Sistema de Fichaje

Proyecto Integrado

Desarrollo de Aplicaciones Web

Alejandro González Cervantes

CONTENIDO

INTRODUCCIÓN

¿EN QUÉ CONSISTE EL PROYECTO?

Será una aplicación web que permitirá a los usuarios registrar su hora de entrada y de salida. Cada usuario podrá fichar al iniciar y finalizar su jornada y el sistema almacenará los registros en una base de datos para su posterior consulta.

MOTIVACIÓN Y JUSTIFICACIÓN DE PROYECTO

Este proyecto surge de la necesidad de dejar constancia de la entrada y salida de los trabajadores, según el Real Decreto-ley 8/2019, en el que todas las empresas deberán de llevar un registro diario de la jornada laboral de sus empleados.

Los beneficios que aportará la utilización de este sistema de fichaje serán:

- Poder cumplir las exigencias del Real Decreto-ley 8/2019, debiendo de conservar los registros, como mínimo, por un periodo de 4 años.
- Facilitar el control de la asistencia del personal de la empresa, minimizando así errores propios de medios físicos.
- Permitir ver rápidamente un resumen de horas trabajadas para facilitar la labor de diferentes departamentos, como el de Recursos Humanos.

Se establecerá como objetivo la creación de un sistema que registre la entrada y salida de usuarios de forma digital para la optimización del control de asistencia y el cálculo de horas trabajadas.

FUNCIONALIDADES PRINCIPALES

REGISTRO DE ENTRADA Y SALIDA

El usuario registrará el evento (ya sea entrada o salida) en el momento en el que haga clic en el botón correspondiente.

VISUALIZACIÓN DE LOS REGISTROS

Cada usuario podrá ver un historial de los registros que haya realizado. Además, el usuario asignado a RRHH podrá ver los registros de cualquiera de los usuarios.

JUSTIFICAR FALTA

El sistema permitirá subir un justificante asociado a unos días para justificar una ausencia en algunos días especificados.

SOLUCIONES QUE SE APORTAN

Este sistema aportará una solución digital y accesible para el control de horarios dentro de una empresa, eliminando la necesidad de un soporte físico y alterable de cara a posibles inspecciones de trabajo, además de la agilización de labores de revisión de incidencias horarias por parte del departamento de RRHH.

ANÁLISIS

REQUISITOS FUNCIONALES DEL SISTEMA

Requisitos funcionales del sistema (¿Qué hace nuestro sistema? ¿Qué funcionalidades tiene?)

QUÉ HACE NUESTRO SISTEMA

Nuestro sistema gestiona usuarios, entradas y ausencias de empleados dentro de una organización. Permite que los usuarios se autentifiquen mediante credenciales y accedan a funcionalidades protegidas. Los administradores pueden gestionar los usuarios del sistema, mientras que los usuarios estándar pueden registrar su entrada al trabajo y registrar ausencias.

FUNCIONALIDADES QUE TIENE

- Autenticación y autorización: Permite a los usuarios iniciar y cerrar sesión con validación de credenciales.
- Gestión de usuarios: Los administradores pueden crear, eliminar y visualizar usuarios registrados.
- Acceso a datos personales: Los usuarios autenticados pueden ver y modificar su propia información personal.
- Registro de entrada (Fichar): Los usuarios pueden registrar su hora de entrada al trabajo.
- Registro de ausencia (Justificantes): Los usuarios pueden registrar ausencias por motivos personales.
- Rutas protegidas: Solo los usuarios autenticados pueden acceder a rutas diferentes al LogIn.

DIAGRAMAS UML

DIAGRAMA DE CASOS DE USO

Este diagrama organiza los casos de uso en función de los roles y muestra las interacciones entre ellos:

Usuario identificado: el usuario tiene acceso a funcionalidades como:

- Logarse: una vez hecho eso, también podrá deslogarse
- Fichar: registrar su entrada o salida.
- Justificantes: enviar o consultar justificantes de ausencia, con la posibilidad de adjuntar un archivo.
- Sus datos: ver y editar sus propios datos personales.

Usuario RRHH: Además de las funcionalidades del usuario identificado, puede:

- Crear nuevo usuario
- Listar de usuarios: ver una lista de todos los usuarios registrados.

