



UNIVERSAE
Instituto Superior de FP

DESARROLLO DE UN ERP

PROYECTO DE ELABORACIÓN DE UNA APLICACIÓN WEB

**CICLO FORMATIVO DE GRADO SUPERIOR DE DESARROLLO DE
APLICACIONES WEB**

ALUMNO: MARIO DIEGUEZ SALAMANCA

TUTOR: JOSE LUIS FERRER

Curso 2023-2024

ÍNDICE

| | |
|---|-----------|
| 1. Resumen | 3 |
| 2. Abstract..... | 4 |
| 3. Introducción..... | 5 |
| 4. Descripción y Objetivos..... | 6 |
| 5. Contenidos..... | 7 |
| 6. Metodología y planificación..... | 10 |
| 6.1. Inicio del proyecto | 10 |
| 6.2. Diseño de la aplicación | 10 |
| 6.3. Desarrollo de la aplicación..... | 11 |
| 6.4. Fase final del proyecto | 12 |
| 6.5. Tiempo invertido en el proyecto | 13 |
| 7. Desarrollo..... | 14 |
| 7.1. Instalación y configuración del entorno | 14 |
| 7.2. Modelos, Migraciones, Controladores, Seeders, Web.php, .env | 17 |
| 7.3. Flujo de ejecución..... | 19 |
| 8. Conclusiones y proyección a futuro..... | 22 |
| 9. Bibliografía..... | 23 |
| 10. Anexos..... | 24 |

1. Resumen

La finalidad del proyecto es desarrollar y mostrar una demo de un sistema de gestión empresarial (ERP) en una red local. Los ERP cada vez son más demandados por las empresas a causa de la creciente necesidad que tienen de gestionar de manera digital aspectos como el acceso y gestión de datos, estudios financieros y un aumento de la productividad, entre otras cosas. Por lo tanto, un ERP busca mejorar la organización, escalabilidad y costes que todos estos procesos conllevan. Para desarrollar este tipo de sistemas hay que tener conocimientos previos sobre las necesidades y/o gestiones que desean cubrir las empresas. Existen dos tipos de software, a medida y comercial. En las aplicaciones a medida el cliente es un particular y se encarga de diseñar (con ayuda de los desarrolladores) la aplicación que desea. Por otro lado, si se tiene la idea de desarrollar un ERP más genérico destinado a distintos sectores empresariales, estaríamos hablando de un software comercial. Este último, debe cumplir con un mínimo de necesidades comunes como pueden ser la gestión de inventario, ventas, compras, proveedores, clientes, empleados, etc., pero con la posibilidad de ampliar estos sistemas de manera independiente a cada empresa, pueden ser módulos ya prediseñados o módulos desarrollados de a petición del cliente. Este proyecto está orientado a un desarrollar un ERP a medida para una empresa farmacéutica, desarrollado en un equipo con sistema operativo Ubuntu, mediante la implementación de una pila LAMP (Linux, Apache, MySQL y PHP). La principal herramienta de programación empleada para la parte del servidor (backend) es uno de los frameworks de PHP (Laravel) más extendidos, que trabaja de manera conjunta con una base de datos relacional (MariaDB). En la parte del cliente (frontend) se utiliza una librería de JavaScript (React) combinado con el lenguaje de marcas HTML y hojas de estilo en cascada (CSS). Estas dos tecnologías se despliegan dentro del mismo espacio de trabajo o monolito gracias a Inertia, que es una biblioteca de enrutamiento del lado del cliente, esto lo consigue a través de adaptadores que comunican el lado del servidor con el lado del cliente sin tener que hacer uso de una API. La mayoría de la carga que tiene que soportar la aplicación se procesa en el navegador.

Palabras Clave: *erp, lamp, laravel, react, inertia.*

2. Abstract

The purpose of the project is to develop and show a demo of a business management system (ERP) on a local network. ERPs are increasingly in demand by companies due to the growing need they have to digitally manage aspects such as data access and management, financial studies and increased productivity, among other things. Therefore, an ERP seeks to improve the organization, scalability and costs that all these processes entail. To develop this type of systems, you must have prior knowledge about the needs and/or procedures that companies wish to cover. There are two types of software, custom and commercial. In custom applications, the client is an individual and is responsible for designing (with the help of developers) the application they want. On the other hand, if you have the idea of developing a more generic ERP intended for different business sectors, we would be talking about commercial software. The latter must meet a minimum of common needs such as inventory management, sales, purchases, suppliers, customers, employees, etc., but with the possibility of expanding these systems independently of each company, they can be modules. already pre-designed or modules developed at the client's request. This project is aimed at developing a custom ERP for a pharmaceutical company, developed on a computer with Ubuntu operating system, through the implementation of a LAMP stack (Linux, Apache, MySQL and PHP). The main programming tool used for the server part (backend) is one of the most widespread PHP frameworks (Laravel), which works together with a relational database (MariaDB). On the client side (frontend) a JavaScript library (React) is used combined with HTML markup language and cascading style sheets (CSS). These two technologies are deployed within the same workspace or monolith thanks to Inertia, which is a client-side routing library, this is achieved through adapters that communicate the server side with the client side without having to do use of an API. Most of the load that the application has to bear is processed in the browser.

