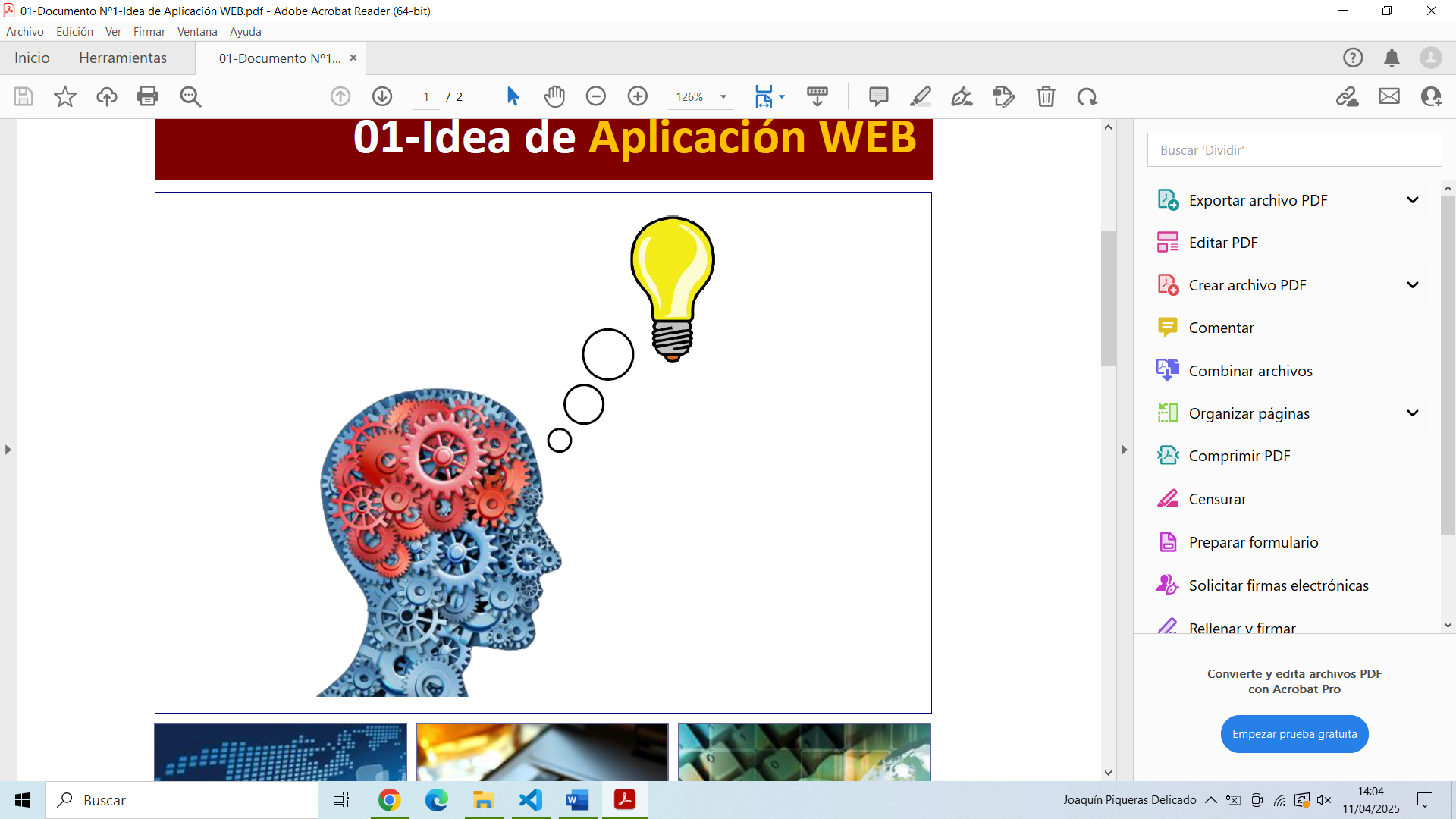


Diseño DE la APLICACIÓN WEB

Proyecto de Desarrollo de Aplicaciones Web



JOAQUÍN PIQUERAS DELICADO



# **01. Diseño de la Aplicación WEB**

**Objetivos de la aplicación:**

Como ya se comentó en la idea del proyecto, el objetivo principal de mi aplicación será la gestión de diferentes centros educativos, sin embargo, esta primera versión de mi aplicación web va a centrase en la gestión de guarderías, las cuales no requieren incorporar gestión de horarios ni de asignaturas.

Para poder gestionar una guardería los objetivos que se deben alcanzar con mi aplicación serán:

* **Inicio de sesión:** Al iniciar la aplicación, debemos permitir al usuario el ingreso a muestra página de inicio siempre y cuando introduzca sus credenciales correctamente.
* **Gestión de usuarios:** La aplicación debe permitir la creación, edición, eliminación y gestión de usuarios de diferentes roles: Administradores, Padres, Alumnos y Profesores.
* **Gestión de grupos:** También se debe permitir la creación, edición y eliminación de grupos (clases) para organizar a alumnos y profesores.
* **Vinculación de Usuarios:** Con la aplicación se podrá vincular los alumnos y profesores a diferentes grupos. Para el caso del rol Alumnos, se permitirá también la vinculación de usuarios con rol de Padres.
* **Mensajes:** Implementación de un sistema de mensajería interna con un sistema de notificación visual para mensajes no leídos.
* **Noticias e Informes:** Creación de noticias por parte de profesores y administradores, y generación de informes (seguimiento académico) de alumnos por parte de los profesores.
* **Faltas de Asistencia:** Los profesores deben poder registrar faltas de asistencia de los alumnos que tengan vinculados.
* **Perfil de Usuario:** Los usuarios deben poder actualizar sus perfiles, incluyendo la foto y la contraseña.

