

- El usuario profesor deberá estar logueado en la aplicación - El usuario profesor debe tener dadas de alta en la base de datos sus horas de guardia
Curso normal del caso de uso: Al loguearse en la aplicación el usuario podrá consultar un apartado en el que se detalla un listado de las guardias realizadas
Postcondiciones: Ninguna

Nombre: Añadir incidencia ID: CU-U05
Descripción: Un usuario podrá añadir incidencias en las guardias realizadas
Actores: Usuario profesor
Precondiciones: - El usuario profesor deberá estar logueado en la aplicación - El usuario profesor debe tener dadas de alta en la base de datos sus horas de guardia - El usuario profesor deberá haber aceptado, confirmado y finalizado una guardia
Curso normal del caso de uso: El usuario deberá consultar el listado de sus guardias y editar el campo de descripción de incidencia
Postcondiciones: Ninguna

Nombre: Consultar listado de guardias ID: CU-U06
Descripción: Un usuario podrá consultar sus horas de guardia realizadas
Actores: Usuario profesor
Precondiciones: - El usuario profesor deberá estar logueado en la aplicación

- El usuario profesor debe tener dadas de alta en la base de datos sus horas de guardia
Curso normal del caso de uso: Al loguearse en la aplicación el usuario podrá ver en el panel de información las horas de guardia realizadas
Postcondiciones: Ninguna

Nombre: Añadir horario ID: CU-A01
Descripción: Un administrador podrá añadir los horarios de los profesores
Actores: Usuario administrador
Precondiciones: - El usuario administrador deberá estar logueado en la aplicación - El usuario profesor del que se añada el horario deberá estar creado
Curso normal del caso de uso: El administrador podrá registrar tanto horas de clase, horas de guardia y guardias de apoyo de un profesor
Postcondiciones: Ninguna

Nombre: Crear usuario ID: CU-A02
Descripción: Un administrador podrá crear otros usuarios
Actores: Usuario administrador
Precondiciones: - El usuario administrador deberá estar logueado en la aplicación

Curso normal del caso de uso: El administrador podrá registrar tanto otros usuarios administradores como usuarios profesores
Postcondiciones: Ninguna

Nombre: Buscar usuario ID: CU-A03
Descripción: Un administrador podrá buscar usuarios
Actores: Usuario administrador
Precondiciones: - El usuario administrador deberá estar logueado en la aplicación - El usuario al que busque deberá estar dado de alta en la base de datos
Curso normal del caso de uso: El administrador podrá visualizar un listado con todos los usuarios, podrá filtrar por los distintos campos para facilitar la búsqueda
Postcondiciones: Ninguna

Nombre: Editar usuario ID: CU-A04
Descripción: Un administrador podrá editar usuarios
Actores: Usuario administrador
Precondiciones: - El usuario administrador deberá estar logueado en la aplicación - El usuario al que edite deberá estar dado de alta en la base de datos - El usuario administrador deberá seleccionar en la tabla de usuarios el usuario a editar

Curso normal del caso de uso: El administrador una vez seleccionado el usuario a editar podrá modificar los campos que considere necesarios salvo el email
Postcondiciones: Ninguna

Nombre: Borrar usuario ID: CU-A05
Descripción: Un administrador podrá borrar usuarios profesor
Actores: Usuario administrador
Precondiciones: - El usuario administrador deberá estar logueado en la aplicación - El usuario profesor al que borre deberá estar dado de alta en la base de datos
Curso normal del caso de uso: El administrador podrá seleccionar la opción de eliminar usuario en el listado de usuarios siempre que el usuario a borrar no sea administrador
Postcondiciones: Confirmar la advertencia de acción irreversible

Nombre: Consultar incidencias ID: CU-A06
Descripción: Un administrador podrá consultar un listado de las incidencias
Actores: Usuario administrador
Precondiciones: - El usuario administrador deberá estar logueado en la aplicación - Deberá haber registradas incidencias en la base de datos
Curso normal del caso de uso: El administrador podrá visualizar una tabla con todas las incidencias

relacionadas con la guardia en la que ocurrió la incidencia
Postcondiciones: Ninguna

Nombre: Crear guardia ID: CU-A07
Descripción: Un administrador podrá crear guardias
Actores: Usuario administrador
Precondiciones: - El usuario administrador deberá estar logueado en la aplicación
Curso normal del caso de uso: El administrador podrá crear una nueva guardia
Postcondiciones: Ninguna

Nombre: Buscar guardia ID: CU-A08
Descripción: Un administrador podrá buscar una guardia
Actores: Usuario administrador
Precondiciones: - El usuario administrador deberá estar logueado en la aplicación - La guardia buscada deberá existir en la base de datos
Curso normal del caso de uso: El administrador podrá crear buscar en el listado de guardias una guardia mediante el filtrado de campos
Postcondiciones: Ninguna

Nombre: Modificar guardia
ID: CU-A09

Descripción:
Un administrador podrá modificar una guardia

Actores: Usuario administrador

Precondiciones:
- El usuario administrador deberá estar logueado en la aplicación
- La guardia a modificar deberá existir en la base de datos
- Deberá seleccionar la guardia a modificar desde el listado de las guardias

Curso normal del caso de uso:
El administrador una vez seleccionada una guardia podrá modificar sus campos

Postcondiciones:
Deberá aceptar el aviso de modificar campos

Nombre: Eliminar guardia
ID: CU-A10

Descripción:

Un administrador podrá eliminar una guardia

Actores: Usuario administrador

Precondiciones:

- El usuario administrador deberá estar logueado en la aplicación
- La guardia a modificar deberá existir en la base de datos
- Deberá seleccionar la guardia a eliminar desde el listado de las guardias

Curso normal del caso de uso:

El administrador una vez seleccionada una guardia podrá eliminarla

Postcondiciones:

Deberá aceptar el aviso cambios irreversibles

5.2. Diagrama de Clases

Un diagrama de clases es una herramienta de modelado que se utiliza en el desarrollo de software para representar de forma visual las clases, interfaces y sus relaciones en un sistema.

La función del diagrama de clases en el desarrollo de software es proporcionar una visión general de la estructura del sistema, ayudando a los desarrolladores a comprender la estructura del sistema, sus clases y cómo se relacionan entre sí.

En este diagrama (Figura 3), las clases son representadas por rectángulos que contienen el nombre de la clase, sus atributos y métodos. Las relaciones entre las clases se representan mediante líneas que conectan las clases, indicando la naturaleza de la relación.

Las clases que se relacionan por líneas que acaban en flecha hacen referencia a la herencia. En éste caso la superclase User tiene dos subclases Usuario y Admin.

Además en ésta herencia podemos observar que la superclase User se encuentra representada en cursiva para indicar que es una **clase abstracta** la cual no se podrá instanciar.

El resto de relaciones que no tienen flecha en los extremos de la línea que las une representan relaciones de asociación las cuales vienen indicadas con una cardinalidad para cada caso.

La cardinalidad en un diagrama de clases hace referencia a la cantidad de objetos que pueden estar relacionados en una asociación. Se representa mediante símbolos y números en las líneas que conectan las clases en el diagrama.

Aquellas en las que relaciones especificadas con 1 indica que la instancia de la clase que se encuentra en el lado opuesto estará relacionada con una única instancia de la otra clase.Por ejemplo, en el diagrama una notificación podrá ser registrada por un único usuario.

La cardinalidad indicada con 0..1 hace referencia a que la instancia de la clase que se encuentra en el lado opuesto estará relacionada con cero o una instancia de la otra clase. Como ejemplo en la figura se muestra en la relación entre Usuario y guardia que una guardia puede estar asignada a cero o un usuario.

Por último aquellas relaciones representadas como 0..* indican que la instancia de la clase que se encuentra en el lado opuesto estará relacionada con cero o muchas instancias de la otra clase. A modo de ejemplo en la relación entre Guardia y notificación, una Guardia puede tener una o muchas notificaciones.

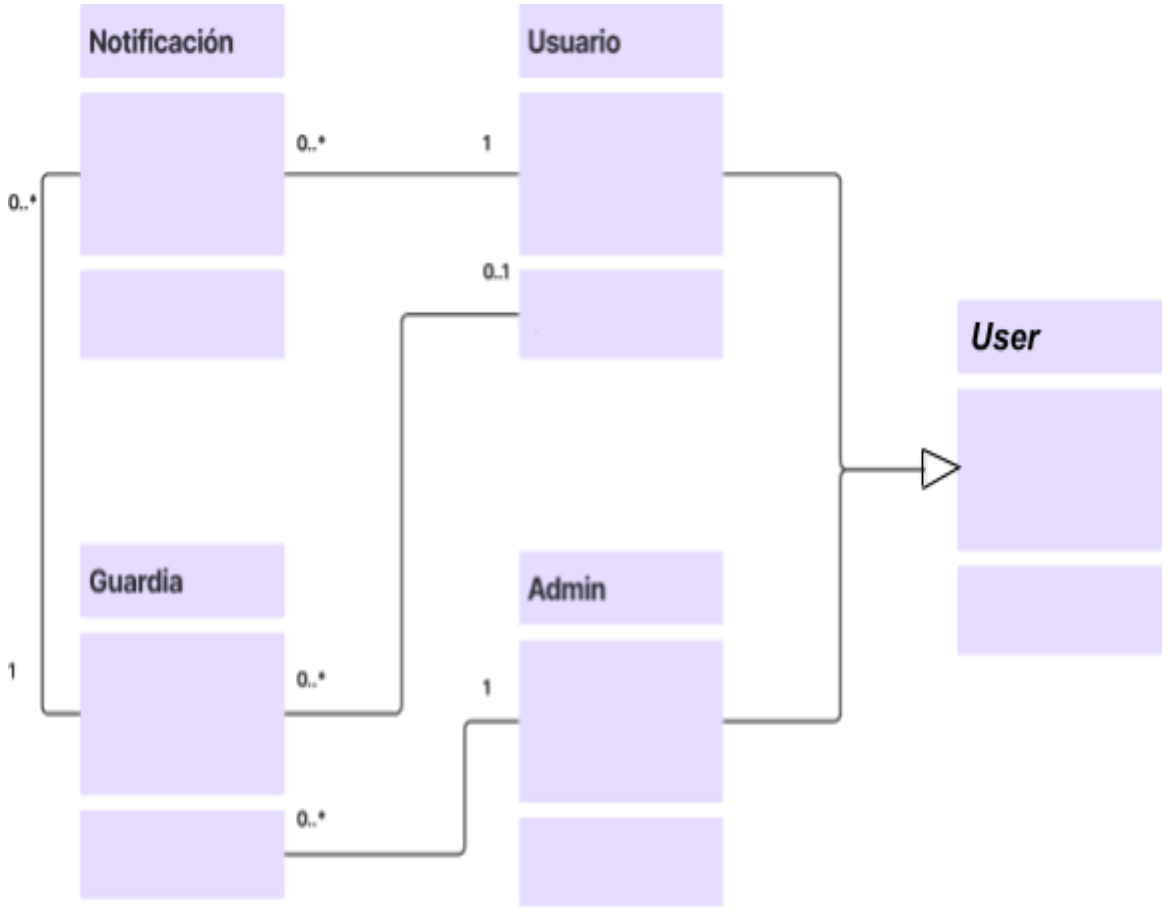


Figura 3. Diagrama de Clases de TeacherCover
Creado con Figma

6. Diseño

6.1. Colores

Un uso adecuado del color es esencial para garantizar una experiencia de usuario coherente y efectiva. Los colores seleccionados como imagen de marca de la aplicación son el morado oscuro y el naranja, los cuales se emplean de manera consistente en toda la interfaz.

El morado oscuro se emplea como color de fondo en elementos principales como la barra de navegación y el **header** y como color de la letra en los botones del menú lateral. También puede se emplea para resaltar elementos importantes como botones de acción, alertas y mensajes de confirmación.

El naranja se emplea como color secundario para resaltar elementos específicos, como botones de llamado a la acción, etiquetas y elementos de navegación activos. También se emplea como color del logo de la aplicación, al igual que para otros iconos.

En todo momento los colores se emplean de forma consciente empleando un contraste lo suficientemente alto para garantizar la **accesibilidad** en la aplicación. Por lo tanto, se escogen tonalidades que se complementen y se puedan distinguir claramente entre sí.