Keywords: *erp, lamp, laravel, react, inertia.*

3. Introducción

Los ERP (Sistemas de Planificación de Recursos Empresariales) como su nombre indica, permiten gestionar y manejar empresas ayudando a su administración y toma de decisiones. Un ERP va a tener toda la información centralizada y automatizada en un único software, esto es mucho más ágil y organizado que tener datos en papel a la hora de hacer estudios y consultas.

Se tiene previsto un incremento exponencial de la demanda de ERPs por parte de las empresas. Estas contemplan los beneficios que conlleva adoptar estos sistemas, entre otras cosas, permiten aumentar la productividad, realizar informes acelerados y ofrecer al personal laboral una herramienta ágil para desempeñar sus tareas.

El ERP expuesto en esta memoria está desarrollado como ERP a medida, ya que no está disponible la opción de instalar módulos y eliminarlos sin ayuda de los desarrolladores. Crear e implementar estos módulos desde cero, supone más costes y tiempo que un ERP comercial, sin embargo, elijo realizar un ERP a medida porque es una herramienta personalizada.

Esta demo se ha desarrollado utilizando dos de las librerías/frameworks más demandados y con más soporte por parte de la comunidad del mercado actual:

- La conexión a la base de datos se realiza utilizando Laravel, este es uno de los dos frameworks punteros para el lenguaje servidor PHP. Laravel se basa en un despliegue de un árbol de directorios que emplea la filosofía MVC (Modelo Vista Controlador) que se instala fácilmente a través de Composer. Esta estructura y organización proporciona a los desarrolladores un sistema de rutas más clara y flexible, además, dispone de una seguridad sobresaliente frente a ataques.
- El código que trabaja en la parte del navegador está integrado en el árbol de directorios que proporciona Laravel, desarrollado con la librería/framework React+JSX. Su principal función es dar dinamismo a la aplicación de cara a los usuarios, manejando el DOM y las peticiones al servidor. React permite renderizar vistas sin tener que recargarlas y aporta una estructura de desarrollo modular gracias al uso de “componentes” y “estados o hooks”.

4. Descripción y Objetivos

El ERP va a estar construido sobre 7 módulos: inventario, compras, ventas, clientes, proveedores, empleados y tareas. La idea es simular un software en red destinado al personal laboral, va a estar desarrollado en un entorno local, ya que publicar una web conlleva costes que en el proyecto no se han asumido. Todos los datos de la empresa que se van a mostrar en la aplicación son ficticios y se van a almacenar en una base de datos relacional ubicada en el mismo equipo.

El objetivo del proyecto es organizar, automatizar, agilizar y economizar una farmacia mediante un software de planificación de recursos empresariales.

Los objetivos específicos del trabajo son:

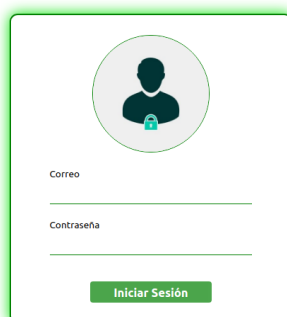
- Llevar a cabo un control de los empleados y sus respectivas tareas.
- Realizar un estudio financiero entre compras y ventas que produce la farmacia.
- Recoger datos de los proveedores y los productos que suministran.
- Documentar la información de los clientes y sus compras para llevar un control de demanda.

5. Contenidos

Este ERP a medida está orientado a una farmacia, compuesto por los módulos de inventario, ventas, compras, proveedores, clientes, empleados y tareas.

Para exponer el proyecto en el servidor web y visualizarlo seguir Anexo 1.