**Interfaz de Usuario:**

La interfaz de usuario será intuitiva y fácil de navegar, con un diseño amigable, responsive para todo tipo de dispositivo y con colores suaves. Para poder acceder a esta interfaz, previamente será necesario iniciar sesión en la página de login, con el correo y contraseña del usuario.

Una vez que el usuario haya iniciado sesión, se le redirigirá a la página o vista de Home, en la cual se mostrará un carrusel con todas las noticias que hayan sido registradas por los profesores o administradores, también aparecerá un header con información del usuario y el botón de cerrar sesión, así como un menú lateral desplegable para poder seleccionar la página a la que queremos navegar.

Las páginas a las que se podrá navegar una vez accedido a la aplicación, dependerán del rol de usuario, pero a continuación muestro un breve resumen de cada una de ellas:

* **Gestión de Usuarios:** Vista en la que aparecerán todos los usuarios registrados, con formulario para crear, editar, eliminar y vincular usuarios. Esta vista solo será accesible para administradores.
* **Gestión de Grupos:** Vista similar a la anterior en la que aparecerán todos los grupos registrados, con posibilidad de crear, editar y borrar grupos de alumnos y profesores.
* **Mensajes:** Un sistema de bandeja de entrada donde los usuarios puedan ver los mensajes recibidos y enviar nuevos.
* **Noticias:** Para el caso de alumnos y padres, esta vista te redirigirá al Home. En caso de acceder como profesor o administrador, accederemos a una vista con la lista de todas las noticias registradas, en la que podremos crear, editar o eliminar noticias.
* **Informes:** En caso de acceder como alumno, se tendrá acceso a la lista de todos los informes generados para este alumno. En caso de acceder como padre, se accederá a todos los informes generados para el alumno asociado. Por último, en caso de acceder como profesor o administrador, se mostrarán todos los informes creados por este usuario y se podrán crear nuevos, o editar y borrar antiguos, asignando cada informe a uno de los alumnos que tenga vinculado el usuario.
* **Faltas:** Vista similar a la de informes, pero mostrando las faltas de asistencia registradas en caso de acceder como alumno o padre. En caso de acceder como profesor o administrador, se podrán registrar, editar o borrar las faltas de los alumnos vinculados.

# **02. Estudio Funcional**

En este apartado, voy a detallar cada una de las funcionalidades que tendrá mi aplicación y como acceder a ellas.

**Funcionalidades:**

* **Inicio de sesión:** La primera vista que a la que va a acceder el usuario al entrar en mi aplicación web, será la del login. En esta vista, el usuario podrá acceder a la página inicial y a cada una de las siguientes funcionalidades siempre que introduzca su correo y contraseña de forma correcta. El usuario tendrá la opción de visualizar la contraseña introducida pulsando el botón que aparecerá en el margen derecho del input de la contraseña. Con esta vista, consigo un inicio de sesión seguro, generando un token temporal en la base de datos que se almacenará en la memoria del navegador. Este token, será enviado en todas las peticiones a nuestro servidor para así obtener datos solamente cuando el usuario esté verificado.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Ilustración 1.-Vista del login

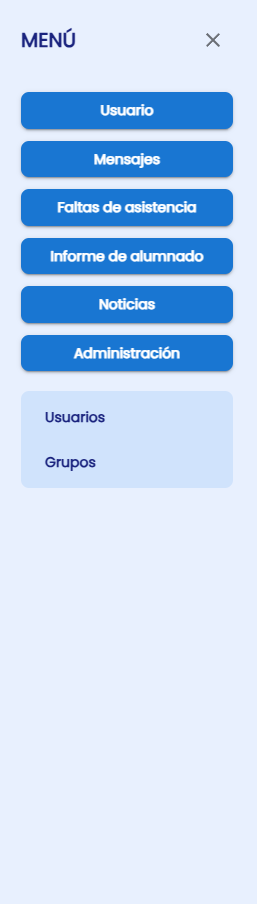
* **Cierre de sesión:** En el margen derecho del encabezado de mi aplicación, nos encontraremos con el icono de cerrar sesión, el cual al pulsarlo cerrará la sesión iniciada por cualquier usuario registrado.

Ilustración 2.-Icono Logout en rojo



* **Gestión de usuarios:** Mi aplicación va a permitir la gestión de los usuarios, es decir, la creación, edición y eliminación de usuarios en mi base de datos. Para esto, desde la aplicación, será necesario acceder desde el panel lateral de navegación a la página de Administración de usuarios. Una vez aquí, podremos visualizar un listado de los datos de todos los usuarios registrados en la base de datos, incluyendo el nombre, apellidos, email y grupo o padres vinculados en caso de tenerlos. Además, desde esta página, el usuario será capaz de añadir nuevos usuarios haciendo clic en el botón de Añadir usuario. Con esta acción, aparecerá una ventana emergente con un formulario en el cual se introducirán los datos del usuario, siendo obligatorios el nombre, apellidos y rol. También será posible editar un usuario de nuestra lista de usuarios, para ello tendremos que localizar el usuario en cuestión y en la columna de Acciones pulsar el botón de Editar. Con esto nos aparecerá un nuevo formulario con los datos de este usuario, pudiendo editar cualquiera de ellos. En esta columna también tendremos el botón de borrar, con el cual se nos avisará en una nueva ventana emergente si realmente deseamos eliminar el usuario seleccionado y en caso de aceptar será eliminado. Por último, podremos visualizar otro botón, denominado Vincular, con el cual podremos asociar a este usuario un grupo en caso de ser alumno o profesor y un padre en caso de ser alumno.