Este diagrama organiza los casos de uso en función de los roles y muestra las interacciones entre ellos.

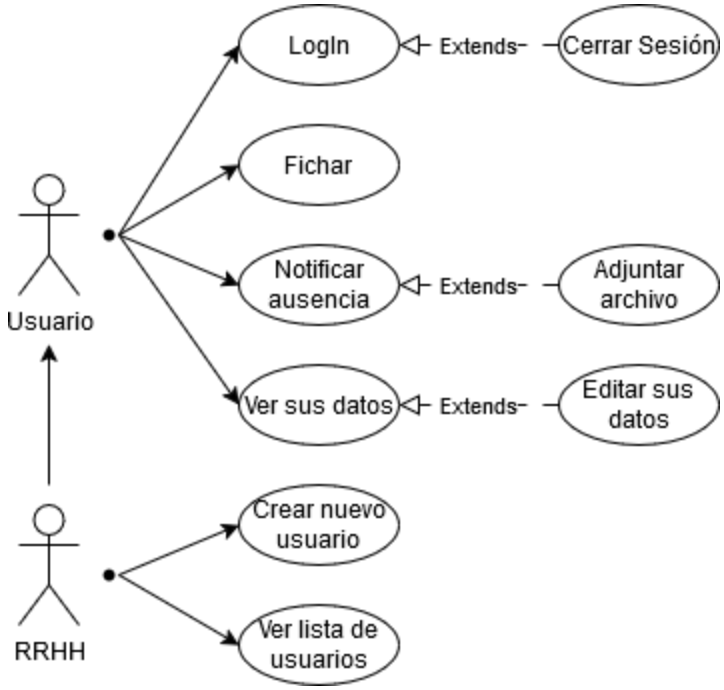


DIAGRAMA DE DESPLIEGUE

Cliente (Navegador Web):

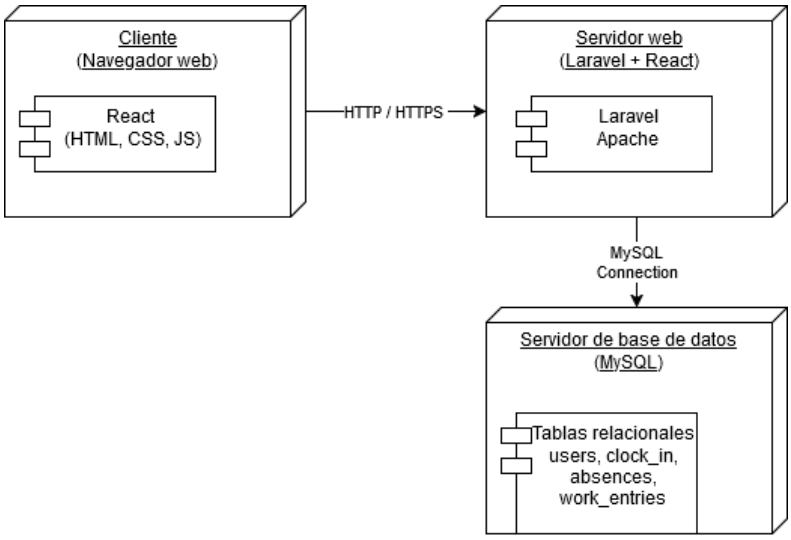
- El dispositivo del usuario que accede a la aplicación mediante un navegador web.
- Hospeda la interfaz React que interactúa con el servidor backend.

Servidor Web:

- Un servidor que aloja la aplicación Laravel para manejar las solicitudes del cliente y servir respuestas.
- Software Apache configurado para manejar peticiones HTTP/HTTPS.
- También sirve los archivos estáticos del frontend React.

Servidor de Base de Datos:

- Un servidor dedicado que almacena las tablas de la base de datos (users, clock_in, absences, work_entries).
- Usa MySQL como sistema de gestión de bases de datos relacionales.