Para garantizar que el contraste es el adecuado se ha realizado un análisis del mismo con la extensión para Google Chrome WAVE Evaluation Tool.

Además de éstos dos colores de identidad de marca se emplean colores menos saturados de la misma gama cromática en otros elementos.

La paleta de color (Imagen 1) es bastante sencilla ya que se pretende no caer en mezclar demasiados colores que puedan distraer al usuario o crear una experiencia desordenada.

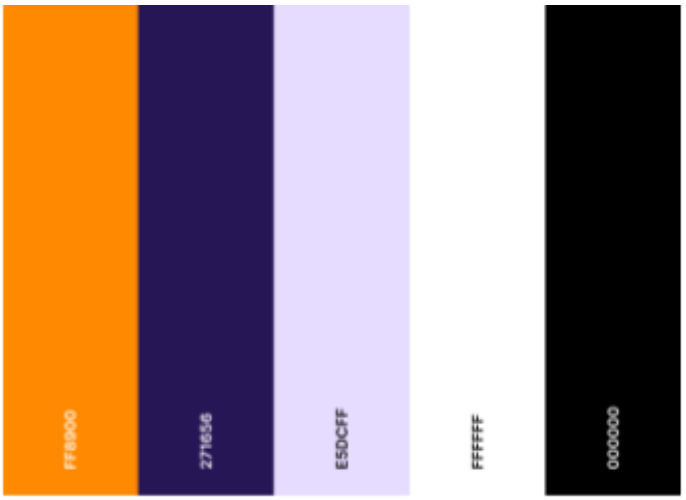


Imagen 1. Paleta de color de TeacherCover
Creada con Coolors

6.2. Diseño de Interfaces

La estructura de la interfaz de la aplicación debe ser clara, intuitiva y fácil de navegar para los usuarios.Es por ello que en las distintas páginas se mantiene una coherencia, para que el usuario reconozca que sigue navegando en el mismo sitio, caracterizada por la sencillez y limpieza en el diseño.

6.2.1. Interfaz de página Login versión escritorio

La interfaz diseñada para la página de login (Imagen 2) es sencilla y limpia. Consiste únicamente en un header con fondo en color morado en el que se encuentra situado el logo en la parte izquierda. En la parte del cuerpo de la página se encuentra como fondo una imagen distorsionada con temática de oficina y sobre ella se sitúa una tarjeta en la que se encuentra el título de la aplicación junto con dos campos de input (usuario y contraseña) y un botón para acceder. Con ésta simplicidad se pretende que el uso sea claro y sencillo para usuarios menos experimentados, además de que aporte limpieza y claridad para el resto de usuarios.

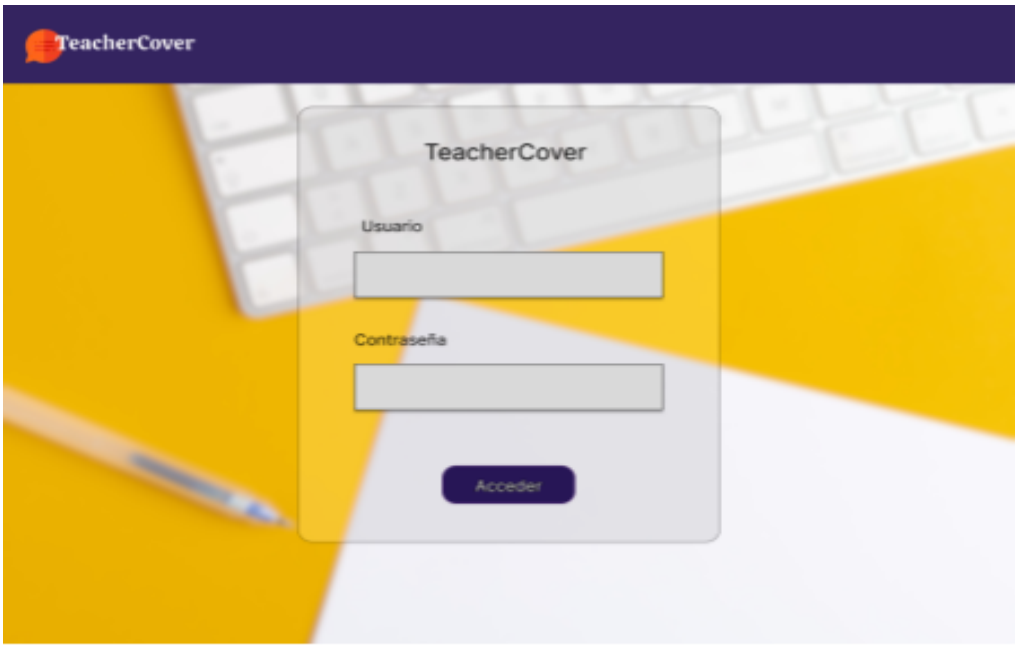


Imagen 2. Interfaz de página login de TeacherCover versión escritorio
Creado con Figma

6.2.2. Interfaz de página Login versión móvil

Al igual que en la versión de escritorio, la versión **responsive** (Imagen 3) cuenta con los mismos componentes pero adaptando el login a una pantalla de móvil permitiendo que los campos de interacción tengan un target lo suficientemente grande para que resulte cómodo su uso.

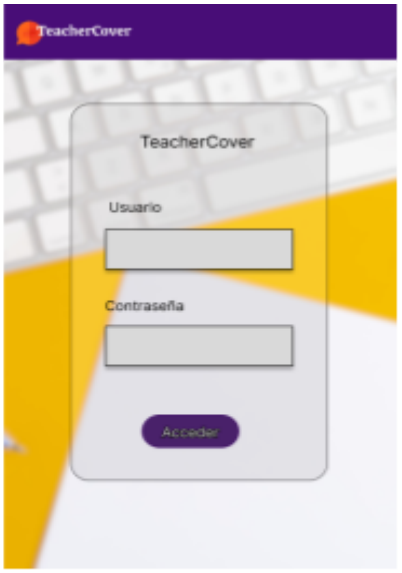


Imagen 3. Interfaz de página login de TeacherCover versión responsive
Creado con Figma

6.2.3. Interfaz de página Principal versión escritorio

Tanto en la versión de escritorio (Imagen 3) como en la versión responsive el resto del sitio consistirá en una página principal “**single page**" en la que según las opciones seleccionadas se mostrará un contenido u otro.

La estructura básica de ésta página consta de tres partes diferenciadas.

1. En primer lugar un header en el que se encontrará el logo de la aplicación a la izquierda y un icono de perfil con un desplegable **dropdown** con las distintas opciones del perfil.

En la parte del cuerpo de la página distinguimos la sección principal y la barra de navegación lateral.

2. En la sección principal se muestra por un lado información relevante como el nombre de perfil de usuario, fecha y avisos de guardias entre otras , y una segunda sección que será la que cambie según la opción del menú seleccionada.

3. La barra de navegación lateral: esta barra se encuentra bien visible en la parte superior de la página y contiene enlaces a las secciones principales de la aplicación, como el calendario de guardias, el registro de profesores y el sistema de notificaciones. En esta encontraremos las distintas opciones que tenga el perfil, distintas para administración y para usuarios, que serán las que modifiquen el contenido de la sección principal.

Las distintas opciones que se muestran en la sección principal según se selecciona dicha opción son:

- Perfil Usuario

1. Calendario de guardias: esta sección muestra de manera clara y ordenada las guardias que ha de realizar el profesor en la semana, pudiendo ser estas: ya realizadas o sin confirmar.
2. Avisos de guardia: en el momento que salga una nueva guardia a realizar, los profesores que tengan acceso a esta podrá seleccionarla, adjudicándosela.
3. Registrar una observación o incidencia a una guardia realizada
4. Fichar como presente en el centro en horario de guardia o suplencia de guardia
5. Ver todas las guardias realizadas por el profesor

- Perfil Administrador

1. Crear cuenta para un nuevo profesor.
2. Observar todas las incidencias reportadas.
3. Ver todas las guardias realizadas
4. Crear una nueva Guardia
5. Añadir horario de guardias

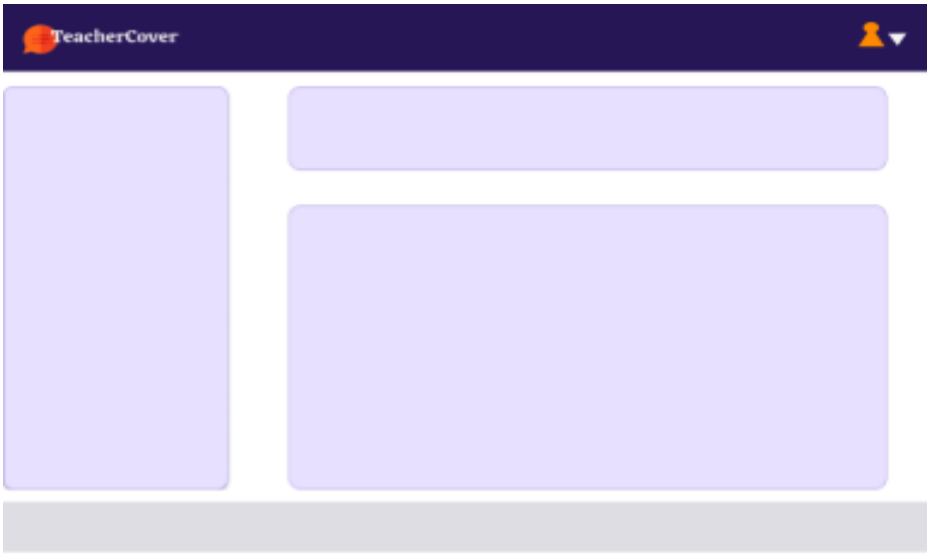


Imagen 4. Interfaz de página principal de TeacherCover versión escritorio
Creado con Figma

6.2.4. Interfaz de página Principal versión móvil

En la página principal, la interfaz cambia para proporcionar una experiencia más cómoda y sencilla al usuario. La modificación principal consiste en reorganizar el **sidebar**, que anteriormente se encontraba en la parte izquierda de la pantalla y tenía una disposición vertical. Ahora en la versión responsive (Imagen 4), el sidebar se ubica en una barra horizontal que se sitúa en primer lugar por encima del contenido, actuando de navbar. La estructura dentro del **navbar** sigue siendo la misma, así como el orden de sus elementos siguiendo la misma lógica. Esto permite que los demás elementos puedan ocupar todo el espacio de la pantalla horizontalmente.

Imagen 5. Interfaz de página principal de TeacherCover versión responsive
Creado con Figma

6.2.5. Interfaz del árbol de proyecto (Angular)

El proyecto TeacherCover sigue el formato de Angular para ordenar todas las clases y elementos de este. Dentro del src tenemos el main.ts, index.html y styles.css que son los archivos principales de todo el proyecto, de ellas dependen todas las demás.

Por debajo tenemos los assets, environments y app, este último es el más importante, ya que es el que almacena la mayor parte del proyecto, aquí tendremos el componente padre (app.component) junto con el app.module que configura todos los componentes.

Podemos ver varias carpetas, cada una de estas es un componente, la cual contiene el archivo ts, el html, y el css correspondiente a cada componente. All mismo nivel tenemos los services, que contiene aquellas clases de typescript encargadas de comunicarse con la base de datos y traerlos a la parte front de la aplicación. Y por último los models, que son las clases orientadas a objetos del proyecto.

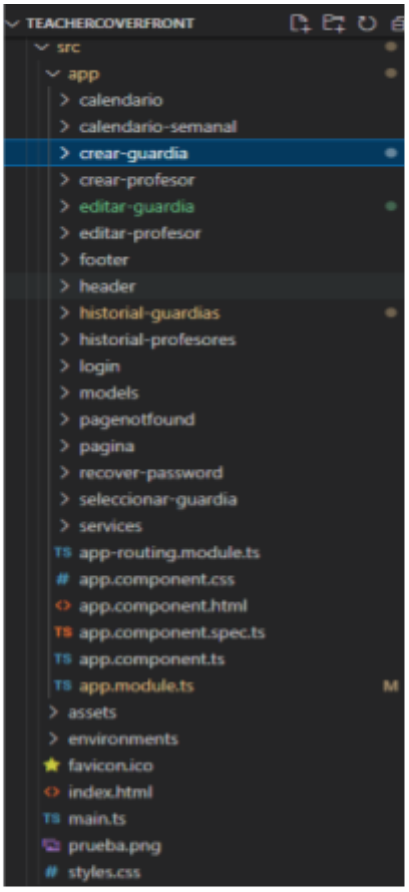


Imagen 6. Estructura de carpetas de la interfaz de TeacherCover
Imagen del IDE

6.3. Modelo de almacenamiento de datos

En la aplicación el almacenamiento de datos se realiza en una Base de Datos NoSQL.