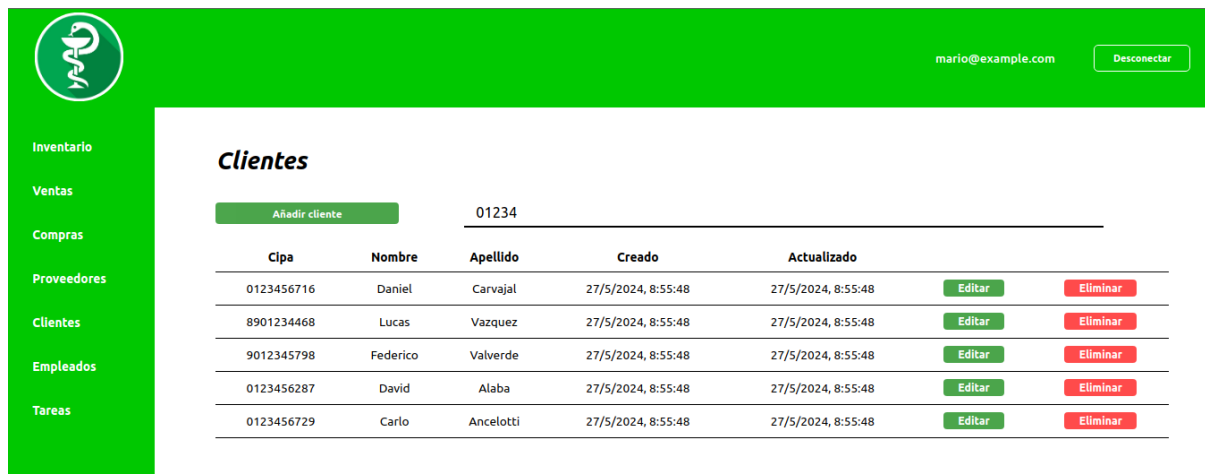
En primer lugar, como primer método de seguridad, los empleados deben iniciar sesión mediante usuario y contraseña para poder entrar a la aplicación, no podrán entrar directamente por rutas, si lo intentan sin haberse autenticado la aplicación les redirecciona directamente al login.



Una vez los empleados hayan iniciado sesión, accederán a la página principal. Aquí podrán visualizar un gráfico que muestra un balance entre las compras y las ventas que se han realizado en los últimos 6 meses. Al lado del gráfico, aparecerán las tareas pendientes que tiene el empleado conectado. De esta manera, nada más entrar al ERP, podrán tener datos financieros del negocio y un listado de las tareas pendientes para poder organizarse y cumplir los tiempos de cada tarea.



En cada módulo se podrán hacer diferentes acciones. Lo que van a tener en común es la visualización de registros, estos están distribuidos en filas, para visualizar un único registro o grupo de registros se podrá buscar en el campo abierto que tienen ubicados todos los módulos en la parte superior, este buscará coincidencias entre todas las columnas de cada registro.



| Cípa | Nombre | Apellido | Creado | Actualizado | | |
|------------|----------|-----------|--------------------|--------------------|--------|----------|
| 0123456716 | Daniel | Carvajal | 27/5/2024, 8:55:48 | 27/5/2024, 8:55:48 | Editar | Eliminar |
| 8901234468 | Lucas | Vazquez | 27/5/2024, 8:55:48 | 27/5/2024, 8:55:48 | Editar | Eliminar |
| 9012345798 | Federico | Valverde | 27/5/2024, 8:55:48 | 27/5/2024, 8:55:48 | Editar | Eliminar |
| 0123456287 | David | Alaba | 27/5/2024, 8:55:48 | 27/5/2024, 8:55:48 | Editar | Eliminar |
| 0123456729 | Carlo | Ancelotti | 27/5/2024, 8:55:48 | 27/5/2024, 8:55:48 | Editar | Eliminar |

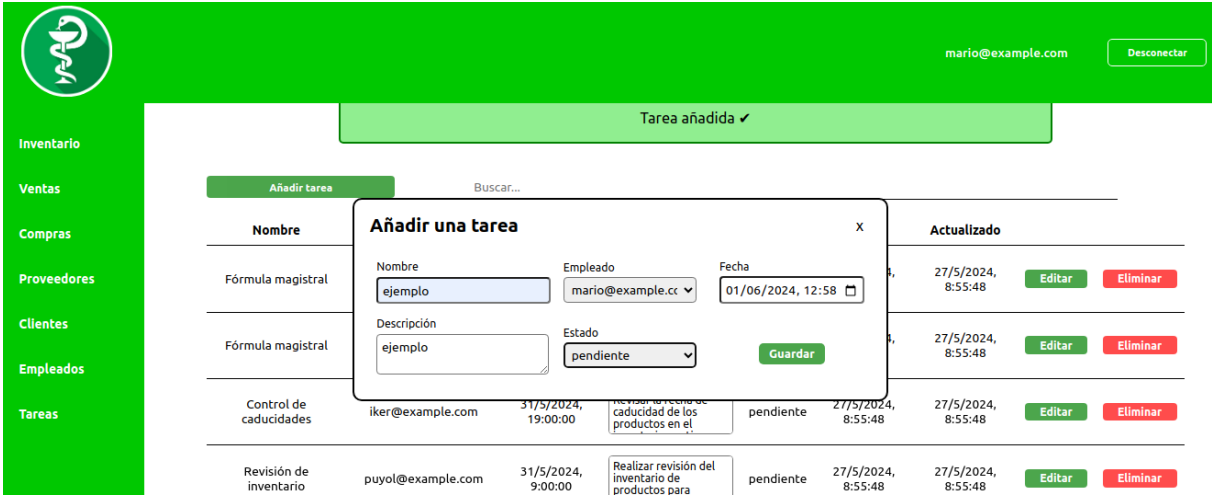
A su vez, cada módulo va a disponer, como ya se ha comentado, de acciones específicas. Algunos están habilitados para añadir, eliminar y editar (clientes, empleados, tareas), pero, sin embargo, otros no van a disponer de todas estas funciones.