DISEÑO

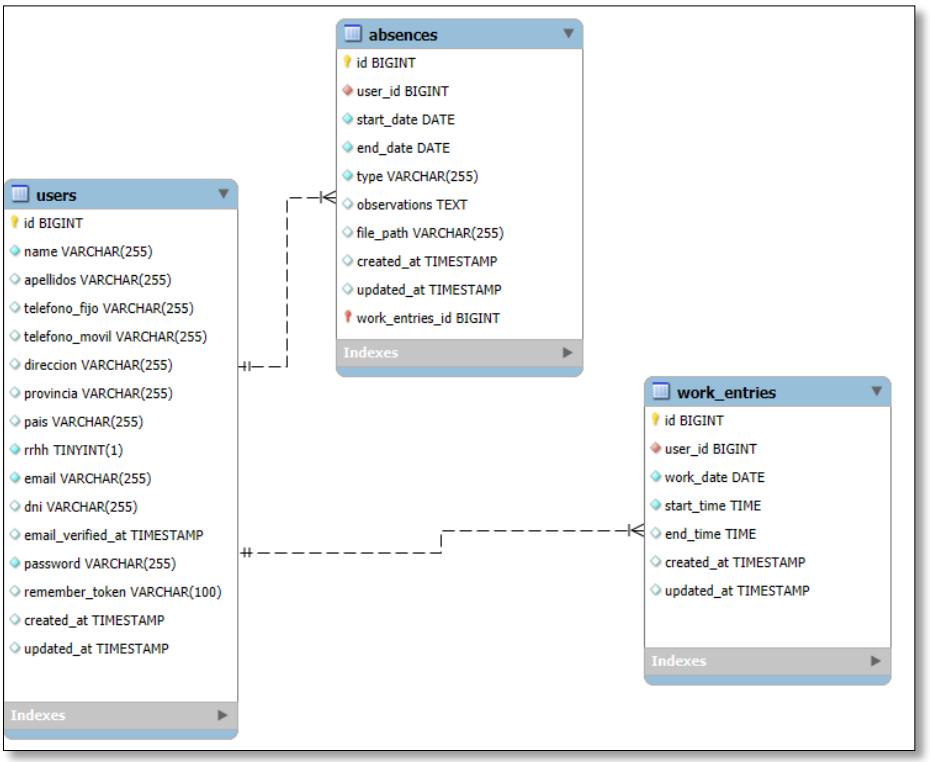
DISEÑO DE LA BASE DE DATOS

La base de datos utilizada en este sistema es relacional, ya que estamos gestionando datos estructurados como usuarios, entradas y ausencias, lo que se adapta bien a un esquema tabular.

TABLAS

Las tablas clave de nuestra base de datos son las siguientes:

- **Usuarios** (users): contiene la información básica de los usuarios, incluyendo los campos para autenticación y gestión de datos personales.
- **Entradas** (work_entries): almacena las entradas registradas por los usuarios. Cada usuario puede registrar múltiples entradas.
- **Ausencias** (absences): almacena las ausencias registradas por los usuarios. Cada usuario puede registrar múltiples ausencias.



MOCKUP DE BAJA FIDELIDAD

Vamos a dividirla en tres grupos, dependiendo del tipo de usuario que lo vea.

USUARIO SIN IDENTIFICAR

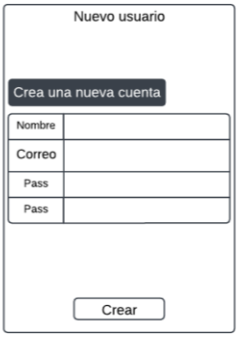
Es la página que verá el usuario nada más llegar.

USUARIO IDENTIFICADO

Estas páginas serán a las que tengan acceso los usuarios identificados correctamente

USUARIO RRHH

A estas páginas tendrá acceso, además de las anteriores, los usuarios que sean RRHH

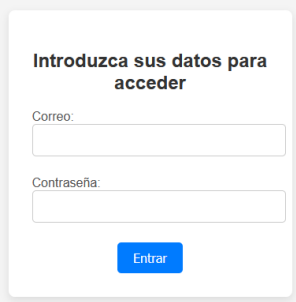


MOCKUP DE ALTA FIDELIDAD

Al igual que con el MockUp de baja fidelidad, vamos a presentar las mismas estructuras, esta vez intentando tener un diseño más serio y corporativo.