Ilustración 3.-Menú de navegación lateral



* Interfaz de usuario gráfica, Aplicación

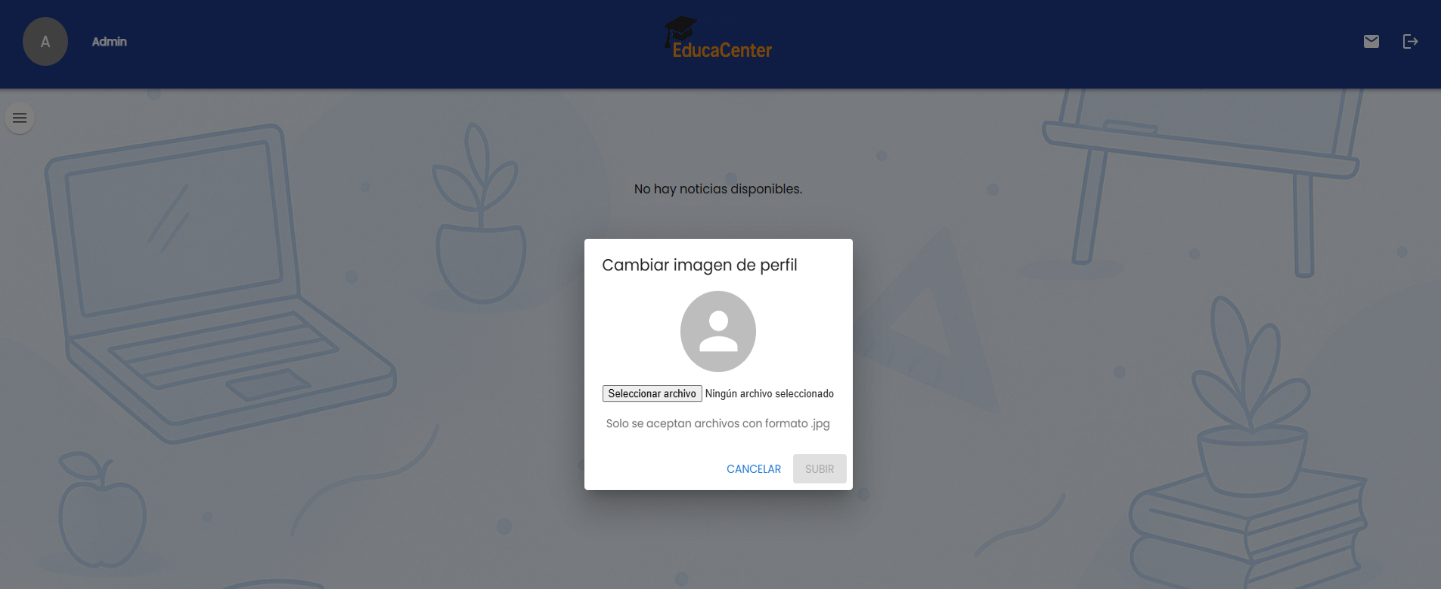
  El contenido generado por IA puede ser incorrecto.**Gestión de grupos:** Desde el panel de navegación, podremos acceder a la opción de Administración de grupos, para poder añadir, editar o borrar un grupo. Al igual que ocurría con la vista de Administración de usuarios, vamos a poder visualizar todos los nombres de los grupos registrados en la base de datos, y en la columna de Acciones tendremos los botones de editar o eliminar cada uno de ellos. Estos botones abrirán una nueva ventana emergente desde la cual podremos confirmar la edición o eliminación del grupo en cuestión. También tendremos arriba a la derecha de nuestra lista un botón para Añadir usuarios, con el cual nos aparecerá un nuevo formulario en el que tendremos que indicar el nombre del nuevo grupo.

Ilustración 5.- Vista de Gestión de grupos

Ilustración 4.-Vista de Gestión de Usuarios

* **Actualizar foto de perfil:** Los usuarios podrán agregar o actualizar su foto de perfil desde cualquier página de nuestra aplicación. Para ello, tendremos que dirigirnos al header de nuestra aplicación y pulsar sobre la foto de nuestro perfil. De esta forma se nos abrirá una nueva ventana flotante con la cual podremos subir un archivo en formato jpg (solo voy a permitir este formato ya que es compatible con la mayoría de los navegadores y no es muy pesado). Al subir este archivo, visualizaremos una vista previa de nuestra foto de perfil en la ventana flotante y si esta es de nuestro agrado podremos pulsar en Guardar para actualizarla.

Ilustración .-Vista de cuadro para actualizar foto de perfil



* **Visualización de usuario:** Los usuarios de la aplicación, podrán acceder a la información de su perfil o bien haciendo clic directamente en su nombre de usuario ubicado en el header o bien dirigiéndonos al menú de navegación lateral, seleccionando la primera opción. En esta vista, se podrá visualizar toda la información personal disponible para el usuario y también cambiar su propia contraseña actual.
* **Actualizar Contraseña:** Al igual que ocurre con la foto de perfil, los usuarios también serán capaces de actualizar su contraseña. Para esto, desde cualquier ruta en nuestra aplicación, podremos pulsar sobre el nombre de usuario ubicado en el header de la aplicación para dirigirnos a nuestro perfil personal. Una vez aquí, si pulsamos el botón de cambiar contraseña, se nos abrirá una nueva ventana emergente, en la cual se nos solicitará escribir la contraseña anterior y la nueva contraseña, pudiendo visualizar u ocultar esta información con el icono del ojo en cada uno de los inputs. Si hacemos clic en Guardar y la contraseña antigua es la correcta, se procederá al cambio de esta, en caso contrario se informará con un mensaje de error al usuario en la misma ventana.
* **Interfaz de usuario gráfica

  El contenido generado por IA puede ser incorrecto.Navegar a Home:** Para poder navegar a nuestra página principal o Home, simplemente bastará con hacer clic en el logo de la aplicación EducaCenter, situado en el centro de nuestro header en todas las vistas.