El módulo de inventario no va a tener habilitada la opción de añadir un registro, ya que se van a agregar productos mediante las compras que se procesen en su módulo correspondiente. En los módulos de ventas, compras y proveedores, no se van a poder editar los registros, pero en el caso del módulo de ventas, se va a poder visualizar cada venta de manera más profunda pinchando en el botón “info”.

Cuando el empleado desee realizar alguna de las acciones nombradas, simplemente va a tener que pinchar en el botón que corresponda a la acción que desea del registro de interés.

En el caso de que se quiera utilizar la opción de añadir, para los módulos habilitados, este botón se va a encontrar a la izquierda del buscador. Los formularios de estos procesos van a aparecer en una ventana emergente o PopUp en el centro de la pantalla, desde ahí se podrá confirmar la solicitud o cerrar la ventana.

Se mostrará en pantalla encima del buscador si se ha procesado correctamente la solicitud o si ha ocurrido algún tipo de error.



Las acciones y procesos que se van a poder llevar a cabo en cada módulo, a modo de resumen, se muestran en la siguiente tabla:

| Módulo | Añadir | Editar | Eliminar | Detalles |
|-------------|--------|--------|----------|----------|
| Inventario | | | | |
| Ventas | | | | |
| Compras | | | | |
| Proveedores | | | | |
| Clientes | | | | |
| Empleados | | | | |
| Tareas | | | | |

6. Metodología y planificación

Para llevar a cabo este proyecto, hay que abordar en orden los siguientes apartados:

6.1. Inicio del proyecto

Para desarrollar un ERP primero hay que conocer los tipos de ERP (a medida y comercial) que hay en el mercado y documentarse sobre las empresas punteras en este tipo de software (Odoo, Holded, Oracle, SAP, etc.) y qué ofrecen para diferenciarse.

Una vez documentado, elegir el tipo de ERP que se quiera desarrollar, este dependerá del tipo de cliente y de las necesidades que desean cubrir. En este caso se ha buscado simplificarlo, ambientando el entorno del ERP a la empresa, pero las funciones van a ser básicas (CRUD).

Decidí orientar un ERP a medida a una oficina de farmacia por la utilidad que este puede suponer para la gestión y organización de estas.

6.2. Diseño de la aplicación

El ERP se diseñará conjuntamente con el cliente, tanto la estética de la interfaz mediante bocetos como la funcionalidad que se quiere cubrir. Para realizar estos bocetos y diagramas de flujo existen diversas herramientas, en mi caso utilicé opciones gratuitas.

Para el diseño estético de la aplicación utilicé Figma. Es una aplicación web que te permite hacer bocetos estéticos de manera ágil y sencilla, basta con registrarse como usuario y ya se puede empezar a diseñar.

A la hora de realizar el diagrama de flujo, usé Canva. Esta es otra aplicación web que ofrece muchas utilidades como pueden ser creación de gráficos y diagramas, además, Canva proporciona plantillas ya prediseñadas para facilitar el trabajo. Para poder utilizarla hay que seguir los mismos requisitos que con Figma, hay que registrarse.

6.3. Desarrollo de la aplicación

En este apartado ya entra en juego únicamente el desarrollador, con todos los datos previos, hay que buscar la manera más eficiente de desarrollar la aplicación. La idea del proyecto, al ser orientado hacia una empresa pequeña, se basa en soportar una aplicación web en un entorno local sin salir de la red hacia internet, proporcionando seguridad e independencia, ya que no se requiere de terceros para soportar un dominio ni se exponen los datos de la empresa. Lo ideal sería tener un equipo único donde estén alojados todos los datos de la empresa y conectarse a ese equipo a través de la misma red.

Teniendo claras las bases de la aplicación, se procede a elegir las tecnologías, esto ya se escoge al gusto del desarrollador, a no ser que se necesiten funciones específicas, pero no es el caso.

En mi caso, me estuve documentando sobre las tecnologías y lenguajes punteros del mercado actual y descubrí las oportunidades que ofrecen tanto Laravel como sistema que se encarga de manejar el código relacionado con el servidor, como React, que se utiliza para dar dinamismo a la web utilizando una estructura modular.

Escogí estas tecnologías principalmente por la demanda que tienen en el mercado y los beneficios que me suponía aprender a utilizar estos lenguajes, como también por la cantidad de soporte por parte de la comunidad que tienen y por su destacada curva de aprendizaje frente a sus competidores. En el caso de Laravel su competidor principal es Symfony, y por la parte de React, existen Angular y Vue como alternativas principales.