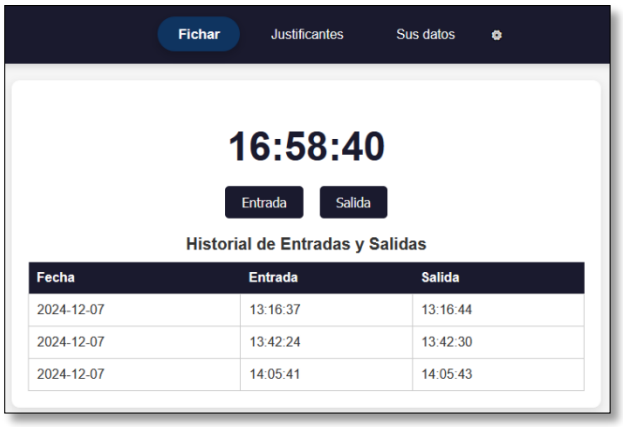
USUARIO SIN IDENTIFICAR

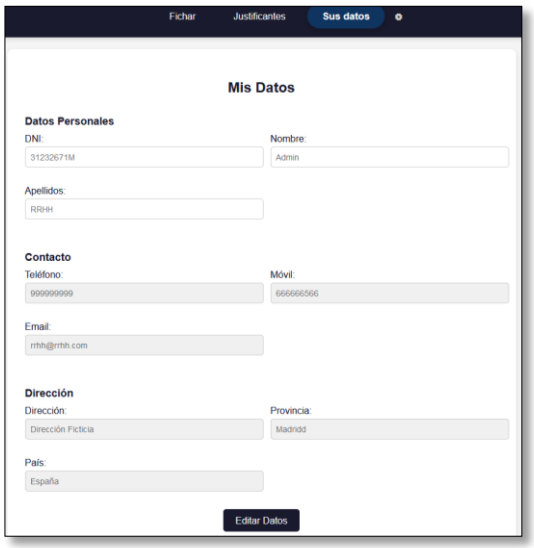
Es la página que verá el usuario nada más llegar.



USUARIO IDENTIFICADO

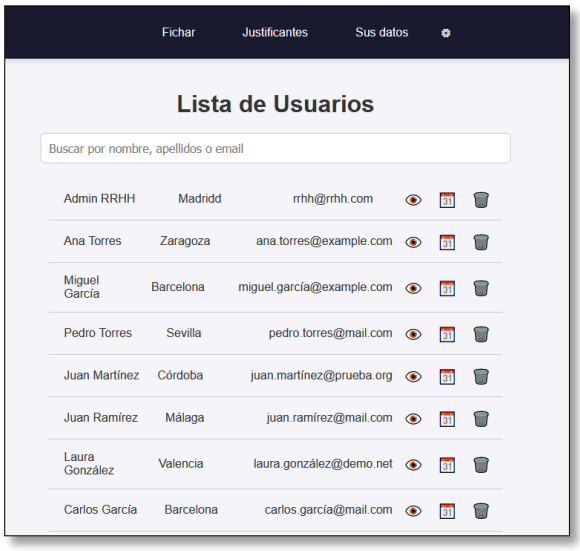
Estas páginas serán a las que tengan acceso los usuarios identificados correctamente

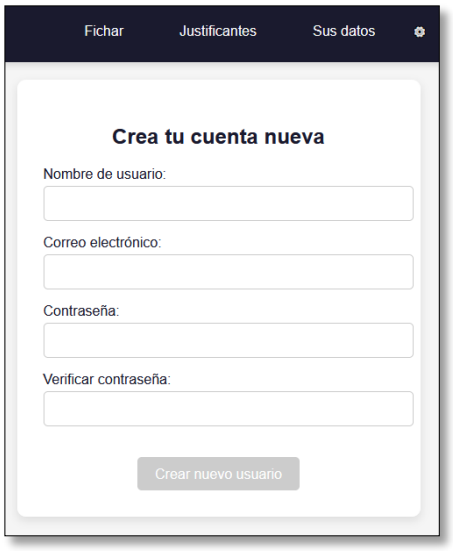




USUARIO RRHH

A estas páginas tendrá acceso, además de las anteriores, los usuarios que sean RRHH





IMPLEMENTACIÓN

Para mantener el sistema sencillo y ligero, y que pueda ser utilizado en una amplia variedad de empresas, se han seleccionado tecnologías accesibles y fáciles de implementar. Este proyecto desarrolla una aplicación web full-stack, donde el **frontend** se centra en proporcionar una interfaz de usuario interactiva y dinámica, mientras que el **backend** gestiona la lógica de negocio, la autenticación y el almacenamiento de datos en la base de datos.