Ilustración .-Vista de cuadro para actualizar contraseña

**Imagen que contiene Logotipo

El contenido generado por IA puede ser incorrecto.**

Ilustración .- Logo EducaCenter

* **Visualización de noticias:** Desde el menú de navegación lateral, podremos acceder a la vista de las noticias. Esta vista será la misma que la que aparece para Home, y en ella podremos visualizar un carrusel de imágenes con el título de la noticia debajo de estas. A ambos lados de la imagen tendremos una flecha, la cual nos permitirá desplazarnos a la siguiente noticia (derecha) o a la anterior (izquierda), siendo la primera notica en aparecer la última que haya sido generada, es decir, la más reciente. Si hacemos clic en seguir leyendo, debajo del título de la noticia, nos aparecerá la información completa de esta.
* **Visualización de informes:** Para poder visualizar los informes generados sobre un alumno, el usuario tendrá que navegar desde el menú lateral a la página de Informes. Una vez aquí, este tendrá acceso a todos los informes generados para el alumno vinculado a este usuario. Al hacer clic en alguno de estos informes (identificados con un número, fecha y apareciendo el más reciente en primera posición), el usuario tendrá acceso a la descripción del informe que haya elaborado el profesor para este alumno.
* **Visualización de faltas de asistencia:** Esta funcionalidad es similar a la visualización de informes. En este caso, el usuario podrá navegar desde el menú lateral a la página de Faltas, en la cual podrá visualizar todas las faltas de asistencia que hayan sido registradas para el alumno vinculado al usuario. Estas faltas aparecerán en forma de lista, y en cada fila podremos visualizar la fecha en la que se produjo la falta de asistencia y si esta estuvo o no justificada.
* **Gestión de noticias:** En caso de que el usuario tenga los permisos adecuados, si este accede a la página de Noticias desde el menú lateral de navegación, tendrá acceso a todas las noticias generadas y podrá modificarlas eliminarlas o crear nuevas. La interfaz para poder gestionar estas noticias será muy similar a la de la gestión de usuarios, en la cual podremos visualizar el título de la noticia y si hacemos clic en el botón de editar, se nos abrirá un formulario en una ventana emergente para poder editar el contenido o título de la noticia. Si pulsamos en borrar nos aparecerá una ventana de confirmación con el nombre del título de la noticia. Además, tendremos otro botón llamado foto, con el cual podremos agregar o editar la foto asociada a cada una de las noticias, las cuales vendrán con una imagen por defecto si no se sube nada. Por último, si hacemos clic en Nueva, podremos generar una nueva noticia con su título y contenido. Cuando la agreguemos, nos aparecerá en la primera fila de nuestra lista, pudiendo entonces añadirle una foto si fuese necesario.
* **Gestión de informes:** La gestión de los informes del alumnado podrá ser realizada solo por aquellos usuarios permitidos. Esta gestión será posible, accediendo a la página informes desde nuestro menú lateral de navegación. Una vez en la vista, el usuario podrá visualizar todos los informes que este usuario haya generado, pudiendo editar o borrar cualquiera de ellos o añadir uno nuevo. La interfaz de usuario para la gestión de estos informes será la misma que para el resto de las gestiones, con el botón de editar o eliminar al lado de cada uno de los informes a gestionar y el botón de añadir para añadir uno nuevo, mediante un formulario en una ventana emergente.
* **Gestión de faltas de asistencia:** Las faltas de asistencia también podrán ser gestionadas por aquellos usuarios que tengan los permisos adecuados. Para poder gestionarlas, de nuevo accederemos a la vista de Faltas, desde el menú lateral de navegación. En esta vista, el usuario podrá visualizar todas las faltas de asistencia que ha registrado para un determinado alumno, visualizando la fecha de la falta de asistencia, el alumno al que ha sido adjudicada y si estaba justificada o no. Si pulsamos en el botón de editar, nos aparecerá un nuevo formulario emergente, donde seleccionar la fecha de la falta, si está o no justificada y el alumno al que va dirigida. Solo aparecerán, en un desplegable, los alumnos que tenga vinculado el usuario que está gestionando la falta. Para crear una nueva falta, aparecerá el mismo formulario, pero vacío, y en caso de querer eliminarla, se mostrará el dialogo de confirmación correspondiente.
* **Mensajería interna:** Para poder acceder a esta funcionalidad, el usuario podrá hacer clic en el icono del mensaje que aparece arriba a la derecha en el header, o bien dirigirse a la vista Mensajes del menú lateral de navegación. Una vez se ha accedido a esta vista, el usuario visualizará una lista con todos los mensajes recibidos, y sus remitentes, apareciendo en primera fila siempre el más reciente. Si se pulsa en alguno de estos mensajes, podremos visualizar el contenido de este. En caso de querer visualizar los mensajes enviados, podremos cambiar a la pestaña llamada Enviados que aparecerá arriba en nuestra vista, obteniendo de nuevo la misma vista que antes, pero para los mensajes que se han enviado y en lugar de aparecer el remitente aparecerá el receptor. Por último, en ambas vistas, nos aparecerá un icono grande abajo a la derecha con el símbolo ‘+’, el cual al pulsarlo nos permitirá crear un nuevo mensaje. Para crear el mensaje nos aparecerá de nuevo una ventana emergente en la que tendremos que seleccionar el usuario al que queremos enviar el mensaje mediante un desplegable, y luego rellenar el título del mensaje y el cuerpo, para finalmente enviarlo con el botón de enviar.

Sin embargo, como se ha resumido en el apartado anterior, en función del tipo de rol que tenga definido el usuario, este tendrá acceso a unas u otras funciones. Por lo tanto, voy a asignar cada una de las funcionalidades definidas anteriormente en mi aplicación a cada tipo de usuario.