Cuando ya se tienen unos conocimientos sobre las tecnologías que se van a emplear, hay que preparar el entorno de desarrollo. Para este proyecto se necesita un contenedor de aplicaciones web o un servidor web y una estructura donde almacenar los datos de la aplicación, ya sea mediante archivos o mediante software. Yo he elegido Apache como servidor web y MariaDB como base de datos relacional.

El siguiente paso es montar el entorno de desarrollo. Gracias al sistema de gestión Composer, se puede crear un proyecto Laravel y su árbol de directorios con un solo comando.

Para integrar el lenguaje React, hace falta también instalar Node.js y su sistema de gestión de paquetes NPM. Si lo instalásemos por separado siguiendo el método tradicional, sería imposible poder integrar React dentro de Laravel, aquí es donde entra Inertia.

Inertia es el que adapta el entorno y hace de enrutador entre React y Laravel, Utilizando Composer, NPM y el propio comando de Laravel php artisan, podremos instalar todas las dependencias de Inertia dentro del espacio de trabajo de Laravel.

Inertia proporciona todo lo necesario para enrutar estos dos lenguajes y cómo utilizar sus propias palabras reservadas para programar dentro de su web oficial con ejemplos.

Después de tener ya todo configurado y preparado, toca desarrollar el proyecto. De normal se empieza desarrollando la lógica que se comunica con el servidor, para después, todas esas funciones adaptarlas para que presenten una estética atractiva para los usuarios. En cambio, al usar React en este proyecto, decidí que la mayoría de la lógica la soportaría el navegador, así que fui desarrollando la aplicación simultáneamente, dejando los estilos de CSS para el final del desarrollo.

Todos los ficheros del árbol de directorios de la aplicación se han desarrollado utilizando GitHub, este es un sistema de control de versiones, el cual proporciona a los desarrolladores un repositorio local y otro en la nube. Proyecto disponible en <https://github.com/MarioDSalamanca/Proyecto-DAW>

6.4. Fase final del proyecto

Una vez desarrollado el proyecto, se llevan a cabo unas pruebas estéticas, lógicas y de seguridad. Estas pruebas las va a realizar el desarrollador, conocedor de los posibles fallos que pueda provocar la aplicación, y el cliente o usuarios que no disponen de datos técnicos de la aplicación, destacando las virtudes y errores que podría contener el ERP, ya sean visuales o de ejecución.

Laravel destaca por su seguridad, ya que está preparado para evitar inyecciones SQL, además de separar la lógica con la base de datos a través de los modelos.

NPM (el gestor de dependencias de Node.js) permite comprimir y codificar todos los archivos correspondientes del frontend, guardando este código en otra carpeta distinta al directorio en el que se desarrollaba la aplicación, esto se hace cuando se saca a producción.

Por último (esto es opcional y depende de sistema operativo del equipo en el que se desarrolle la aplicación), creé un archivo el cual ejecuta una copia de seguridad de la base de datos. Este archivo se ejecuta periódicamente gracias al demonio cron de Ubuntu, con la finalidad de ofrecer respaldo a la base de datos en caso de que pudiesen acceder a ella o sufrir algún tipo de ataque.

6.5. Tiempo invertido en el proyecto

| Nombre | Duración | Fecha Inicio | Fecha Fin | Horas |
|-----------------------------|----------|--------------|------------|------------|
| Inicio del proyecto | 2 días | 13/02/2024 | 15/02/2024 | 5 |
| Diseño de la aplicación | 7 días | 16/02/2024 | 22/02/2024 | 14 |
| Desarrollo de la aplicación | 90 días | 23/02/2024 | 22/05/2024 | 360 |
| Fase final del proyecto | 4 días | 23/05/2024 | 26/05/2024 | 12 |
| | | | | 391 |

7. Desarrollo

Para exponer el proyecto en el servidor web y visualizarlo seguir Anexo 1.

Este proyecto va a trabajar sobre una base de datos relacional SQL de MariaDB, compuesto por 7 tablas, una para cada módulo, a excepción de la tabla detalle_ventas, que relaciona la tabla inventario con la tabla ventas al tener una relación de muchos a muchos. (Anexo 1)

7.1. Instalación y configuración del entorno

Para preparar el entorno, primero debemos tener instalado en nuestro equipo un servidor web (Apache), sus librerías para traducir lenguaje PHP y un sistema de base de datos (MariaDB).

```
sudo apt-get install php8.2 libapache2-mod-php php8.2-dev php8.2-zip php8.2-curl php8.2-  
mysql php8.2-gb php8.2-xml php8.2-mbstring mariadb-server curl unzip php -v  
sudo systemctl enable apache2  
sudo systemctl enable mariadb  
sudo mysql_secure_installation
```