FRONTEND

El frontend constituye el punto de interacción principal para el usuario, con una interfaz diseñada para facilitar el registro de entradas y salidas mediante una experiencia visual intuitiva y accesible.

REACT

React [3] es una biblioteca de JavaScript [7] creada para desarrollar interfaces de usuario dinámicas, eficientes y escalables. Es ampliamente reconocida por su enfoque basado en **componentes reutilizables** y su capacidad para gestionar estados de manera declarativa. Gracias a su diseño, React permite a los desarrolladores construir aplicaciones que respondan de forma fluida a las interacciones del usuario, ofreciendo un rendimiento óptimo incluso en aplicaciones complejas.

CARACTERÍSTICAS DESTACADAS

- **Basado en Componentes:** React organiza la interfaz de usuario en componentes independientes, encapsulando en cada uno su propia lógica, estructura y estilo. Este enfoque modular no solo facilita la reutilización del código, sino que también mejora la claridad y el mantenimiento a medida que la aplicación crece.

- **Virtual DOM:** React utiliza un DOM (Document Object Model) virtual, una representación ligera del DOM real en memoria. Este sistema identifica los cambios necesarios y actualiza únicamente las partes afectadas, lo que reduce el tiempo de renderizado y mejora la experiencia del usuario al evitar recargas completas de la página.

- **Declarativo:** Con React, los desarrolladores definen lo que desean que la interfaz muestre en función del estado de la aplicación, y React se encarga de realizar los cambios necesarios automáticamente. Esto hace que el código sea más predecible, fácil de depurar y sencillo de entender.

- **Gestión del Estado:** React proporciona herramientas para manejar estados locales y globales, permitiendo una sincronización efectiva entre los datos y la interfaz. Esto asegura que cualquier cambio en los datos se refleje automáticamente en la vista, eliminando la necesidad de actualizaciones manuales.

- **Ecosistema Extensible:** React se integra fácilmente con librerías y herramientas complementarias, como sistemas de enrutamiento, gestión de estados globales y manejo de solicitudes HTTP. Este ecosistema permite a los desarrolladores personalizar la solución a las necesidades específicas del proyecto.

VENTAJAS

- Rendimiento Óptimo con el uso del Virtual DOM y la renderización eficiente, que permiten que las aplicaciones construidas con React sean rápidas y responsivas, incluso cuando manejan grandes volúmenes de datos o interacciones complejas.

- Flexibilidad y Escalabilidad, ya que se adapta a proyectos de cualquier tamaño, desde aplicaciones simples hasta sistemas empresariales complejos. Su diseño modular permite una escalabilidad natural sin necesidad de reestructurar el código base.

- La reutilización de los componentes reduce la duplicación de código y garantizan una interfaz consistente. Esto también mejora el mantenimiento a largo plazo, ya que los cambios en un componente se reflejan en todas las partes donde se utiliza.

- Soporte y Comunidad activos, que continuamente mejora su ecosistema y proporciona recursos para resolver problemas comunes. Además, es respaldado por Facebook, lo que garantiza su continuidad y evolución

HTML5

HTML5 es la tecnología que proporciona la estructura de las páginas de la aplicación. Ofrece elementos semánticos como **<header>**, **<footer>** y **<section>,** que mejoran la accesibilidad y la comprensión del código. Además, es compatible con todos los navegadores modernos y no requiere configuraciones adicionales, lo que asegura la portabilidad del sistema.

CSS3

CSS3 [8] se utiliza para estilizar la interfaz, asegurando un diseño visualmente atractivo y responsive. Esto incluye:

- **Estilos personalizables**: Cada empresa puede adaptar colores, tipografía y espaciado a través de hojas de estilo específicas.

- **Responsive Design**: Mediante técnicas como media queries, la interfaz se adapta a dispositivos móviles y de escritorio.

BACKEND

El backend se encarga de la lógica del sistema y el almacenamiento seguro de datos. Gestiona tareas esenciales como la autenticación, el procesamiento de solicitudes de usuario y las interacciones con la base de datos.