**Funcionalidad por tipo de usuario:**

|  |  |
| --- | --- |
| Tipo de Usuario | Funcionalidades |
| Administrador | Inicio de sesión  Cierre de sesión  Gestión de usuarios  Gestión de Grupos  Actualizar foto de perfil  Visualizar usuario  Actualizar Contraseña  Navegar a Home  Visualización de Noticias  Gestión de Noticias  Gestión de Informes  Gestión de Faltas de Asistencia  Mensajería interna |
| Profesor | Inicio de sesión  Cierre de sesión  Actualizar foto de perfil  Visualizar usuario  Actualizar Contraseña  Navegar a Home  Visualización de Noticias  Gestión de Noticias  Gestión de Informes  Gestión de Faltas de Asistencia  Mensajería interna |
| Alumno | Inicio de sesión  Cierre de sesión  Actualizar foto de perfil  Visualizar usuario  Actualizar Contraseña  Navegar a Home  Visualización de Noticias  Visualización de Informes  Visualización de Faltas de Asistencia  Mensajería interna |
| Padre/Madre | Inicio de sesión  Cierre de sesión  Actualizar foto de perfil  Visualizar usuario  Actualizar Contraseña  Navegar a Home  Visualización de Noticias  Visualización de Informes  Visualización de Faltas de Asistencia  Mensajería interna |

# **03. Estudio de Datos**

Para poder diseñar la base de datos a utilizar en mi aplicación EducaCenter, tuve que tener en cuenta los objetivos y funcionalidades descritas anteriormente. Tras analizar cada uno de los requisitos de mi aplicación, decidí generar un total de 10 tablas, las cuales voy a detallar en las siguientes líneas, con sus correspondientes claves primarias y foráneas. He decidido utilizar el inglés para crear las tablas y el código de mi aplicación (para garantizar una mayor accesibilidad y escalabilidad de mi aplicación), sin embargo, mostraré también la traducción de las tablas al español.

**Tablas:**

* **absences** (**Ausencias**)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Campo | Tipo | Clave | Nulo | Valor por defecto |
| id | int(11) | Primaria | No | AUTO\_INCREMENT |
| student\_id | int(11) | Foránea | No | - |
| teacher\_id | int(11) | Foránea | No | - |
| date | date | - | No | - |
| justified | tinyint(1) | - | Sí | 0 |

* **announcements (Noticias)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Campo | Tipo | Clave | Nulo | Valor por defecto |
| id | int(11) | Primaria | No | AUTO\_INCREMENT |
| title | varchar(255) | - | No | - |
| content | text | - | No | - |
| user\_id | int(11) | Foránea | No | - |
| photo | varchar(255) | - | Sí | NULL |
| created\_at | timestamp | - | No | current\_timestamp() |

* **groups (Grupos)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Campo | Tipo | Clave | Nulo | Valor por defecto |
| id | int(11) | Primaria | No | AUTO\_INCREMENT |
| name | varchar(50) | - | No | - |

* **messages (Mensajes)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Campo | Tipo | Clave | Nulo | Valor por defecto |
| id | int(11) | Primaria | No | AUTO\_INCREMENT |
| sender\_id | int(11) | Foránea | No | - |
| receiver\_id | int(11) | Foránea | No | - |
| subject | varchar(255) | - | Sí | NULL |
| body | text | - | Sí | NULL |
| is\_read | tinyint(1) | - | Sí | 0 |
| created\_at | timestamp | - | No | current\_timestamp() |

* **parents (Padres)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Campo | Tipo | Clave | Nulo | Valor por defecto |
| id | int(11) | Primaria | No | AUTO\_INCREMENT |
| user\_id | int(11) | Foránea | No | - |

* **reports (Imformes)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Campo | Tipo | Clave | Nulo | Valor por defecto |
| id | int(11) | Primaria | No | AUTO\_INCREMENT |
| student\_id | int(11) | Foránea | No | - |
| teacher\_id | int(11) | Foránea | No | - |
| content | text | - | No | - |
| created\_at | timestamp | - | No | current\_timestamp() |

* **schedules (Horarios)**

Esta tabla no es utilizada en mi aplicación, pero he decidido crearla ya para una futura implementación en caso de tener suficiente tiempo.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Campo | Tipo | Clave | Nulo | Valor por defecto |
| id | int(11) | Primaria | No | AUTO\_INCREMENT |
| group\_id | int(11) | Foránea | No | - |
| subject | varchar(100) | - | No | - |
| teacher\_id | int(11) | Foránea | No | - |
| weekday | enum('Monday','Tuesday',...) | - | No | - |
| start\_time | time | - | No | - |
| end\_time | time | - | No | - |
| classroom | varchar(50) | - | Sí | NULL |

* **students (Estudiantes)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Campo | Tipo | Clave | Nulo | Valor por defecto |
| id | int(11) | Primaria | No | AUTO\_INCREMENT |
| user\_id | int(11) | Foránea | No | - |
| group\_id | int(11) | Foránea | Sí | NULL |
| parent\_id | int(11) | Foránea | Sí | NULL |

* **teachers (Profesores)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Campo | Tipo | Clave | Nulo | Valor por defecto |
| id | int(11) | Primaria | No | AUTO\_INCREMENT |
| user\_id | int(11) | Foránea | No | - |
| group\_id | int(11) | Foránea | No | - |

* **users (Usuarios)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Campo | Tipo | Clave | Nulo | Valor por defecto |
| id | int(11) | Primaria | No | AUTO\_INCREMENT |
| name | varchar(100) | - | No | - |
| surname | varchar(100) | - | Sí | NULL |
| email | varchar(150) | Única | Sí | NULL |
| password | varchar(255) | - | Sí | NULL |
| role | enum('admin','teacher',...) | - | No | - |
| profile\_photo | varchar(255) | - | Sí | NULL |
| token | varchar(64) | - | Sí | NULL |
| token\_expires\_at | datetime | - | Sí | NULL |
| created\_at | timestamp | - | No | current\_timestamp() |

**Diagrama Relacional:**

Una captura de pantalla de una computadora

El contenido generado por IA puede ser incorrecto.Esquema generado por phpMyAdmin, donde se pueden observar cada una de las relaciones que se establecen entre las tablas de mi base de datos.