A continuación, se muestran los comandos para instalar Composer en el equipo y configurarlo para poder utilizarlo de manera global.

```
curl -sS https://getcomposer.org/installer | php  
sudo mv composer.phar /usr/local/bin/composer  
sudo chmod +x /usr/local/bin/composer
```

Se va a desplegar un proyecto de Laravel v10 en la carpeta que proporciona apache para subir archivos al servidor. Para poder editar los archivos hay que darle permisos a la carpeta. Además, se creará un archivo para definir un host virtual.

```
sudo chown -R www-data:www-data /var/www/html/  
sudo usermod -a -G www-data mario  
sudo chmod 775 html  
sudo composer create-project laravel/laravel Proyecto-DAW  
sudo chmod -R 775 Proyecto-DAW/  
sudo chown -R www-data:www-data Proyecto-DAW/
```

sudo nano /etc/apache2/sites-available/laravel.conf

```
GNU nano 6.2 /etc/apache2/sites-enabled/laravel.conf
<VirtualHost *:80>
    DocumentRoot /var/www/html/Proyecto-DAW/Proyecto-DAW/

    <Directory /var/www/html/Proyecto-DAW/Proyecto-DAW/ >
        Options +FollowSymLinks
        AllowOverride All
        Require all granted
    </Directory>

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

sudo a2ensite laravel.conf

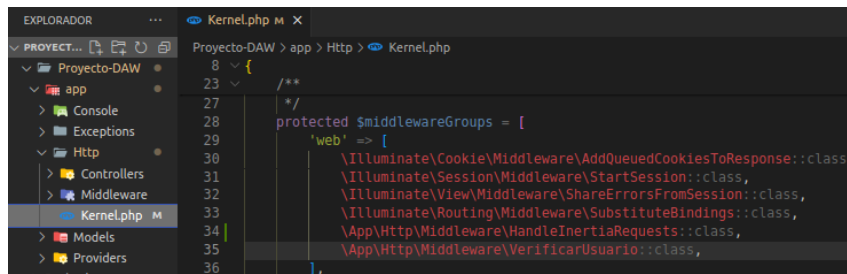
Para utilizar React e Inertia, hay que instalar Node.js v21 y NPM v10 a través del programa de instalación NVM.

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.7/install.sh | bash
nvm install 21
```

Una vez termine la instalación, desplegar las dependencias de Inertia + React en el proyecto de Laravel.

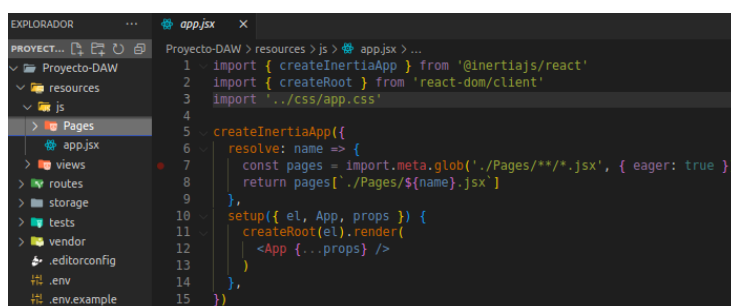
```
/Proyecto-DAW$ composer require inertiajs/inertia-laravel
```

```
/Proyecto-DAW$ php artisan inertia:middleware
```



```
/Proyecto-DAW$ npm install @inertiajs/react @inertiajs/inertia
```

```
/Proyecto-DAW$ npm install react react-dom
```



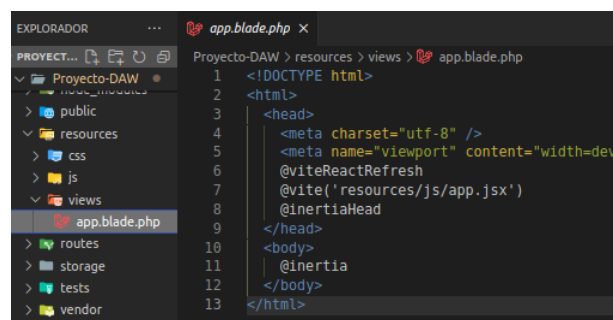
Para compilar el código del frontend, se usa Vite. Este soporta React + JSX y renderiza las vistas en caliente (al instante).