Las bases de datos NoSQL (Not Only SQL) son un tipo de sistema de gestión de bases de datos. A diferencia de las bases de datos relacionales tradicionales, basadas en un modelo de tablas y relaciones, las bases de datos NoSQL utilizan modelos de datos más flexibles y menos estructurados, como documentos o sistemas de clave-valor. Esto les permite manejar una gran variedad de datos, permitiendo a la aplicación una gran escalabilidad y una mayor agilidad y flexibilidad.

En la Figura 5 se muestra cómo se relacionan los distintos documentos de la base de datos de TeacherCover. Estos documentos en una Base de Datos no relacional son la unidad básica de almacenamiento y pueden tener estructuras variables y dinámicas. El formato de los documentos suele ser JSON (JavaScript Object Notation), que es un formato de datos fácilmente legible.

La relación entre los documentos en una base de datos NoSQL se establece mediante la incorporación de datos anidados, es decir, se pueden incluir documentos completos como campos de otros documentos.

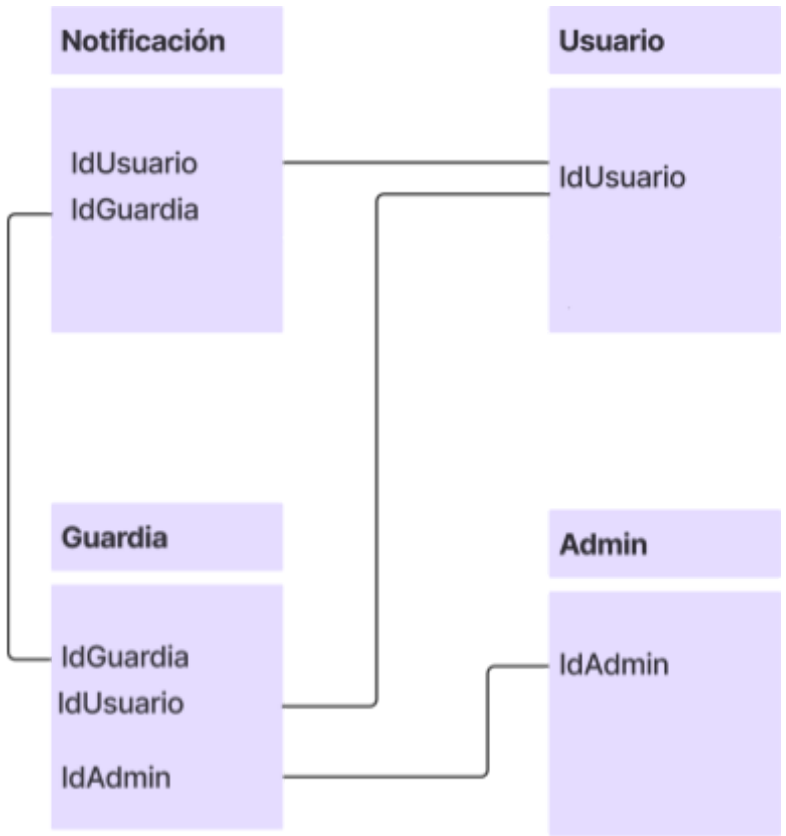


Figura 4. Modelo de almacenamiento de datos de TeacherCover
Creado con Figma

7. Revisión de los requerimientos

Para obtener los requerimientos de la aplicación se realizaron inicialmente una serie de entrevistas de las cuales se obtuvieron las historias de usuario (Figura 6) sobre las que trabajar para construir la aplicación.

Las historias de usuario son una técnica utilizada en el desarrollo de software para describir una funcionalidad del sistema desde la perspectiva del usuario final. Es una parte importante de la **metodología agile**. Estas describen objetivos o requisitos que desean obtener los distintos perfiles de usuarios de una manera simple. Con ellas el equipo de desarrollo define distintas características a tener en cuenta y así no sólo se centra en los requisitos más técnicos. En este caso existen dos perfiles en la columna como, Profesor o Administrador que definen los distintos requisitos.

Las historias de usuario se leen de la siguiente forma: "**Como** *Profesor* **quiero** *una opción* **para** *poder informar de algún suceso durante la guardia*"

COMO	QUIERO	PARA
Profesor	poder logearme	acceder a mi horario de guardias
Administrador	una apartado	poder crear y modificar las guardias
Profesor	una opción	aceptar guardias del día
Profesor	un calendario	ver las guardias realizadas el último mes
Administrador	la opción	crear usuarios de la aplicación
Profesor	poder consultar las guardias realizadas	llevar un registro
Profesor	una opción	poder informar de algún suceso durante una guardia
Profesor	un elemento	Qué me informe si tengo notificaciones nuevas

Figura 5. Historias de Usuario de TeacherCover
Elaboración propia

De las historias de usuario se obtiene el backlog con el cual se definen a grandes rasgos las tareas a realizar en el proyecto.

El backlog del proyecto es una lista priorizada de elementos en los que debe trabajar el equipo de desarrollo para lograr los objetivos definidos en el proyecto. Gracias a la priorización de tareas del backlog el equipo de desarrollo puede así trabajar primero en aquellas tareas que sean de mayor importancia para conseguir que el proyecto cumpla antes con estas características y obtener así lo antes posible el producto mínimo viable.

En la figura 7 se muestran las tareas que forman el backlog sacadas de las historias de usuario de la figura 6.

Las tareas se encuentran ordenadas por prioridad, siendo la primera de la lista la más urgente. Además se encuentran catalogadas por estimación de tiempo siguiendo la técnica de estimación T-Shirt Sizes, la cual se basa en asignar tallas de camiseta (XS, S, M L, XL, XXL) a la hora de estimar los tiempos de realización de las tareas en vez de utilizar números, lo cual ayuda a que los miembros de forma rápida y visual puedan identificar aquellas tareas que requieren mayor dedicación de tiempo. Las estimaciones de tiempo se corresponden aproximadamente a XS: 0,5 horas, S: 1 hora, M: 2 horas, L: 3 horas, XL: 6 horas, XXL: 12 horas.

Backlog	
Tarea	Estimación
login de acceso	XL
crear guardias (modificar y eliminar)	XL
crear usuarios (modificarlos y eliminarlos)	XL
realizar avisos de las guardias	XXL
crear calendario para visualizar las guardias	L
consultar listado de guardias	M
informar de un suceso en la guardia	L

Figura 6. Backlog de tareas de TeacherCover

Elaboración propia

8. Plan de Pruebas

Nº prueba	Especificación de las pruebas
1	Objetivo probado: Acceder a la aplicación Requisitos Probados: - Sólo aquellos usuarios con credenciales tendrán acceso a la aplicación Prueba realizada: - Se accede a la aplicación en la página de login y se comprueba que si el usuario y contraseña no están registrados en la base de datos no permite el acceso a la aplicación, en caso contrario, si son correctos dejará continuar a la siguiente pantalla.
2	Objetivo probado: Acceso por primera vez a la aplicación Requisitos Probados: - Al inciar sesión por primera vez se exige un cambio de contraseña Prueba realizada: - Se accede a la aplicación en la página de login y se introducen por primera vez unas credenciales válidas, es decir un correo electrónico registrado en la base de datos y la contraseña, IESinfanta23, otorgada por defecto a todos los usuarios al registrarlos en la base de datos. Una vez introducidos los datos correctamente aparece un input para escribir la nueva contraseña.
3	Objetivo probado: Recuperar contraseña Requisitos Probados: - Se realizan todos los pasos de modificar la contraseña. Sólo los usuarios registrados pueden realizarlos.

	Prueba realizada: - Se realizan los pasos de hacer clic en “¿Has olvidado tu contraseña?”, este redirige a un nuevo formulario que permite rellenar el correo electrónico. Al dar al botón enviar código se recibe un email con un enlace a la página de cambiar contraseña y al resetearla se prueba a acceder con la nueva contraseña de forma satisfactoria. Se comprueba además que al añadir una dirección de correo no incluida en la base de datos, al hacer clic en el botón enviar código se avisa de que no es una cuenta existente.
4	Objetivo probado: Intentar acceder directamente a la página principal Requisitos Probados: - Sólo debe mostrarse la página principal si hay una sesión iniciada Prueba realizada: - Se prueba a copiar la url de la página principal y se intenta acceder sin tener una sesión abierta. La página la rechaza redirigiendo a la página de login.
5	Objetivo probado: Realizar filtrados de columnas en la tabla de historial Requisitos Probados: - Debe dejar realizar varios filtrados a la vez Prueba realizada: - Se realiza la prueba de realizar varios filtrados a la vez en las distintas columnas. Tanto los filtrados de ordenación (ascendente-descendente) como los filtrados por texto se pueden combinar en la búsqueda.
	Objetivo probado: Calendario día actual dinámico Requisitos Probados:

6	- Debe actualizarse y pasar de franja horaria avisando al profesor de cuando tenga una hora de guardia Prueba realizada: - La aplicación cada vez que llega un cambio de hora se actualiza marcando la nueva franja horaria en la que se encuentra el profesor, y dependiendo de si tiene una guardia o no se le notifica al usuario con un mensaje personalizado
7	Objetivo probado: Crear profesor Requisitos Probados: - Debe crearse un profesor nuevo con los datos introducidos por un usuario administrador Prueba realizada: - Desde el perfil de un usuario administrador se accede a la opción “Crear profesor” y permite introducir los datos de un nuevo profesor, al terminar el registro este se añade a la base de datos.
8	Objetivo probado: Editar profesor Requisitos Probados: - Al modificar los datos de un profesor estos deben actualizarse en la base de datos Prueba realizada: - Se prueba a cambiar los datos de un profesor existente y a visualizarlo para comprobar que efectivamente se han realizado los cambios y ahora contiene los nuevos.
	Objetivo probado: Eliminar profesor Requisitos Probados: - Al eliminar un profesor este desaparece de la tabla del historial de profesores

9	Prueba realizada: - Se prueba a eliminar un profesor y comprobar que este ya no está en la base de datos ni sale en la tabla de todos los profesores. Antes de realizar la operación se pide una confirmación
10	Objetivo probado: Crear guardia Requisitos Probados: - Comprobar que se crea una nueva guardia con los datos introducidos por un usuario administrador Prueba realizada: - Desde el perfil de un usuario administrador se accede a la opción “Crear guardia” y permite introducir los datos de una nueva guardia, al crearla de manera exitosa esta se añade a la base de datos.
11	Objetivo probado: Editar guardia Requisitos Probados: - Al modificar los datos de una guardia estos deben actualizarse en la base de datos Prueba realizada: - Se prueba a cambiar los datos de una guardia existente y a visualizarlos para comprobar que efectivamente se han realizado los cambios y ahora contiene los nuevos.
12	Objetivo probado: Eliminar guardia Requisitos Probados: - Al eliminar una guardia esta desaparece de la base de datos Prueba realizada:

	- Se prueba a eliminar una guardia y comprobar que esta ya no está en la base de datos. Sólo podrán eliminarse aquellas que no hayan finalizado.
13	Objetivo probado: Seleccionar guardia Requisitos Probados: - Permite al profesor entrar y seleccionar una guardia para realizarla Prueba realizada: - Se prueba en una hora de guardia normal o de apoyo a intentar seleccionar una guardia, pudiendo sólo elegir las disponibles, y solo una por profesor