**Relaciones principales entre tablas:**

* Un usuario puede enviar muchos mensajes a otros usuarios, por eso se relaciona la tabla Mensajes con los usuarios mediante las claves foráneas remitente\_id y receptor\_id.
* Un profesor puede generar muchos informes sobre sus alumnos, y un alumno puede tener varios informes. La relación está en la tabla Informes.
* Un profesor también puede registrar faltas de asistencia de sus alumnos en la tabla Faltas de Asistencia, al igual que ocurre con los informes.
* Un alumno y un profesor pueden pertenecer a un solo grupo, de manera que en función del grupo se pueda asociar cada alumno con su profesor.
* Un alumno podrá tener un solo padre, de esta forma el padre tendrá acceso a los informes y faltas de su hijo.
* Un profesor o administrador podrá generar muchas noticias, que serán visibles para todos los usuarios.

# **04. Estudio de Implantación de la Aplicación WEB**

Para poder implantar mi aplicación en la web y conseguir así una aplicación completamente funcional y lista para su uso en producción, he tenido que seguir diferentes fases de implantación, las cuales detallo a continuación.

**Fases de Implantación:**

1. **Desarrollo y Pruebas:**

Para poder trabajar de forma local en mi aplicación web y realizar cada una de las pruebas unitarias necesarias para asegurar el buen funcionamiento de esta, he decido trabajar con XAMPP para desarrollar mi proyecto de backend y con Vite y React para el proyecto de frontend.

Una vez instalado XAMPP, he decidido crear un proyecto en htdocs solo con archivos php. La idea es crear un servidor que actúe como un API REST, con diferentes endpoitns, de manera que en función del tipo de petición que le envíe mi aplicación nos devuelva o modifique los datos correspondientes a la tabla de este endpoint.



Para poder trabajar con una base de datos en local, he aprovechado la herramienta phpMyAdmin que nos ofrece XAMPP, creando así la base de datos EducaCenter, con cada una de las tablas ya descritas en el punto anterior. Además, una vez generada la base de datos, puedo exportar un archivo sql el cual puedo utilizar para poder generar esta misma base de datos en cualquier otro servidor.

Por último, la interfaz gráfica y diferentes funcionalidades que ofrecerá mi aplicación serán posibles gracias al amplio abanico de posibilidades y elementos que nos ofrece React. Para poder compilar y ejecutar este proyecto, he decidido instalarme Vite, el cual simplemente mediante un comando (npm run dev), nos crea un servidor local en el cual podemos visualizar de forma instantánea todos los cambios que vamos generando.

1. **Despliegue:**

Una vez que la aplicación funcione correctamente en nuestro entorno local, podemos comenzar a realizar el despliegue de esta en la web. Para poder desplegar la aplicación existen diferentes opciones, pero al estar desarrollando una aplicación con fines educativos, voy a optar por las opciones gratuitas. Estas alternativas, al ser de coste cero, presentan muchas limitaciones y poco espacio de almacenamiento, pero son suficientes para comprobar todas las funcionalidades a implementar en mi aplicación. A continuación, se muestran cada una de las plataformas utilizadas para desplegar los distintos componentes de mi aplicación:

* + **Servidor API REST:** para poder generar un servidor que actúe como API REST, que no presenten limitaciones ni genere problemas al realizar peticiones de tipo Cross-Origin, me he decantado por Render. Este servicio web, nos permite vincular un repositorio en GitHub con nuestra cuenta, para después desplegarlo. Para poder tener un mayor control sobre nuestro servidor y no tener problemas con las peticiones Cross-Origin, tengo que crear un archivo Docker con la configuración deseada de mi servidor para que Render al detectar este archivo monte la imagen del contenedor y se pueda desplegar correctamente.
  + **Base de Datos:** ya que en local estoy trabajando con phpMyAdmin, he decido utilizar la plataforma freeSqlDatabase para poder tener mi base de datos en la nube. A pesar de que el plan gratuito no ofrece una gran cantidad de almacenamiento, esta plataforma permite gestionar tus bases de datos con phpMyAdmin, lo cual me facilita mucho el manejo e integración de la base de datos previamente creada en local.
  + **Dominio:** por último, para poder acceder a mi aplicación, será necesario desplegar nuestro proyecto con React en una plataforma que nos genere un dominio público y gratuito. En este caso, me he decantado por usar la plataforma Netlify, ya que simplemente con vincular mi proyecto en GitHub a la cuenta creada, automáticamente se despliega mi aplicación en el dominio que yo decida (en mi caso [educacenter.netlify.app](https://educacenter.netlify.app/)).

1. **Seguridad:**

Para conseguir la mayor seguridad posible en mi aplicación, y que sólo los usuarios registrados por el administrador puedan acceder a las diferentes funcionalidades que se ofrecen, voy a incorporar autenticaciones con JWT. Con los Json Web Tokens, lo que se consigue es una mayor seguridad a la hora de acceder a nuestra aplicación, ya que cuando se inicie sesión en la aplicación, se creará un token en base 64 para ese usuario, el cual se enviará en el header de las peticiones que se realicen a nuestro servidor backend. Si este token es correcto, el servidor nos permitirá el acceso, pero si no lo es, no podremos acceder a ningún tipo de información. Además, para incrementar la seguridad, he establecido un periodo de validez de estos tokens de 4 horas, de manera que, tras las 4 horas de creación del token, el usuario deberá volver a iniciar sesión para generar uno nuevo. Estos tokens, serán almacenados en la sesión de nuestro navegador, pero también serán eliminados automáticamente transcurridas 4 horas desde que se almacenaron.

Por otro lado, las contraseñas que se almacenan en mi base de datos, siempre se almacenarán utilizando el encriptado bcrypt, que nos proporciona la función password\_hash de PHP, impidiendo de esta forma la visualización de cualquier contraseña desde nuestra base de datos.