/Proyecto-DAW\$ npm install @vitejs/plugin-react



```
vite.config.js > default > plugins > input
1 import { defineConfig } from 'vite';
2 import laravel from 'laravel-vite-plugin';
3 import react from '@vitejs/plugin-react';
4
5 export default defineConfig({
6   plugins: [
7     laravel({
8       input: 'resources/js/app.jsx',
9       refresh: true,
10     }),
11     react(),
12   ],
13 });
```

Vamos a editar el “entry point” de la aplicación. Por defecto, las vistas de Laravel van a ser archivos .blade.php. En el index o app de las vistas de Laravel, forzamos una redirección al archivo app de la carpeta js, donde se van a crear las vistas con React.



```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8" />
5     <meta name="viewport" content="width=device-width, initial-scale=1">
6     @viteReactRefresh
7     @vite('resources/js/app.jsx')
8     @inertiaHead
9   </head>
10  <body>
11    @inertia
12  </body>
13 </html>
```


7.2. Modelos, Migraciones, Controladores, Seeders, Web.php, .env

Para desplegar el modelo, la migración y el controlador de un módulo, se utiliza el comando de Laravel “php artisan make:model *Módulo* -mrc”. A continuación, se muestran unos ejemplos de cómo se preparan estos archivos.

En el modelo se van a exponer los campos editables, el campo de clave primaria y las relaciones que tiene la tabla con otras (/app/Models/).

```

3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7 use App\Models\Empleados;
8 use App\Models\Cientes;
9 use App\Models\Detalle_ventas;
10
11 class Ventas extends Model
12 {
13     use HasFactory;
14
15     // Nombre de la clave primaria
16     protected $primaryKey = 'idVenta';
17
18     // Añadir los campos accesibles
19     protected $fillable = ['importe', 'fecha', 'idDetalleVenta', 'idCliente', 'idEmpleado'];
20
21     public function clientes() {
22         return $this->belongsTo(Cientes::class, 'idCliente');
23     }
24
25     public function empleados() {
26         return $this->belongsTo(Empleados::class, 'idEmpleado');
27     }
28
29     public function detalle_ventas() {
30         return $this->hasMany(Detalle_ventas::class, 'idVenta');
31     }
32 }
33

```

En el archivo de migración se define la estructura de la tabla, con los tipos de datos. Estos archivos se ejecutan usando el comando “php archisan migrate”, este comando despliega todas las migraciones que se encuentren en la carpeta de migraciones. Si no existe la base de datos, al desplegar las migraciones, se crea (/database/migrations).

```

7 return new class extends Migration {
8
9     public function up(): void {
10
11         Schema::create('ventas', function (Blueprint $table) {
12             $table->id('idVenta');
13             $table->decimal('importe');
14             $table->dateTime('fecha');
15             $table->unsignedBigInteger('idCliente')->nullable();
16             $table->foreign('idCliente')
17                 ->references('idCliente')
18                 ->on('clientes')
19                 ->onDelete('set null')
20                 ->onUpdate('cascade');
21             $table->unsignedBigInteger('idEmpleado')->nullable();
22             $table->foreign('idEmpleado')
23                 ->references('idEmpleado')
24                 ->on('empleados')
25                 ->onDelete('set null')
26                 ->onUpdate('cascade');
27             $table->timestamps();
28         });
29     }
30
31     public function down(): void
32     {
33         Schema::dropIfExists('ventas');
34     }
35 }
36

```

Los controladores van a procesar la lógica del backend a través de métodos, resolverán peticiones procedentes de las vistas y devolverán respuestas (/app/Http/Controllers/).

```

5 use App\Models\Cientes;
6 use App\Models\Ventas;
7 use App\Models\Detalle_ventas;
8 use App\Models\Empleados;
9 use App\Models\Inventario;
10 use Illuminate\Http\Request;
11 use Illuminate\Support\Str;
12 use Inertia\Inertia;
13
14 class VentasController extends Controller {
15
16     // Index del módulo de ventas
17 > public function index(){~
18 }
19
20     // Añadir un registro a la tabla
21 > public function insert(Request $request) {~
22 }
23
24     // Eliminar un registro de la tabla
25 > public function delete(Request $request) {~
26 }
27 }

```

Las rutas de la aplicación y las configuraciones de los middlewares se van a definir en el archivo web.php, este archivo comunica el frontend con el backend.

```

3 use App\Http\Controllers\CientesController;
4 use App\Http\Controllers\ComprasController;
5 use App\Http\Controllers>LoginController;
6 use App\Http\Controllers\EmpleadosController;
7 use App\Http\Controllers\HomeController;
8 use App\Http\Controllers\InventarioController;
9 use App\Http\Controllers\ProveedoresController;
10 use App\Http\Controllers\TareasController;
11 use App\Http\Controllers\VentasController;
12 use Illuminate\Support\Facades\Route;
13
14 // Login fuera del middleware para poder acceder e iniciar sesión
15 Route::get('/login', [LoginController::class, 'showLogin'])->name('login');
16 Route::post('/login', [LoginController::class, 'auth'])->name('login');
17
18 // Comprobar el inicio de sesión para poder acceder a las rutas de la app
19 Route::middleware(['verificar_usuario'])->group(function () {
20
21     // Rutas para llamar a los métodos de los controladores
22     // Home
23     Route::get('/', [HomeController::class, 'index'])->name('home');
24     // Cerrar sesión y salir del middleware
25     Route::post('/logout', [HomeController::class, 'logout'])->name('logout');
26
27     // Clientes
28     Route::get('/clientes', [ClientesController::class, 'index'])->name('clientes.index');
29     Route::post('/clientes/añadir', [ClientesController::class, 'insert'])->name('clientes.insert');
30     Route::post('/clientes/eliminar', [ClientesController::class, 'delete'])->name('clientes.delete');