8.1. Plan de pruebas extendido

Nº prueba	Especificación de las pruebas
1	Objetivo probado: Acceder a la aplicación sin rellenar campos Requisitos Probados: - Al intentar realizar un login sin introducir los campos, se ejecuta la validación de formulario mostrándole al usuario que son campos obligatorios sin dejarle continuar. Prueba realizada: - Se intenta un acceso a la aplicación y se comprueba que efectivamente no deja continuar avisando al usuario de que hay que rellenar los campos obligatorios.
2	Objetivo probado: Acceder a la aplicación datos con formato incorrecto Requisitos Probados: - Introducir datos con el formato incorrecto al introducir los datos de ingreso, mostrando un mensaje para que el usuario sea conocedor de ello. Prueba realizada: - Se intenta un acceso a la aplicación y se comprueba que efectivamente no deja continuar avisando al usuario de que hay datos con un formato incorrecto, esto se comprueba mediante el uso de una **expresión regular**, la cual tiene el formato del correo electrónico correcta y comprueba el introducido
	Objetivo probado: Acceder a la aplicación con credenciales no válidas Requisitos Probados:

Nº prueba	Especificación de las pruebas
3	- Introducir datos para realizar un login y que estos no coincidan con ninguna cuenta, se notifica con un mensaje mostrando este problema. Prueba realizada: - Se intenta un acceso a la aplicación y se comprueba que efectivamente no deja continuar avisando al usuario de que no hay ninguna cuenta que cumpla con esos datos.
4	Objetivo probado: Recuperar contraseña Requisitos Probados: - Al solicitar una recuperación de contraseña con un correo inválido o que no exista en la base de datos, se le muestra al usuario un mensaje tipo **toast** Prueba realizada: - Se realiza una prueba de recuperar contraseña introduciendo el correo electrónico con un formato incorrecto, y también vacío, viendo como se muestra la notificación.
5	Objetivo probado: Crear guardia formulario Requisitos Probados: - Todos los campos del formulario son obligatorios y se comprueban antes de realizar la operación de introducir una guardia Prueba realizada: - Al intentar crear una guardia con campos vacíos se muestra al usuario un mensaje de error sin dejarle continuar en el proceso.
	Objetivo probado: Editar guardia formulario

Nº prueba	Especificación de las pruebas
6	Requisitos Probados: - Al editar una guardia salen todos los datos de esta ya introducidos en el formulario y permite cambiar estos, comprobando que los nuevos datos son válidos. Prueba realizada: - Al comprobar editar una guardia se ven todos los datos de la propia guardia y permite cambiarlos validando que estos son correctos.
7	Objetivo probado: Crear profesor formulario Requisitos Probados: - Todos los campos del formulario son obligatorios y se comprueban antes de realizar la operación de introducir un nuevo profesor Prueba realizada: - Al intentar crear un profesor con campos vacíos se muestra al usuario un mensaje de error sin dejarle continuar en el proceso.
8	Objetivo probado: Editar profesor formulario Requisitos Probados: - Al editar un profesor salen todos los datos de esta ya introducidos en el formulario y permite cambiar estos, comprobando que los nuevos datos son válidos. Prueba realizada: - Al comprobar editar un profesor se ven todos los datos de la propia guardia y permite cambiarlos validando que estos son correctos

9. Conclusiones

9.1. Objetivos alcanzados

Se ha alcanzado el objetivo de la implementación y el aprendizaje de un nuevo framework para la realización de la parte front-end del proyecto. En este caso el framework Angular.

También ha sido posible la integración de firebase para todos los procesos back-end del proyecto. Con este se ha logrado obtener un alojamiento para permitir que la página se encuentre en internet y no ser así un proyecto únicamente en local.

Con la incorporación de firebase también se incorporó la firestore database lo que ha supuesto aprender a trabajar con bases de datos no relacionales.

En cuanto a los objetivos propuestos propios de la aplicación se han conseguido implementar ambos perfiles, usuario y administrador con sus distintas opciones de menú.

Se ha conseguido que la actualización se realice en tiempo real y que permita así al entrar en una franja horaria que salgan los avisos relativos a la misma.

Se ha integrado tanto opción de guardia como guardia de apoyo con su distinta lógica de asignación de guardias.

También ha sido posible implementar la opción de añadir incidencias a las guardias y que el administrador pueda visualizarlas en tiempo real.

La aplicación web es totalmente responsive lo que permite por tanto su uso desde cualquier tipo de dispositivo.

Por otro lado a destacar como dificultad en la realización del proyecto estaría el realizar la aplicación con una base de datos no relacional ya que, quizá por desconocimiento, ha dificultado las consultas y por tanto limitado las posibilidades de funcionalidad de la aplicación.

9.2. Conclusiones del trabajo

En cuanto al aprendizaje, la realización del proyecto ha resultado muy enriquecedora ya que ha permitido emplear nuevas tecnologías, lo cual ha permitido poder aprender a trabajar con los frameworks Angular y Firebase.

Con la incorporación de firebase también se incorporó la “firestore database” lo que ha supuesto un aprendizaje extra al trabajar con bases de datos no relacionales.

9.3. Vías futuras

Como vías futuras, para la ampliación y mejora del proyecto, se sugieren las siguientes opciones:

1. Crear incidencias de las guardias:

    - Implementar un sistema mejorado que permita a los usuarios reportar incidencias relacionadas con las guardias. Esto podría incluir la posibilidad de agregar descripciones detalladas, permitir adjuntar imágenes u otros archivos relevantes, asignar prioridades a las incidencias y realizar un seguimiento del estado de su resolución a los administradores.

    - Diseñar una interfaz intuitiva y fácil de usar para que los usuarios puedan registrar incidencias de forma rápida y sencilla. Esto podría incluir la capacidad de agregar comentarios o actualizaciones adicionales a medida que se resuelven las incidencias.

2. Realizar notificaciones utilizando Firebase Cloud Messaging (FCM) en navegadores web:

    - Integrar FCM en la aplicación web para enviar notificaciones push a los usuarios. Esto permitiría enviar mensajes instantáneos y relevantes sobre las guardias, como cambios en los horarios, recordatorios de horas de guardia o actualizaciones importantes.

    - Configurar la lógica de backend para enviar notificaciones push a través de FCM cuando ocurran eventos específicos relacionados con las guardias. Por ejemplo, enviar una notificación cuando se asigne una nueva guardia, se cambie un turno o se resuelva una incidencia reportada.

3. Integrar el **Bot** de Telegram para gestionar las guardias:

    - Emplear el bot de Telegram permitiendo a los usuarios interactuar y gestionar las guardias a través de la plataforma de mensajería. Esto podría incluir funciones como la solicitud de información sobre las próximas guardias, la consulta de horas de guardia realizadas o la recepción de recordatorios personalizados.
    - Configurar la integración entre el Bot de Telegram y el sistema existente de gestión de guardias. Esto implicaría establecer una comunicación bidireccional para recibir comandos o mensajes del Bot y procesarlos en el backend, así como enviar respuestas o actualizaciones relevantes de vuelta al Bot y a los usuarios.

4. Permitir que los administradores creen nuevas guardias desde el calendario mediante la selección de franjas horarias correspondientes:
    - Implementar una funcionalidad en el calendario que permita a los administradores crear nuevas guardias haciendo clic en la franja de hora deseada.
    - Al hacer clic en una franja horaria, se abrirá un formulario donde se podrán ingresar los detalles de la guardia, como el profesor a cubrir, el aula o información relevante para la guardia.

- Validar los datos ingresados para garantizar la integridad de la información y evitar conflictos en los horarios existentes.

5. Implementar la creación automática de guardias por parte de los administradores basada en los horarios predefinidos de los profesores en la base de datos:

    - Establecer un sistema en la base de datos que contenga los horarios de clase y guardias de los profesores.
    - Al recibir una notificación de ausencia de un profesor para un día específico, permitir al administrador seleccionar al profesor afectado y el día en cuestión.
    - Utilizando la información del horario del profesor seleccionado, generar automáticamente las guardias correspondientes según su horario. Esto podría incluir opciones como generar las guardias de todo el día o para las franjas horarias especificadas.

6. Implementación de un sistema de seguimiento de estados de las guardias:
    - El sistema permitiría que las guardias pasen por tres estados identificados con colores de semáforo: pendiente, aceptada, confirmada y finalizada. Estos estados reflejarían el progreso y la situación actual de cada guardia asignada.
    - Esta implementación permitiría un seguimiento más detallado y transparente del proceso de asignación y realización de las guardias. Los administradores podrían visualizar fácilmente el estado actual de cada guardia, lo que facilitaría la planificación y la comunicación dentro del sistema de gestión de guardias. Además, esta funcionalidad podría proporcionar una base sólida para futuras mejoras, como la generación automática de informes de guardias realizadas o la asignación inteligente de guardias basada en disponibilidad y preferencias de los profesores.

    Los estados estados funcionarían de la siguiente manera:

    1. Pendiente: Cuando una guardia se crea por defecto su estado se indica como "pendiente". Una vez entre en la franja horaria en la que se debe realizar los profesores de guardia visualizarán esta guardia como pendiente identificada con un color azul claro. En este estado, la guardia aún no ha sido aceptada y no está asignada a ningún profesor.

    2. Aceptada: Una vez que el profesor selecciona una guardia de las que se encuentren en estado pendiente la guardia será asignada a ese profesor, y esta pasa al estado de "aceptada". Aquí, se indica que el profesor ha confirmado su disponibilidad y está comprometido a realizar la guardia en el aula correspondiente. Esta información puede ser visible para otros usuarios del sistema ya que se identifica la guardia con el color verde.

    3. Confirmada: Cuando el profesor se encuentra en el aula donde realizará la guardia, este deberá confirmar que está realizando la guardia y esta pasa al estado de "confirmada". Esto significa que el profesor está presente y listo para asumir la responsabilidad de la guardia. Este estado permite una mejor coordinación y evita confusiones o malentendidos sobre la presencia del

profesor en el lugar correcto. La guardia durante su hora de realización se visualizará en color ámbar.

4. Finalizada: Una vez que el tiempo asignado para la guardia ha concluido, el profesor deberá confirmarlo, y esta se marca como "finalizada". Esto indica que el período de guardia ha terminado y se puede considerar que el profesor ha completado con éxito su responsabilidad. Además, el sistema puede registrar la realización de la guardia realizada para fines de seguimiento y generación de informes.

Estas vías futuras ofrecen mejoras significativas en la gestión de las guardias para una experiencia más interactiva y accesible. Al implementar estas funcionalidades, se puede mejorar la eficiencia y la comunicación en el sistema de gestión de guardias.

10. Anexos

10.1. Manual de instalación

10.1.1. Instalación de Angular

Para poder trabajar con Angular el primer requisito es tener instalado Node.js. En caso de no tenerlo instalado se puede descargar desde la página oficial de Node.js (https://nodejs.org/es).

Para trabajar con el proyecto de Angular descargado desde Github (https://github.com/Xeadnor/TeacherCover):

1. Desde la terminal (cmd) navegar hasta la carpeta del proyecto usando el comando '**cd**'.

```
C:\Users> cd TeacherCover
```

2. Ejecutar el comando '**npm install**' para instalar todas las dependencias del proyecto.

```
C:\Users\TeacherCover\teachercover> npm install
```

3. Ejecutar el comando '**npm install -g @angular/cli**' para instalar Angular CLI globalmente en el sistema.

```
C:\Users\TeacherCover\teachercover> npm install -g
@angular/cli
```

4. Si se quiere abrir el proyecto automáticamente en VisualStudio Code se debe ejecutar el comando '**code .**'

```
C:\Users\TeacherCover\teachercover> code .
```

5. Ejecutar el comando '**ng serve**' para compilar y levantar el proyecto de Angular. Este comando compila el proyecto y lo sirve en un servidor local. Se puede acceder al proyecto desde el navegador web en la dirección 'http://localhost:4200'. Si al comando se le añade la opción '**-o**' al final el proyecto se abrirá automáticamente en el navegador configurado por defecto.

```
C:\Users\TeacherCover\teachercover> ng serve -o
```

'ctrl + c' para detener el servidor una vez levantado.

10.1.2. Instalación de firebase

Para iniciar un proyecto con Firebase se deben seguir los siguientes pasos:

1. Crear una cuenta de Firebase, en la página de oficial: https://firebase.google.com/

    - Haz clic en el botón "Empezar" y luego en "Ir a la consola".
    - Inicia sesión con tu cuenta de Google o crea una nueva.
    - Haz clic en "Agregar proyecto" y sigue las instrucciones para completar la creación del proyecto.