LARAVEL

Laravel [5] es el framework principal para el desarrollo del backend de esta aplicación. Está construido en **PHP** [6], un lenguaje de programación del lado del servidor ampliamente utilizado por su simplicidad, flexibilidad y soporte en casi cualquier entorno de servidor web.

Laravel sigue el patrón de arquitectura **MVC (Modelo-Vista-Controlador)**, que separa claramente las responsabilidades del código, lo que mejora la organización y mantenibilidad de las aplicaciones. Esto significa que representa los datos y la lógica de negocio, gestiona la presentación de los datos y actúa como intermediario entre las vistas y los modelos.

CARACTERÍSTICAS DESTACADAS

- **Eloquent ORM:** es el sistema ORM (Object-Relational Mapping) de Laravel que permite interactuar con la base de datos de manera intuitiva usando modelos en lugar de realizar consultas SQL manuales. Esto mejora la legibilidad y reduce errores, además de evitar la inyección de código.
- **Sistema de Rutas Avanzado:** Laravel ofrece un sistema de enrutamiento sencillo y expresivo para definir cómo las solicitudes HTTP son manejadas por la aplicación. Esto permite separar claramente las responsabilidades y facilita la modificación futura de las rutas.
- **Migraciones y Seeders:** Las migraciones permiten gestionar cambios en la estructura de la base de datos, como la creación de tablas o columnas. Los seeders, por su parte, llenan la base de datos con datos iniciales o de prueba.

VENTAJAS

- Curva de aprendizaje amigable gracias a su documentación exhaustiva y comunidad activa.
- Ecosistema robusto, que incluye herramientas como Laravel Forge (para despliegue) y Laravel Nova (para administración).
- Seguridad: Laravel ofrece protección contra ataques comunes como CSRF, SQL Injection y XSS.

MYSQL

MySQL [4] es el sistema de gestión de bases de datos elegido para este proyecto. Su rendimiento y escalabilidad lo convierten en una solución ideal para almacenar información de usuarios y registros de entradas y salidas. Sus principales ventajas son:

- **Consultas optimizadas**: Soporta operaciones complejas con eficiencia.
- **Flexibilidad**: Escalabilidad horizontal y vertical para futuras ampliaciones.
- **Compatibilidad**: Integra perfectamente con Laravel mediante Eloquent ORM.

ASPECTOS DEL CÓDIGO RELEVANTES

FRONTEND

En el archivo Users.jsx, por ejemplo, se utiliza React para gestionar el estado y la lógica de la aplicación. Algunos aspectos clave incluyen:

Uso del Hook useState: Este hook es utilizado para gestionar el estado de la aplicación. Se utilizan diferentes estados como users, filteredUsers, searchTerm, selectedUser, etc. Este enfoque asegura que la interfaz se actualice de forma reactiva cuando los datos cambian.

```
 const [users, setUsers] = useState([]);
  const [filteredUsers, setFilteredUsers] = useState([]);
  const [searchTerm, setSearchTerm] = useState("");
```

Manejo de Peticiones HTTP con fetch: Se realiza una petición a la API para obtener los datos de los usuarios. Se utiliza el método fetch para realizar solicitudes GET a la URL de los usuarios.

```
const response = await fetch("http://localhost:8000/users");
```

Mapeo de Datos para la Vista: Después de obtener los usuarios desde el backend, se mapean los datos para mostrar una lista de usuarios en la interfaz.

```
{filteredUsers.map((user) => (
<div style={styles.tableRow} key={user.id}>
            <div style={styles.tableCell}>
              {`${user.name} ${user.apellidos}`}
            </div>
))}
```

Modal para Edición y Vista de Usuarios: Con las vistas modal se facilita la visualización de los datos de una manera más eficiente. Estos modales se controlan con los estados showViewModal y showEditModal, mostrando y ocultando las vistas correspondientes.

```
{showViewModal && (
  <div>
   <h2>Detalles del Usuario</h2>
   <p>{selectedUser.name}{" "}{selectedUser.apellidos}</p>
  </div>
)}
```

BACKEND

En el backend, se gestiona la autenticación y protección de las rutas mediante **Laravel Sanctum**, que es un paquete de Laravel que proporciona un sistema de autenticación basado en tokens para aplicaciones SPA (Single Page Applications), móviles y APIs. Ofrece dos enfoques principales:

- **Autenticación basada en cookies**: Ideal para aplicaciones SPA que comparten dominio con el backend, utilizando sesiones para gestionar la autenticación.
- **Tokens personales**: Permite emitir tokens de acceso para autenticar solicitudes realizadas por clientes externos (como aplicaciones móviles o scripts).