1. **Mantenimiento:**

Si la aplicación resulta de interés para cualquier empresa o centro educativo, se realizarán actualizaciones periódicas, corrección de errores, y nuevas funcionalidades de acuerdo con las necesidades los centros que quieran darle uso

# **05. Código relevante**

EducaCenter es una aplicación web funcional dividida en dos proyectos interconectados que permiten una clara separación entre la lógica de negocio y la interfaz de usuario:

**1. Proyecto Backend (PHP + MySQL):**

El backend está desarrollado en PHP puro y actúa como API REST, gestionando la lógica de negocio y las operaciones sobre la base de datos relacional. Cada funcionalidad descrita anteriormente (usuarios, informes, mensajes, etc.) se corresponde con uno o varios endpoints definidos en archivos PHP independientes. Estos endpoints reciben peticiones HTTP (GET, POST, PUT, DELETE) y devuelven respuestas en formato JSON.

En la carpeta config de este proyecto, nos encontramos con tres archivos: cors.php, database.php y tokensExp.php. En el primero se definen las restricciones cors de nuestra aplicación, es decir, el tipo de peticiones que se aceptan y la dirección url permitida para poder acceder a mis archivos php (que en mi caso es la dirección donde tengo alojada mi app: educacenter.netlify.app). Por otro lado, en databse.php definiremos el usuario y contraseña para acceder a nuestra base de datos, así como la dirección en la que se encuentre, pudiendo también elegir si trabajar con la base de datos local o remota. Por último, en el tercer archivo mencionado, definiremos el tiempo en horas a partir del cual se caducará nuestro token, restringiendo así el acceso a la aplicación, obligando a iniciar de nuevo sesión al usuario por inactividad (por defecto la he establecido a 4 horas pero puede modificarse cambiando la variable $horasExpiracion).

**2. Proyecto Frontend (React + Vite):**

El frontend está desarrollado con React y utiliza Vite como herramienta de desarrollo y empaquetado. Este proyecto gestiona toda la interfaz de usuario de manera dinámica y responsive, utilizando para ello la biblioteca de componentes de interfaz de usuario MUI (Material User Interface), dando un toque moderno y equilibrado en mi aplicación.

La comunicación entre el frontend y el backend se realiza mediante Axios, una biblioteca que permite realizar fácilmente peticiones HTTP. Para ello, se ha creado un archivo de configuración donde se define la URL base del servidor backend, facilitando la modificación de la dirección del servidor sin tener que cambiar el resto del código. Los archivos de configuración para realizar la conexión con nuestro servidor se encuentran dentro de la carpeta api de nuestro proyecto frontend. Aquí nos encontramos con varios archivos en javascript: auth.js, config.js, axios.js, login.js y logout.js. En config.js definimos si vamos a trabajar apuntando a nuestro servidor local o remoto, cambiando la variable option por local o backend, lo cual es de gran utilidad para realizar pruebas en local y en caso de que todo este correcto subir los cambios a remoto cambiando esta variable. Aquí también deberíamos cambiar el valor de la constante API\_BASE en caso de modificar la ruta de nuestro servidor backend. Por otro lado, con axios.js creamos una instancia personalizada axios a la que llamaremos api, a la cual vincularemos los tokens de acceso de forma automática al iniciar la sesión. De esta forma podremos realizar las peticiones a nuestros endpoints de forma más sencilla escribiendo por ejemplo api.get(url), sin tener que completar el cuerpo de la solicitud. Por último, los archivos login y logout.js consisten en dos funciones personalizadas que a su vez hacen uso de las funciones definidas en auth.js para poder autenticarse en el servidor y almacenar los datos del usuario en localstorage. Por seguridad, estos datos he decidido almacenarlos en local por un máximo de dos horas mediante la constante hours definida en App.jsx.

config.js:

// Voy a crear un archivo de configuración para la API

// Si option es "local", la API será localhost/api y si es "backend", la API será localhost/backend/api

const option = "backend";

let API\_BASE;

if (option === "local") {

    // Localhost

    API\_BASE = 'http://localhost/EducaCenter/api';

}

if (option === "backend") {

    // Backend

    API\_BASE = 'https://educacenter-backend.onrender.com/api';

}

export default API\_BASE;

axios.js:

import axios from "axios";

import API\_BASE from "./config";

// Voy a crear un archivo de configuración para realizar las peticiones a la API con axios

// Crear una instancia de Axios

const api = axios.create({

  baseURL: API\_BASE,

  headers: {

    "Content-Type": "application/json",

  },

});

// Interceptor para añadir el token automáticamente si existe

api.interceptors.request.use(

  (config) => {

    const token = localStorage.getItem("EducaCenterToken");

    if (token) {

      config.headers["Authorization"] = `Bearer ${token}`;

    }

    return config;

  },

  (error) => Promise.reject(error)

);

export default api;

auth.js:

// En este archivo se gestionan las funciones de autenticación y sesión del usuario.

// Se utiliza localStorage para almacenar el token y el rol del usuario una vez que se ha autenticado.

// Guarda el token en localStorage

export const login = (token, role, name, surname, id) => {

  localStorage.setItem("EducaCenterToken", token);

  localStorage.setItem("EducaCenterRole", role);

  localStorage.setItem("EducaCenterUser", [name || "Usuario", surname || ""].join(" ").trim());

  localStorage.setItem("EducaCenterId", id);

  localStorage.setItem("EducaCenterLoginTime", Date.now().toString());

};

// Elimina el token del localStorage

export const logout = () => {

  localStorage.removeItem("EducaCenterToken");

  localStorage.removeItem("EducaCenterRole");

  localStorage.removeItem("EducaCenterUser");

  localStorage.removeItem("EducaCenterLoginTime");

  localStorage.removeItem("EducaCenterId");

};

// Comprueba si el usuario está autenticado

export const isAuthenticated = () => {

  return localStorage.getItem("EducaCenterToken") !== null;

};

// Obtiene el token actual (por si hay que usarlo en el headers)

export const getToken = () => {

  return localStorage.getItem("EducaCenterToken");

};

Cada una de las funcionalidades está encapsulada en componentes de React, que se organizan de manera modular según su función (usuarios, grupos, noticias, etc.). Además, se emplea React Router para gestionar la navegación entre las distintas vistas de la aplicación, como se ve en App.jsx.