2. Configurar el proyecto en Firebase:

    - Una vez creado el proyecto, selecciona "Desarrollar" en el menú de la barra lateral izquierda.
    - Elige el servicio que deseas utilizar, como Firestore, Authentication, Storage, etc., y sigue las instrucciones para configurar cada servicio según tus necesidades. Por ejemplo, si deseas utilizar Firestore, puedes hacer clic en "Firestore Database" y luego en "Crear base de datos".

3. Obtener la configuración de Firebase:

    - En la consola de Firebase, haz clic en el icono de configuración (la rueda dentada) en la parte superior izquierda y selecciona "Configuración del proyecto".
    - En la pestaña "General", desplázate hacia abajo hasta encontrar la sección "Tus aplicaciones".
    - Haz clic en el botón "</>" junto a la plataforma web.
    - Se abrirá una ventana emergente con la configuración de Firebase, que incluye el objeto de configuración con las credenciales y la información del proyecto.

4. Configurar Angular para utilizar Firebase:

    En la carpeta raíz del proyecto Angular,en el archivo `**src/environments/environment.ts**` agrega la configuración de Firebase del punto anterior al objeto environment en el archivo `**environment.ts**`. Puedes asignar la configuración al campo `**firebaseConfig**`. El objeto debe tener la siguiente estructura:

```
export const environment = {
  production: false,
  firebaseConfig: {
    apiKey: 'TU_API_KEY',
    authDomain: 'TU_AUTH_DOMAIN',
```

```
    databaseURL: 'TU_DATABASE_URL',
    projectId: 'TU_PROJECT_ID',
    storageBucket: 'TU_STORAGE_BUCKET',
    messagingSenderId: 'TU_MESSAGING_SENDER_ID',
    appId: 'TU_APP_ID',
  },
};
```

5. Importar los módulos de Firebase en Angular:

   - Abre el archivo `**src/app/app.module.ts**`.
   - Importa los módulos necesarios de Firebase. Por ejemplo, si deseas utilizar Firestore, puedes importar el módulo `**AngularFirestoreModule**` de la siguiente manera:

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AngularFireModule } from '@angular/fire';
import { AngularFireAuthModule } from '@angular/fire/auth';
import { AngularFirestoreModule } from '@angular/fire/firestore';

import { environment } from '../environments/environment';
import { AppComponent } from './app.component';

@NgModule({
  declarations: [AppComponent],
  imports: [
    BrowserModule,
    AngularFireModule.initializeApp(environment.firebaseConfig),
    AngularFireAuthModule,
    AngularFirestoreModule,
  ],
  providers: [],
  bootstrap: [AppComponent],
})
export class AppModule {}
```

Una vez configurado el proyecto:

1. Ejecutar el comando '**npm install -g firebase-tools**' para instalar las herramientas de Firebase globalmente.

```
C:\Users\TeacherCover\teachercover> ng serve -o
```

2. Ejecutar el comando '**firebase login**' e ingresa con las credenciales de Firebase para iniciar sesión en Firebase desde la línea de comandos. La cuenta debe ser la propia del proyecto para poder acceder.

```
C:\Users\TeacherCover\teachercover> firebase login
```

3. Una vez logueado con firebase ejecutar el comando '**`ng add @angular/fire`**' para agregar AngularFire al proyecto de Angular. Este comando instala las dependencias de AngularFire y agrega la configuración necesaria en el archivo app.module.ts.

```
C:\Users\TeacherCover\teachercover> ng add @angular/fire
```

4. Ejecutar el comando '**`firebase init`**' para inicializar el proyecto en Firebase.

```
C:\Users\TeacherCover\teachercover> firebase init
```

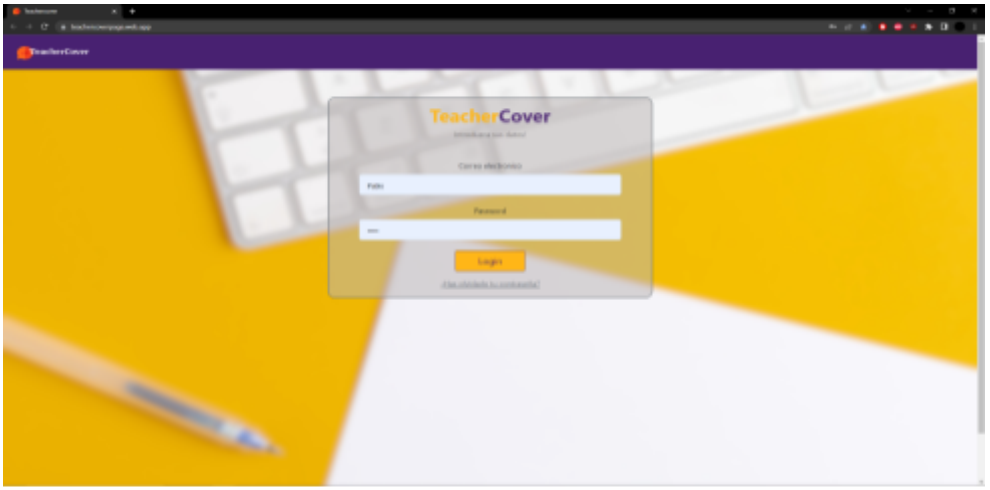
5. Una vez inicializado el proyecto de Firebase, para compilar y desplegar la aplicación Angular en Firebase se debe ejecutar el comando '**`ng build --prod && firebase deploy`**'. Este comando compila tu proyecto para producción y lo despliega en Firebase Hosting.