En este proyecto, se utiliza el enfoque de **tokens personales**, que son generados para cada usuario tras un proceso de autenticación exitoso. Estos tokens pueden incluir permisos personalizados para controlar el acceso a recursos específicos.

CREACIÓN DE UN USUARIO Y GENERACIÓN DE UN TOKEN

En el archivo **UserController.php**, el método store gestiona la creación de nuevos usuarios y la asignación de un token de autenticación personal. Este proceso incluye los siguientes pasos:

1. **Validación de Datos**

   Antes de procesar la solicitud, se valida que los datos enviados por el cliente sean correctos y cumplan con las reglas definidas. Esto asegura que solo se procesen solicitudes válidas, evitando datos mal formados.

2. **Creación del Usuario**

   Se utiliza el modelo User junto con la función create() para insertar un nuevo registro en la base de datos. Este método toma un array asociativo con los datos del usuario, como el nombre, correo electrónico y contraseña. Algunas claves importantes:

   - La contraseña se encripta utilizando Hash::make(), una función de Laravel que aplica un algoritmo de hashing seguro.
   - Campos opcionales como apellidos, telefono_fijo, direccion, entre otros, se procesan usando el operador de fusión (??) para asignar valores por defecto si no se proporcionan.

```php
$user = User::create([
  'name' => $validatedData['name'],
  'email' => $validatedData['email'],
  'password' => Hash::make($request->password),
  'apellidos' => $validatedData['apellidos'] ?? null,
  'telefono_fijo' => $validatedData['telefono_fijo'] ?? null,
  'telefono_movil' => $validatedData['telefono_movil'] ?? null,
  'direccion' => $validatedData['direccion'] ?? null,
  'provincia' => $validatedData['provincia'] ?? null,
  'pais' => $validatedData['pais'] ?? null,
  'dni' => $validatedData['dni'] ?? null
]);
```

3. **Generación del Token**

   Una vez creado el usuario, se genera un **token personal** mediante el método createToken() del modelo User. Este método devuelve un objeto de tipo PersonalAccessToken, del cual se extrae el token como una cadena de texto utilizando plainTextToken.

```
$token = $user->createToken('auth_token')->plainTextToken;
```

4. **Devolución de la Respuesta**

   Finalmente, el token generado se devuelve en la respuesta al cliente, junto con cualquier información relevante del usuario. Este token se incluye en las cabeceras de las solicitudes posteriores para autenticar al usuario.

VALIDACIÓN DE DATOS EN EL MÉTODO LOGIN

En el método login, se valida y autentica al usuario para garantizar que solo aquellos con credenciales correctas puedan acceder a los recursos protegidos.

```
$credentials = $request->validate([
  'email' => 'required|string|email',
  'password' => 'required|string',
]);

if (!Auth::attempt($credentials)) {
  return response()->json(['message' => 'Credenciales incorrectas'],
401);
}
```

1. **Validación de Datos**

   El campo email sea obligatorio (required), esté en formato válido (email) y sea una cadena (string).

   La contraseña (password) también sea obligatoria y de tipo cadena.

2. **Intento de Autenticación**

   Auth::attempt($credentials) verifica si las credenciales coinciden con las registradas en la base de datos. Si la autenticación falla, se devuelve un mensaje de error con el estado HTTP 401 (No autorizado).

3. **Seguridad y Protección**

   Este enfoque evita accesos no autorizados, bloqueando usuarios con credenciales inválidas y devolviendo mensajes genéricos para no exponer detalles del sistema.

Este enfoque garantiza que solo los usuarios con credenciales válidas puedan acceder a los recursos protegidos.

CONCLUSIONES

Este proyecto ha logrado implementar una solución eficiente y escalable para la gestión de usuarios, entradas de trabajo y ausencias utilizando un enfoque full-stack. El backend, desarrollado con Laravel, ha permitido gestionar la autenticación mediante Sanctum y la interacción con la base de datos a través de Eloquent ORM, mientras que el frontend, con React, proporciona una interfaz dinámica y fluida que facilita la interacción del usuario con el sistema. La arquitectura modular y la separación de responsabilidades entre el cliente y el servidor han permitido mantener el código limpio y escalable.