App.jsx:

import { useState, useEffect } from 'react';

import { BrowserRouter, Routes, Route, Navigate } from 'react-router-dom';

import { ThemeProvider } from '@mui/material/styles';

import CssBaseline from '@mui/material/CssBaseline';

import theme from './components/theme';

import ProtectedRoute from './components/ProtectedRoute';

import { logoutUser } from './api/logout';

import Login from './views/Login';

import Home from './views/Home';

import User from './views/User';

import Messages from './views/Messages';

import Absences from './views/Absences';

import Reports from './views/Reports';

import News from './views/News';

import UsersAdmin from './views/UsersAdmin';

import GroupsAdmin from './views/GroupsAdmin';

// Este componente se encargará de gestionar las rutas de la aplicación.

// Si la ruta es /, se cargará el componente Login, que es la página de login de la aplicación teniendo en cuenta que el usuario no ha iniciado sesión.

// Para acceder al resto de rutas, el usuario debe iniciar sesión, ya que son rutas protegidas.

function App() {

  // Estado para gestionar la autenticación

  const [isAuthenticated, setIsAuthenticated] = useState(false);

  // En la siguiente constante se define el tiempo de expiración del token en horas.

  const hours = 2;

  // Verificar si el token ha expirado al cargar la aplicación

  // Si ha transcurrido más de 2 horas desde el inicio de sesión, se eliminará el token y se recargará la página.

  useEffect(() => {

    const loginTime = localStorage.getItem("EducaCenterLoginTime");

    const expirationTime = 60\*hours \* 60 \* 1000;

    if (loginTime && Date.now() - parseInt(loginTime, 10) > expirationTime) {

      localStorage.clear();

      window.location.reload(); // Opcional: forzar recarga

    }

  }, []);

  // Verificar si hay un token al cargar la aplicación

  useEffect(() => {

    const token = localStorage.getItem('EducaCenterToken');

    if (token) {

      setIsAuthenticated(true);

    }

  }, []);

  // Función para manejar el inicio de sesión exitoso

  const handleLogin = () => {

    setIsAuthenticated(true);

  };

  // Función para manejar el cierre de sesión

  const handleLogout = async () => {

    try {

      await logoutUser(); // Llama al backend para cerrar sesión

    } catch (error) {

      console.error('Error cerrando sesión en backend:', error);

    } finally {

      localStorage.removeItem('EducaCenterToken');

      setIsAuthenticated(false);

    }

  };

  return (

    <ThemeProvider theme={theme}>

    <CssBaseline /> {/\* Normaliza estilos por defecto \*/}

      <BrowserRouter>

        <Routes>

          {/\* Ruta de Login - redirige a Home si ya está autenticado \*/}

          <Route

            path="/login"

            element={

              isAuthenticated ?

                <Navigate to="/" /> :

                <Login onLogin={handleLogin} />

            }

          />

          {/\* Rutas protegidas \*/}

          <Route

            path="/"

            element={

              <ProtectedRoute isAuthenticated={isAuthenticated}>

                <Home onLogout={handleLogout} />

              </ProtectedRoute>

            }

          />

          <Route

            path="/usuario"

            element={

              <ProtectedRoute isAuthenticated={isAuthenticated}>

                <User onLogout={handleLogout} />

              </ProtectedRoute>

            }

          />

          <Route

            path="/mensajes"

            element={

              <ProtectedRoute isAuthenticated={isAuthenticated}>

                <Messages onLogout={handleLogout} />

              </ProtectedRoute>

            }

          />

          <Route

            path="/faltas"

            element={

              <ProtectedRoute isAuthenticated={isAuthenticated}>

                <Absences onLogout={handleLogout} />

              </ProtectedRoute>

            }

          />

          <Route

            path="/informes"

            element={

              <ProtectedRoute isAuthenticated={isAuthenticated}>

                <Reports onLogout={handleLogout} />

              </ProtectedRoute>

            }

          />

          <Route

            path="/noticias"

            element={

              <ProtectedRoute isAuthenticated={isAuthenticated}>

                <News onLogout={handleLogout} />

              </ProtectedRoute>

            }

          />

          <Route

            path="/admin/usuarios"

            element={

              <ProtectedRoute isAuthenticated={isAuthenticated}>

                <UsersAdmin onLogout={handleLogout} />

              </ProtectedRoute>

            }

          />

          <Route

            path="/admin/grupos"

            element={

              <ProtectedRoute isAuthenticated={isAuthenticated}>

                <GroupsAdmin onLogout={handleLogout} />

              </ProtectedRoute>

            }

          />

          {/\* Redirigir cualquier otra ruta a la página principal o login según autenticación \*/}

          <Route

            path="\*"

            element={<Navigate to={isAuthenticated ? "/" : "/login"} />}

          />

        </Routes>

      </BrowserRouter>

    </ThemeProvider>

  );

}

export default App;

También se ha implementado una gestión de estados eficiente utilizando useState y useEffect, y en algunos casos se han utilizado ventanas modales (ventanas emergentes) para realizar acciones como crear, editar o eliminar registros sin abandonar la página actual mediante el compoenente Dialog de MUI. Otros componentes muy utilizados de MUI han sido CircularProgress, Skeleton, SnackBar, Button o Table.

# **06. Conclusión**

En conclusión, EducaCenter tiene muchas funcionalidades y puede crecer para adaptarse a diferentes tipos de centros educativos, desde guarderías hasta escuelas primarias o incluso secundarias. Además, el diseño de la aplicación será lo más flexible y escalable posible, para permitir la expansión a nuevas funcionalidades como la gestión de horarios y asignaturas en un futuro próximo.