```
C:\Users\TeacherCover\teachercover> ng build --prod &&
firebase deploy
```

10.2. Manual de Usuario

10.2.1. Entrar en la aplicación

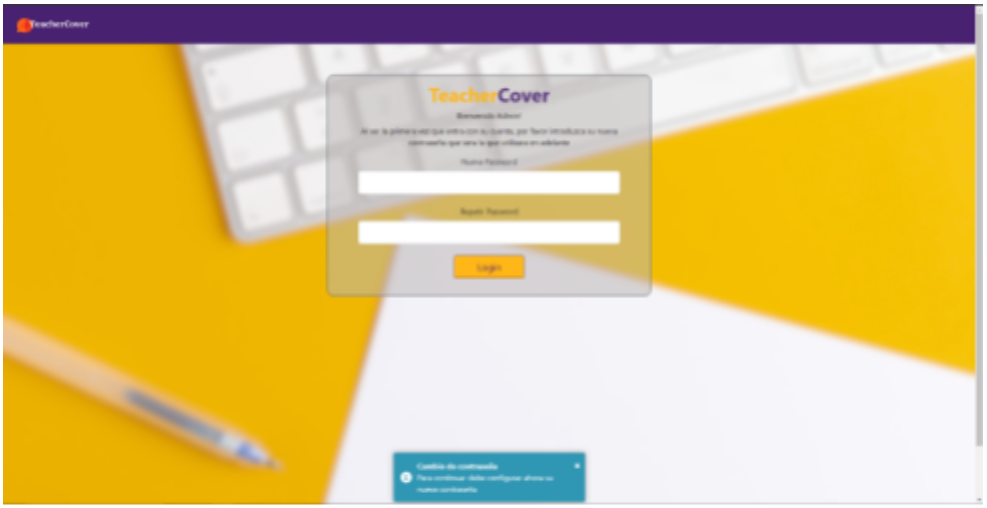
Para entrar en la aplicación se accede mediante la url teachercoverpage.web.app, la cual accede siempre al abrirla al componente Login.



Login inicio TeacherCover
Aplicación real

10.2.2. Primer login

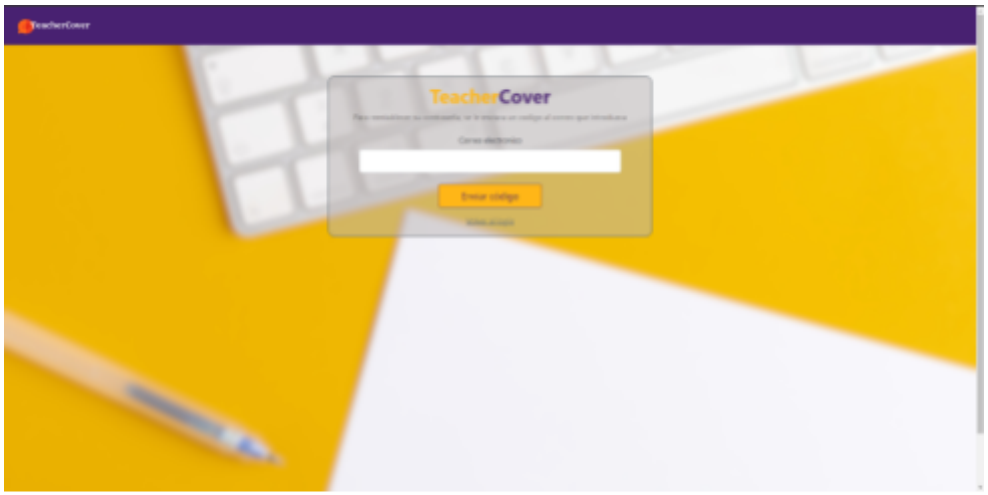
Si es la primera vez que un usuario entra en la aplicación con los datos que le proporciona un administrador, este será redirigido a la siguiente interfaz donde podrá introducir su nueva contraseña



Primera entrada a la aplicación
Aplicación real

10.2.3. ¿Has olvidado tu contraseña?

Si un usuario no se acordara de su contraseña tendría la opción de pulsar en el enlace “¿Has olvidado tu contraseña?”. Este le redirigirá a la siguiente pantalla donde debería escribir su correo electrónico. Al realizarlo se le enviará un email con información para proceder con el cambio de contraseña.

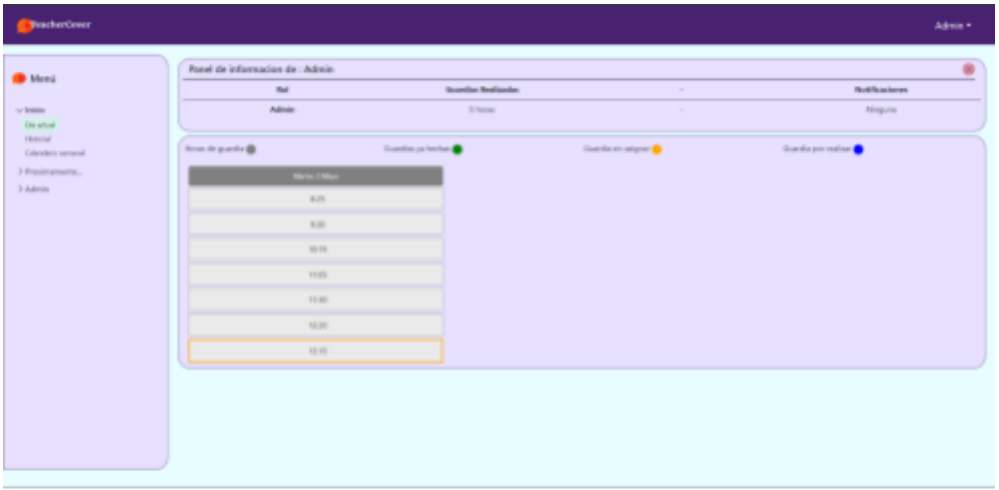


Recuperar contraseña
Aplicación real

10.2.4. Inicio aplicación

Cuando se inicia sesión, se inicia la página con el recurso “Día actual” cargado automáticamente, el cual funciona como vista inicial o home.

Este recurso muestra el horario del día actual con una leyenda para informar al usuario y enseñarle a usar la aplicación. Desde este componente el profesor será capaz de ver la hora de guardia que tiene ese día, y al llegar esa hora se le permitirá ver las guardias disponibles y elegir una de ellas.



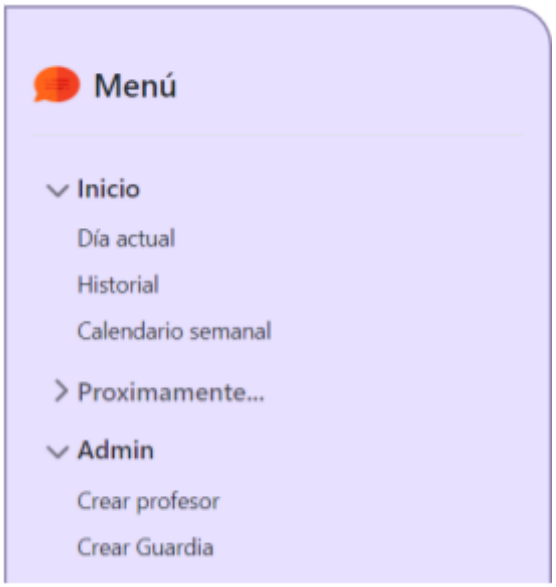
Página principal de la aplicación
Aplicación real

10.2.5. Menú

En el sidebar tenemos un apartado “Menú” compuesto por varios botones desplegables los cuales permiten acceder a las diversas opciones que tendrá un usuario. Por defecto el usuario se encuentra al entrar en la página “Día actual”.

El primer bloque desplegable se muestra a todos los tipos de usuarios. Este contiene gran parte de la aplicación y sus funcionalidades como consultar el historial de guardias del profesor, ver su calendario semanal de guardias, elegir una guardia y ver las disponibles.

El perfil de Administrador cuenta con un segundo bloque llamado “Admin”. En este apartado se podrá acceder a las opciones propias de administrador. Permite gestionar a los profesores y a las guardias, tanto crear, eliminar y modificar.



Menú de la aplicación
Aplicación real

10.2.6. Historial guardias

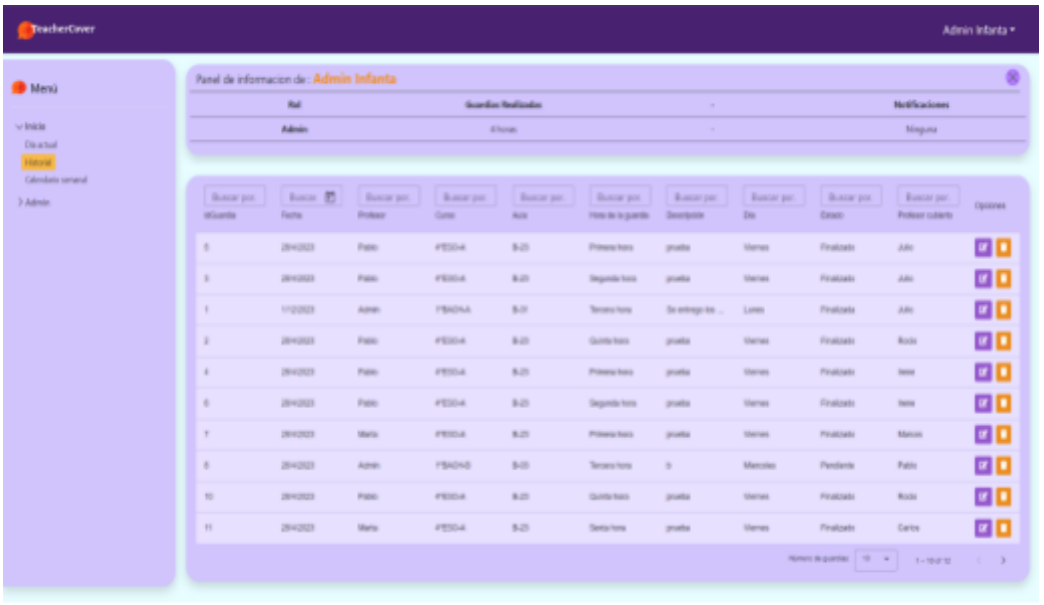
El historial es una de las características más complejas y útiles para los usuarios y administradores. Desde esta interfaz se podrán visualizar las guardias y todos los datos adjuntos a ellas, desde aquí también será posible eliminarlas o modificarlas.

Un perfil usuario profesor solo tendrá acceso a ver sus propias guardias realizadas. En cambio, un administrador, tendrá acceso a todas las guardias realizadas por todos los usuarios, y podrá borrarlas o modificarlas.

Esta interfaz cuenta con varios filtrados, el primero de ellos es de filtrado por texto, todas las columnas tienen un cuadro de texto donde podrá el usuario escribir su búsqueda deseada. Este filtrado es dinámico y admite varios a la vez, se puede

realizar una búsqueda escribiendo en 3 campos distintos, por el profesor “Carlos”, un “jueves” en el aula “B-23”.

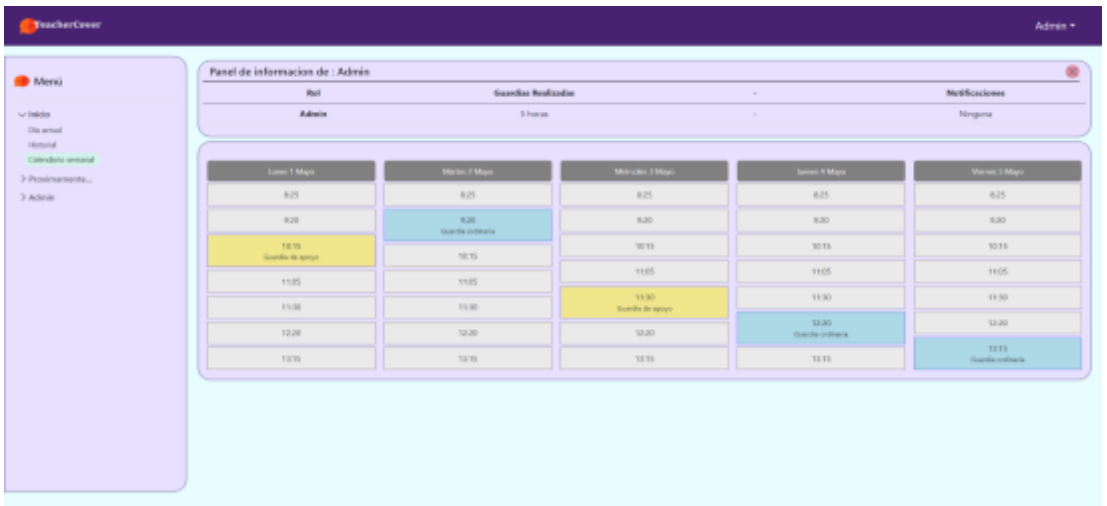
El segundo filtrado se realiza al hacer click en el nombre de una columna, este filtra de manera ascendente o descendente.



Historial de guardias
Aplicación real

10.2.7. Calendario semanal

La última opción en el desplegable es la vista del calendario semanal, esta es una interfaz meramente informativa. Se le marcará al usuario que días y en qué horas tiene horas de guardia o guardias de apoyo, ambas representadas con distintos colores en la tabla para que resulte más sencillo identificarlas.



Calendario semanal
Aplicación real

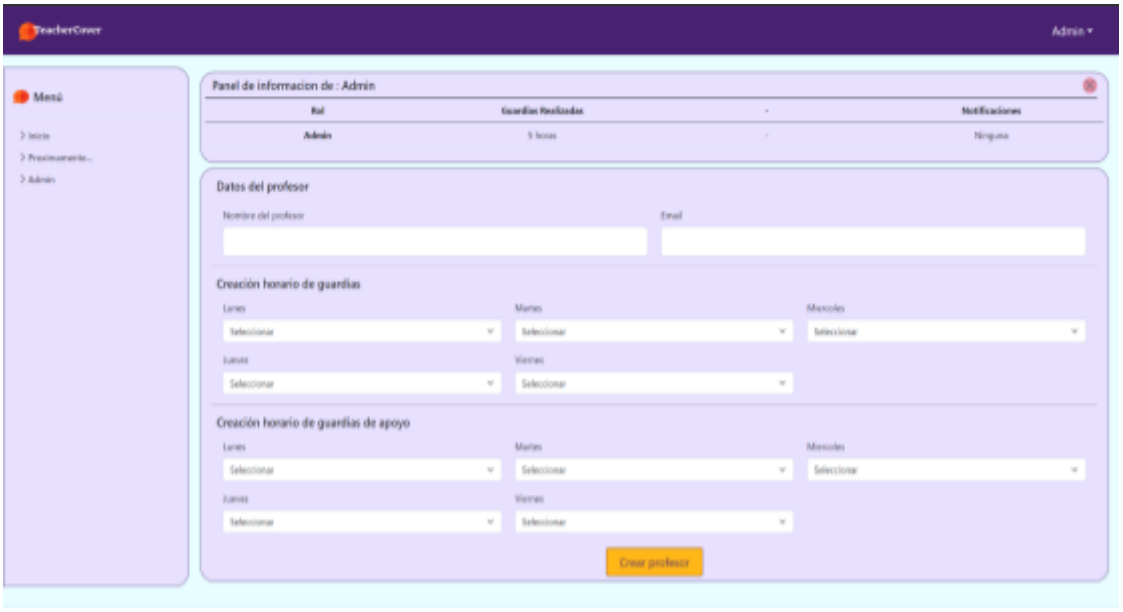
10.2.8. Historial profesores

Esta pantalla sigue la misma estética y funcionalidad que el historial de guardias. Esta es una de las opciones a las cuales los perfiles de administrador tienen acceso, desde aquí los administradores podrán visualizar a todos los profesores, usar los filtros ya mencionados anteriormente, y la opción de borrarlos o editarlos.

Listado de profesores
Aplicación real

10.2.9. Crear profesor

La primera de las opciones de un administrador es la de crear un usuario (profesor), este introducirá el nombre, email, horario de guardias y horario de guardias de apoyo. El resto de campos se calculan y se introducen solos, como la contraseña, que por defecto inicialmente es “IESinfanta23”

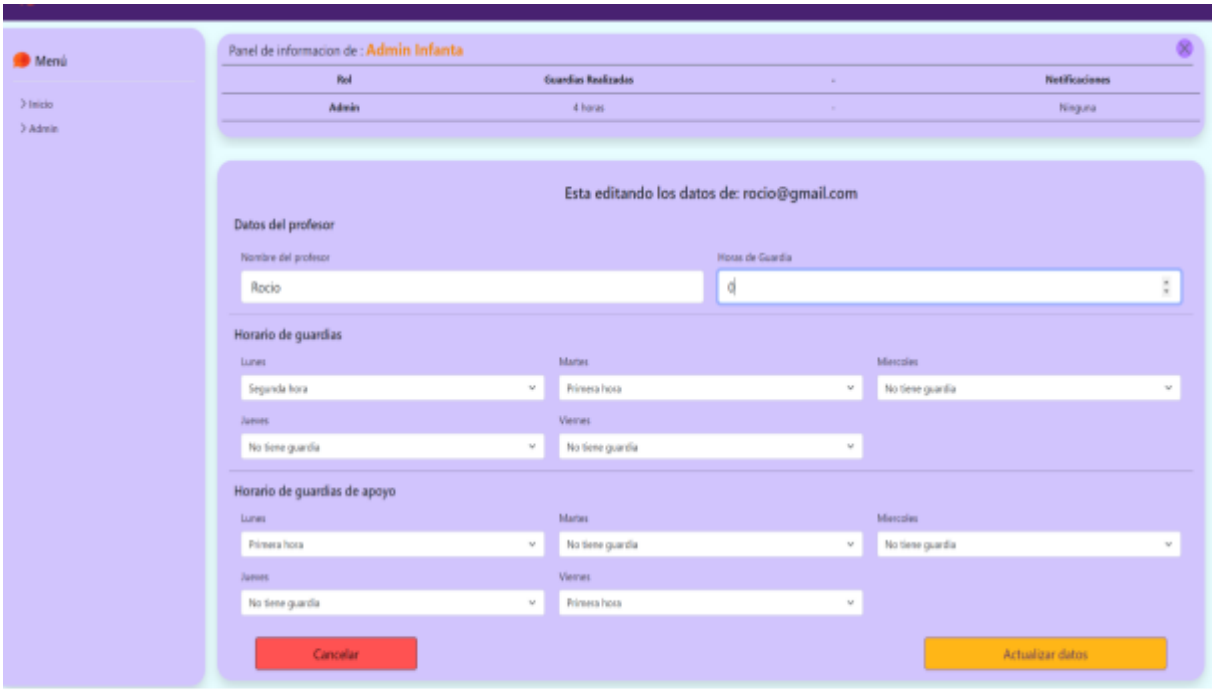


Crear nuevo profesor
Aplicación real

10.2.10. Editar profesor