A lo largo del desarrollo, se ha mantenido un enfoque en la simplicidad, asegurando que las solicitudes sean autenticadas. La integración directa con fetch y el uso de Laravel Sanctum para la gestión de sesiones y tokens ha sido eficaz y adecuado a las necesidades del proyecto. El resultado es una aplicación robusta y fácil de mantener que cubre los requisitos funcionales del sistema de manera efectiva.

TRABAJOS FUTUROS

En el futuro, se planea mejorar la funcionalidad de la aplicación con nuevas características que enriquecerán la experiencia de usuario y ampliarán las capacidades del sistema.

Visualización desde la web de los archivos subidos por los usuarios: Se implementará una funcionalidad que permita a los usuarios visualizar los archivos que han subido a través de la plataforma. Esto incluirá la integración de una vista de archivos adjuntos (por ejemplo, justificantes de ausencia, documentos de recursos humanos, etc.) dentro de su perfil o en secciones específicas de la aplicación, mejorando la accesibilidad y gestión de documentos.

Integrar fichaje con geolocalización: Se añadirá una característica de geolocalización para el sistema de fichaje, permitiendo a los usuarios registrar su entrada y salida desde su ubicación física. Esto garantizará que los fichajes solo se realicen cuando el usuario esté en el lugar adecuado, añadiendo una capa extra de seguridad y precisión al sistema de control de tiempo.

Fichaje con huella digital (biométrico): Se integrará un sistema de fichaje mediante biometría, específicamente huella digital, para garantizar una mayor precisión y seguridad en el registro de entrada y salida de los usuarios. Esto permitirá eliminar posibles errores o manipulaciones, mejorando la autenticidad del proceso de fichaje y la integridad de los registros de tiempo.

Mejorar el nivel de seguridad: por problemas técnicos con la validación de tokens CSRF que afectaron el funcionamiento de las solicitudes, se decidió inhabilitar temporalmente esta protección en el archivo app.php. Esto permitió avanzar con el desarrollo y las pruebas sin interrupciones, aunque incrementó el riesgo de vulnerabilidades. Se recomienda reactivar esta validación una vez solucionados los problemas, especialmente antes de desplegar el sistema en producción, para garantizar la seguridad del proyecto.

Estas mejoras contribuirán a hacer la aplicación más robusta y adaptada a las necesidades de los usuarios y de la empresa.

REPOSITORIO

En proyecto está colgado en GIThub en el siguiente enlace:
https://github.com/alegoncer/proyectointegradoDAW.git

También se puede acceder a través del QR de esta página

REFERENCIAS

1. **Gobierno de España**. (2019). Real Decreto-ley 8/2019, de 8 de marzo, de medidas urgentes de protección social y de lucha contra la precariedad laboral en la jornada de trabajo. Boletín Oficial del Estado, núm. 61, de 12 de marzo de 2019, páginas 24253 a 24265. Recuperado de https://www.boe.es/buscar/doc.php?id=BOE-A-2019-3481

2. **Laravel**. (s.f.). *Laravel Documentation*. Laravel. Recuperado de https://laravel.com/docs

3. **React**. (s.f.). *React Documentation*. React. Recuperado de https://reactjs.org/docs/getting-started.html

4. **MySQL**. (s.f.). *MySQL Documentation*. Oracle. Recuperado de https://dev.mysql.com/doc/

5. **Laravel Sanctum**. (s.f.). *Laravel Sanctum Documentation*. Laravel. Recuperado de https://laravel.com/docs/11.x

6. **PHP**. (s.f.). *PHP Manual*. The PHP Group. Recuperado de https://www.php.net/manual/en/

7. **JavaScript (MDN Web Docs)**. (s.f.). *JavaScript Documentation*. Mozilla. Recuperado de https://developer.mozilla.org/en-US/docs/Web/JavaScript

8. **CSS**. (s.f.). *CSS Documentation*. W3C. *Recuperado* de https://www.w3.org/TR/css3-roadmap/

9. **Diagramas UML.** (*n.d.). Diagramas UML: Tutoriales y ejemplos de UML. *Recuperado de* **https://diagramasuml.com/**