Editar profesor es una de las opciones junto a borrar de historial profesores, la estética y lógica de esta sigue la misma que la de crear profesor, todos los campos son los mismos menos el email, que se reemplaza por horas de guardia, la principal diferencia es que todos estos campos están ya rellenos con los datos del profesor al que se haya querido modificar o visualizar.

Una vez el profesor tenga los nuevos datos deseados se pulsa el botón de actualizar y estos se cambiarán.

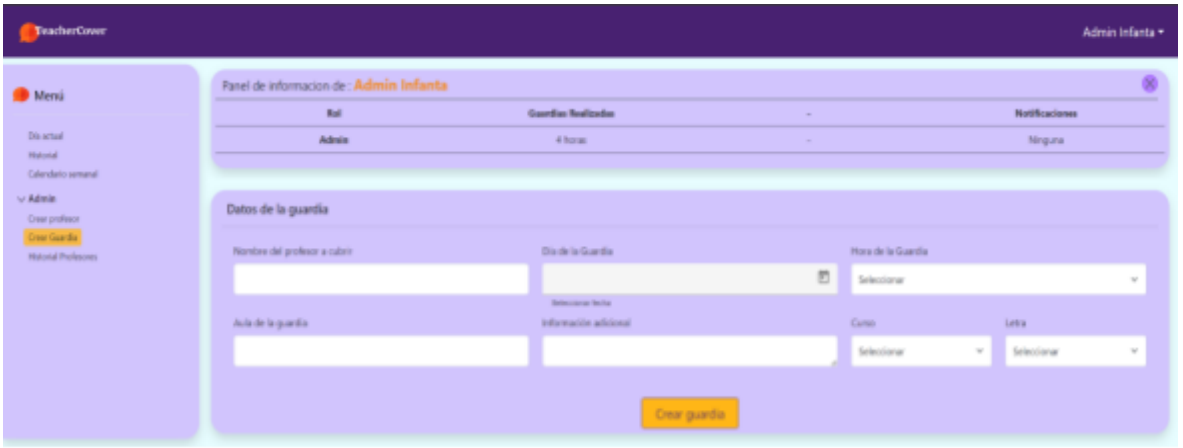


Editar un profesor
Aplicación real

10.2.11. Crear guardia

Otra de las opciones de un administrador es la de crear una guardia, este introducirá el nombre del profesor a cubrir, el cual tiene un propio campo para buscar a este profesor entre todos los disponibles en la base de datos, el día de la guardia, la hora, el aula, si el profesor ha dejado algún mensaje o información necesaria, curso y letra.

Esta se introduce a la base de datos completando automáticamente todos los demás campos hasta que un profesor la seleccione y la realice.

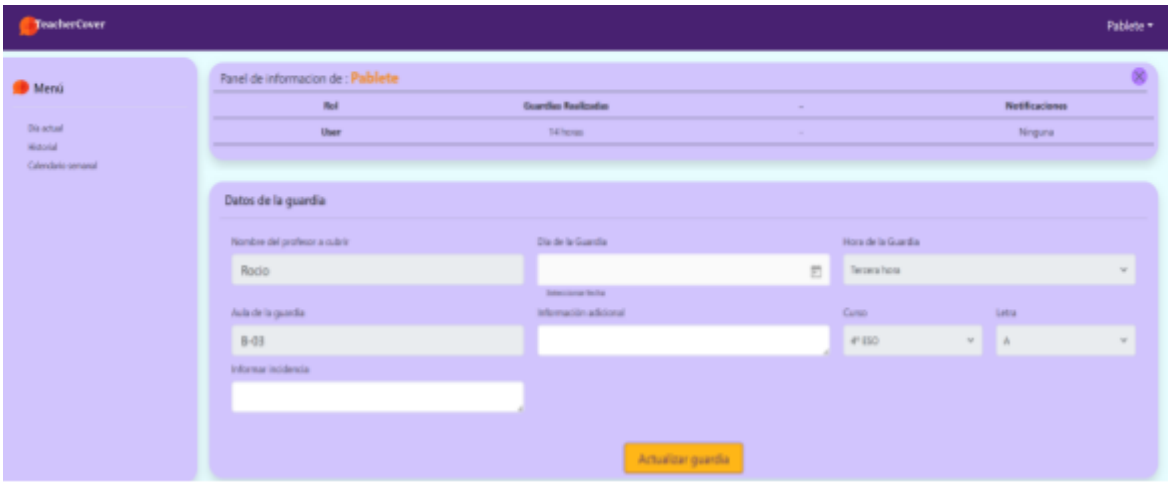


Crear nueva guardia
Aplicación real

10.2.12. Editar guardia

Otra de las opciones que puede encontrar un usuario administrador es la de editar una guardia. La guardia seleccionada desde el listado de guardias abrirá una vista de formulario con los campos que aparecen en la vista de crear guardia rellenos con los datos actuales de dicha guardia. El administrador podrá editar todos los campos de la guardia salvo el campo de incidencia que aparecerá como deshabilitado.

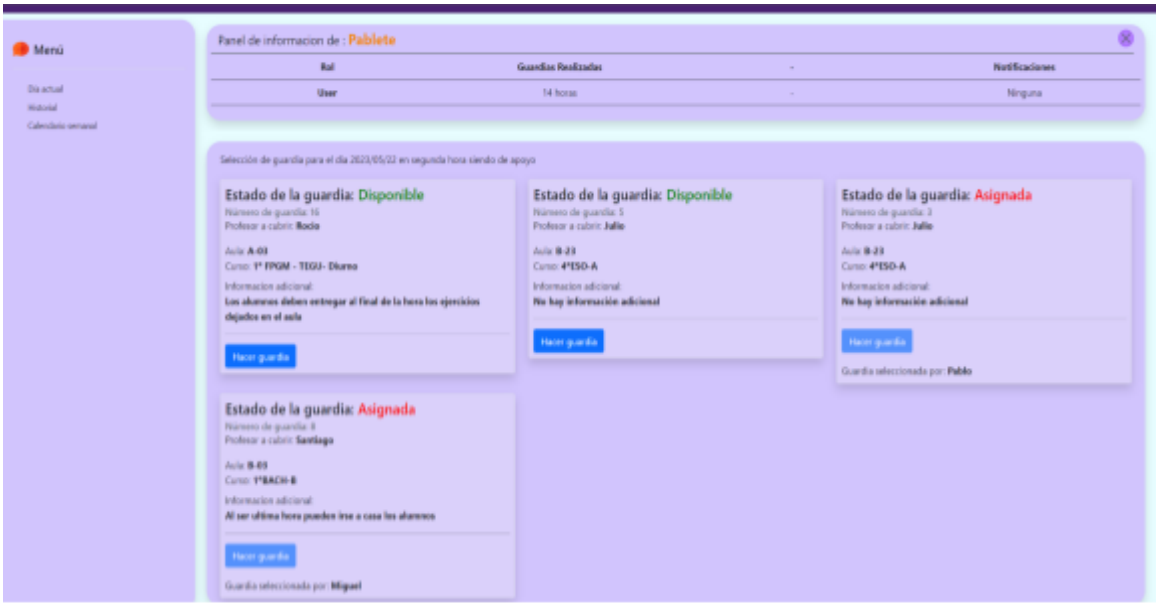
En caso de un usuario profesor podrá realizar los mismos pasos que usuario administrador seleccionando desde la lista de guardias una guardia. Los campos al igual que con un usuario administrador vendrán rellenos en el formulario pero en este caso todos aparecerán deshabilitados para su modificación salvo el campo de incidencia.



Editar una guardia
Aplicación real

10.2.13. Seleccionar guardia

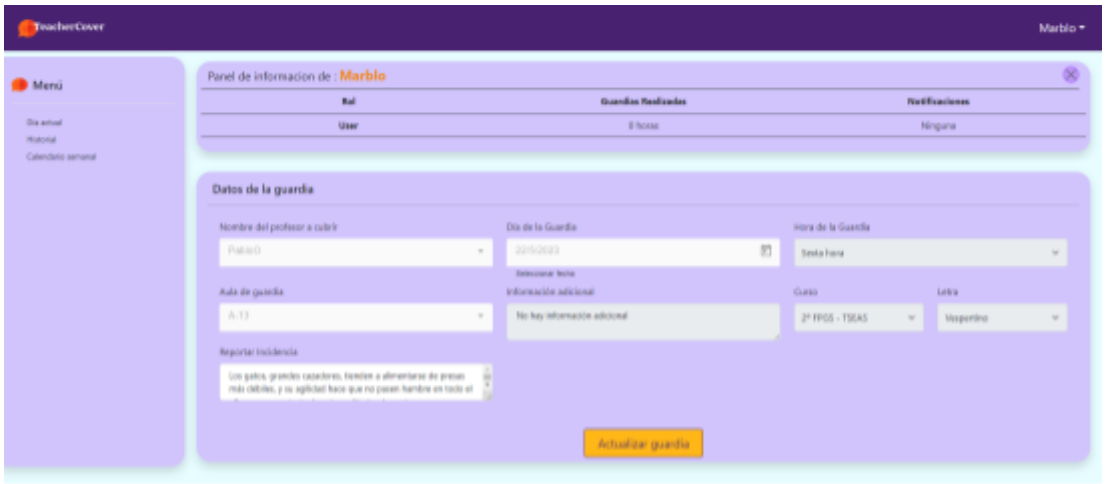
Cuando un profesor entre a una hora de guardia normal o de apoyo este entra la opción de ver las guardias que hay que cubrir esa hora, estas se irán actualizando en tiempo real a medida que los demás profesores las vayan eligiendo, para seleccionar una habrá un botón el cual con pulsarlo ya se le asigna al profesor.



Seleccionar guardia
Aplicación real

10.2.14. Crear incidencia

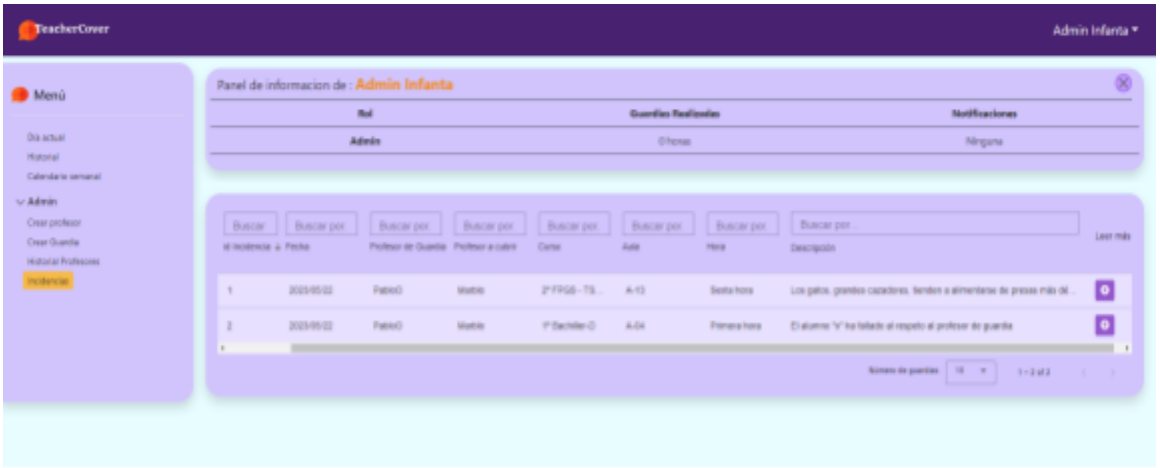
Para que un usuario profesor pueda crear una incidencia deberá realizar los pasos que se muestran en el apartado Editar Guardia (10.2.12) de este anexo. Al rellenar el campo de incidencia se creará una nueva incidencia que podrán visualizar los administradores.
Al clicar en la opción de editar guardia se muestra un formulario que permite editar o añadir texto en el campo de incidencia.



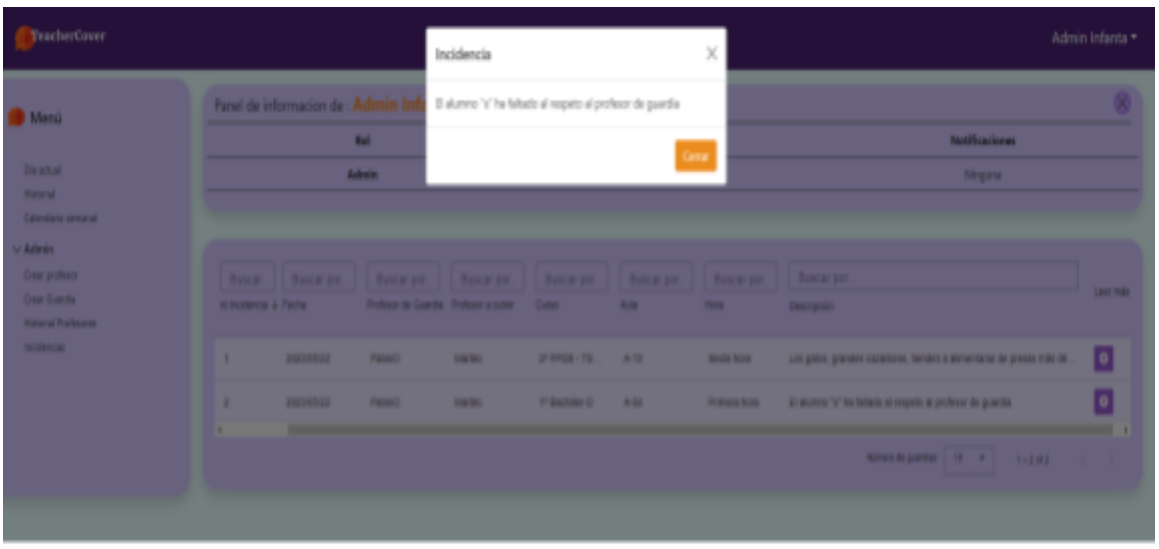
Editar guardia/ añadir incidencia
Aplicación real

10.2.15. Listado de incidencias

Un acuario administrador en las opciones de administrador tiene un apartado de incidencias en el que se muestra un listado con todas las incidencias existentes.
El formato en el que se muestran las incidencias es una tabla como la de historial de guardias o profesores, con campos que permiten filtrar por los distintos contenidos.
En la última columna de la tabla “Leer más”, junto a la descripción de la incidencia aparece un botón “+” que al pinchar sobre él se muestra una pantalla emergente con todo el texto de la descripción de la incidencia, que permite hacer scroll, por si este fuese demasiado largo para el campo de la tabla.



Listado de incidencias
Aplicación real



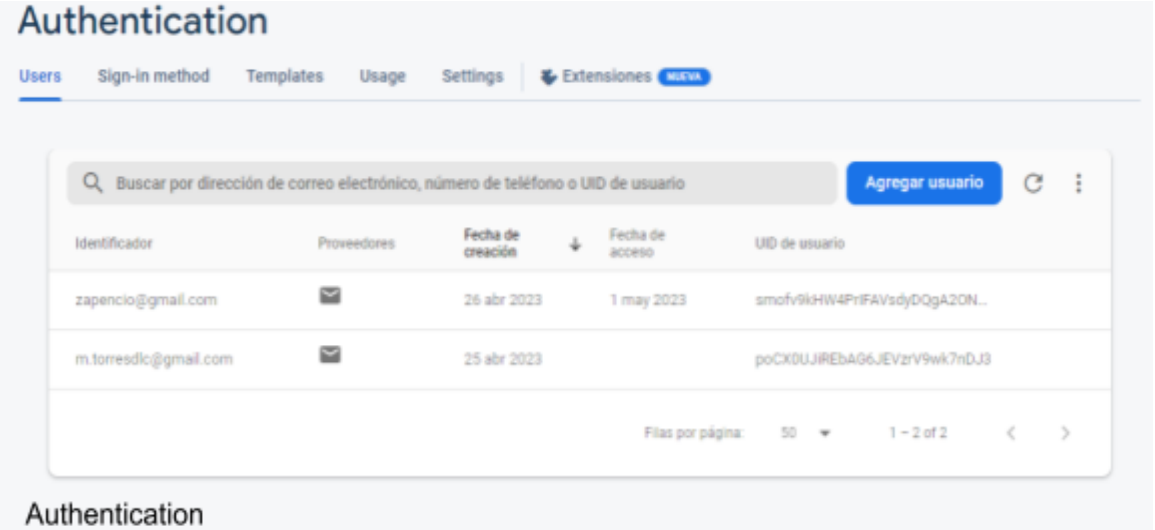
Leer más descripción de la incidencia
Aplicación real

10.3. Módulos de Firebase

TeacherCover tiene configurado toda la parte de servidor y backend en Firebase, lo que supone que todos los métodos y el trato de datos relacionados con la persistencia de estos mismos recae sobre la base de datos de Firebase, en este caso Firestore cloud, la cual tiene una serie de método propios para lograr un fácil acceso a ellos.

10.3.1. Login

Para el login a la aplicación de TeacherCover se hace uso del módulo Authentication de Firebase. Como se puede ver en la imagen, mantiene un registro de los usuarios que en este caso el administrador crea, proporcionando un correo y una contraseña, esta última queda encriptada y no es visible para mantener la seguridad.



Aplicación Firebase

Una vez se tiene almacenado los usuarios y trás importar firebase/auth, firebase ofrece un largo repertorio de métodos. Específicamente para loguearse dispone del auth.login(email,password) el cual se encarga de comprobar si el usuario existe y los datos son correctos.

```
try {
  await this.auth.login(emailLogin, password);
  this.router.navigate(['/pagina/calendario']);
  this.toastr.success(
    "Bienvenido " + newProf.getName(),
    "Inicio de sesión correcto",
    {
```

```
      timeOut: 3000,
      closeButton: true,
      positionClass: "toast-top-right",
    }
  );
} catch (error: any) {
  this.toastr.error(
    "Los datos introducidos no coinciden con ninguna cuenta",
    "Datos incorrectos",
    {
      timeOut: 3000,
      closeButton: true,
      positionClass: "toast-bottom-center",
    }
  );
}
// En el auth.service.ts
login(user: string, pass: string) {
  return this.auth.signInWithEmailAndPassword(user, pass);
}
```

10.3.2. Resetear contraseña

Firebase también facilita la funcionalidad de recuperar contraseña.

Dispone del método sendResetPassword(email) el cual recibe por parámetro el email del usuario que ha olvidado su contraseña. Este método se encarga de enviarle un correo al usuario el cual contiene un link que redirecciona a una página para poder cambiar la contraseña.

```
this.auth.sendResetPassword(email);

// En el auth.service.ts
sendResetPassword(email: string) {
  this.auth.sendPasswordResetEmail(email).then(
    () => {
      // éxito
    },
    (error) => {
      console.log("error");
    }
  );
}
```

Glosario de términos

- **Accesibilidad**: Característica de la interfaz que garantiza que todas las personas, independientemente de sus capacidades, puedan utilizar la aplicación sin dificultades.
- **Aplicación web**: Software que se ejecuta en un navegador web y que está diseñado para realizar una tarea específica.
- **Back-end**: Parte de una aplicación o sitio web que se ejecuta en el servidor y que procesa las solicitudes del usuario, realiza la lógica de negocio y accede a la base de datos para recuperar y almacenar información. Incluye la programación y la infraestructura que se utiliza para gestionar el servidor y la base de datos.
- **Bot**: Programa informático diseñado para realizar tareas automatizadas o simular la interacción humana. Los bots pueden variar en su complejidad y funcionalidad, desde bots simples que realizan tareas repetitivas hasta bots más sofisticados que utilizan inteligencia artificial y aprendizaje automático para interactuar y responder de manera más natural.
- **Clase abstracta**: Clase que no puede ser instanciada por sí misma sino que actúa como un modelo para otras clases. Funciona como plantilla para definir métodos y propiedades que deben de ser implementados por las clases que la heredan.
- **Control de versiones**: herramienta que permite el seguimiento y control de cambios en el código de un proyecto.
- **Dropdown**: Componente de la interfaz que permite al usuario desplegar opciones adicionales al hacer clic en un botón o icono.
- **Enlace de datos bidireccional**: Característica de Angular que permite la sincronización de datos entre el modelo de la aplicación y la vista.
- **Experiencia de Usuario o UX(User Experience)**: Es la interacción general y la satisfacción que tiene un usuario al utilizar un producto o servicio. Incluye todos los aspectos relacionados con cómo se siente y percibe el usuario durante su interacción, como la facilidad de uso, la eficiencia, la utilidad y la satisfacción general.
- **Expresión Regular o RegEx**: Es una secuencia de caracteres que define un patrón de búsqueda. Se utiliza principalmente en programación y procesamiento de texto para buscar, validar y manipular cadenas de caracteres según ciertos criterios.
- **Framework**: Estructura de software que proporciona herramientas y funciones para facilitar el desarrollo de aplicaciones.
- **Front-end**: Parte de una aplicación o sitio web visible y con la que interactúan directamente los usuarios. Incluye el diseño visual, la interfaz de usuario y la funcionalidad que permite la interacción con el usuario.
- **Header**: Sección superior de la interfaz que suele contener el logo y otros elementos importantes como el menú, botón de login o buscador.
- **IDE**: Entorno de desarrollo integrado, un software que proporciona un conjunto completo de herramientas para desarrollar, depurar y mantener aplicaciones.
- **Interfaz**: Conjunto de elementos visuales y funcionales que permiten al usuario interactuar con la aplicación.
- **Interfaz de Usuario o UI (User Interface)**:Es el medio a través del cual un usuario interactúa con un sistema o aplicación. Incluye todos los elementos visuales, como

botones, formularios, menús y elementos de navegación, así como las acciones y comportamientos asociados a ellos. El objetivo principal de la interfaz de usuario es proporcionar una forma intuitiva y eficiente para que los usuarios interactúen con el sistema.

- **Inyección de dependencias**: Técnica de programación en la que se pasan las dependencias de un objeto como parámetros en lugar de crearlas dentro del objeto.
- **Login**: página o sección de la aplicación donde el usuario ingresa sus credenciales para acceder a su perfil de la aplicación.
- **Metodología ágil**: Es un enfoque de gestión de proyectos de software que se centra en la entrega rápida, iterativa e incremental de un producto de software de alta calidad que cumpla con las necesidades y expectativas del cliente o usuario final.
- **Navbar**: Elemento de la interfaz de usuario que se utiliza comúnmente en diseño web. Se encuentra en la parte superior de la página y suele contener enlaces a otras páginas, botones de acceso rápido o un menú desplegable.
- **Responsividad**: Característica de la interfaz que permite adaptarse a diferentes tamaños de pantalla y dispositivos.
- **Sidebar**: Término utilizado en diseño de interfaces de usuario refiriéndose a una barra vertical u horizontal que contiene enlaces o accesos directos a diferentes secciones de un sitio web o aplicación.
- **Single page**: Diseño de la página principal donde toda la información se encuentra en una sola página. La información se va mostrando sobre la misma página sin modificar la URL según la opción seleccionada para mostrar.
- **Toast**: En el contexto de las aplicaciones móviles y web es un pequeño mensaje emergente o notificación que se muestra de forma temporal en la pantalla para informar al usuario sobre un evento o estado específico. Por lo general, aparece en la parte superior o inferior de la pantalla y desaparece automáticamente después de unos segundos.

Bibliografía y Webgrafía

- Angular. (2022, 28 de febrero). Setting up the local environment and workspace. Recuperado el 6 de abril de 2023, de https://angular.io/
- Bootstrap v5.0. (s.f.). Getting started - Introduction. Recuperado de https://getbootstrap.com/docs/5.0/getting-started/introduction/
- Figma. (s.f.). Recuperado de https://www.figma.com/
- Firebase. (2023, 6 de abril). Recuperado de https://firebase.google.com/?hl=es-419
- FontAwesome. (s.f.). Docs. Recuperado de https://fontawesome.com/
- NPM. (2023, 3 de mayo). ngx-toastr. Recuperado el 11 de mayo de 2023, de https://www.npmjs.com/package/ngx-toastr
- Sebastian. (2018, 17 de febrero). Firebase Cloud Storage With Angular. Medium. Recuperado de https://medium.com/codingthesmartway-com-blog/firebase-cloud-storage-with-angular-394566fd529
- StackBlitz. (s.f.). StackBlitz Angular. Recuperado de https://stackblitz.com/
- Stack Overflow. (s.f.). Recuperado de https://stackoverflow.com/
- TypeScript. (s.f.). TypeScript-Classes. Recuperado de https://www.tutorialsteacher.com/typescript/typescript-class
- W3Schools. (s.f.). TypeScript Tutorial. Recuperado de https://www.w3schools.com/typescript/index.php