```

Para rellenar la base de datos con registros de prueba (para hacer pruebas los desarrolladores), se ejecuta el archivo DatabaseSeeder.php de la carpeta seeders. Estos registros se insertan utilizando el comando “php artisan db:seed”.

```

15 class DatabaseSeeder extends Seeder
16 {
17     public function run() {
18
19         $proveedores = [
20             ['empresa' => 'Bidaforma'],
21             ['empresa' => 'Cofares'],
22             ['empresa' => 'Bayer'],
23         ];
24
25         // Crear registros de proveedores
26         foreach ($proveedores as $proveedor) {
27             Proveedores::create($proveedor);
28         }
29     }
30 }

```

Las configuraciones de conexión a la base de datos, el estado del proyecto (desarrollo o producción), configuraciones de PHPMailer y comunicaciones con Vite, se van a especificar en el archivo .env.

7.3. Flujo de ejecución

Como primer método de seguridad del ERP, tenemos un middleware para controlar el acceso a las rutas. Se definen en el archivo web.php, dentro de la función del middleware, todas las rutas que van a estar condicionadas por el middleware. La clase del middleware hay que referenciarla en el fichero Kernel.php.

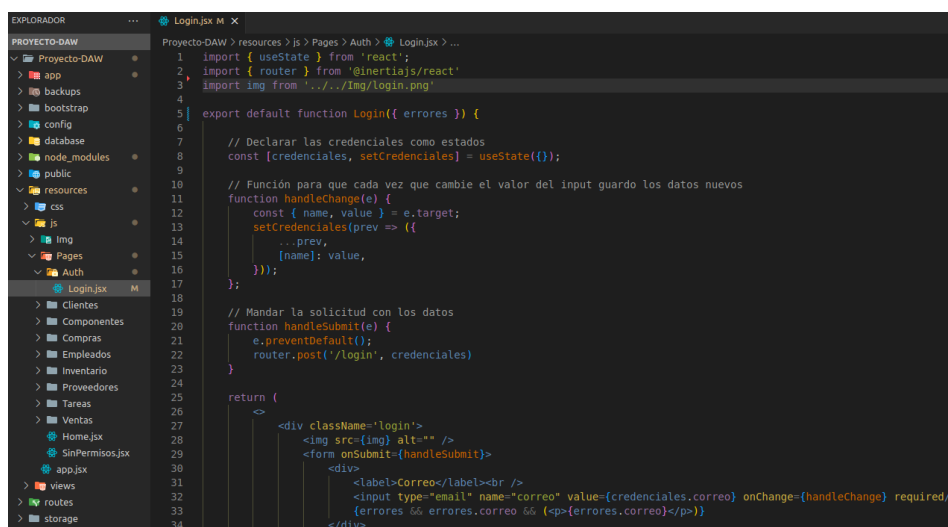
```
// Login fuera del middleware para poder acceder e iniciar sesión
Route::get('/login', [LoginController::class, 'showLogin'])->name('login');
Route::post('/login', [LoginController::class, 'auth'])->name('login');

// Comprobar el inicio de sesión para poder acceder a las rutas de la app
Route::middleware(['verificar_usuario'])->group(function () {

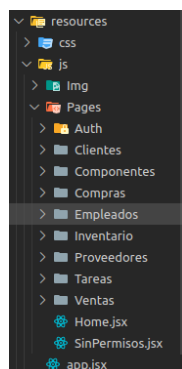
    // Rutas para llamar a los métodos de los controladores
```

Dentro del middleware se va a verificar si la ruta de la petición es la del login, si no lo es, verifica si existe la variable de sesión “usuario_autenticado”, si no existe redirige a la vista del login. Si existe esa variable de sesión, con return \$next(\$request), da paso a las rutas que tiene dentro de la función en web.php. (Anexo 2)

En la vista se crea un objeto a través del hook useState() para guardar las credenciales del usuario y mandar la solicitud mediante router.post(url,estado). Cada estado tiene su setter para definir sus valores, este se va a usar cuando se llame a la función handleChange con el evento onChange.



Dentro del controlador del login, verificamos las credenciales que llegan desde el frontend. Las contraseñas de los empleados están codificadas con bcrypt en la base de datos, este es un método de cifrado que ofrece Laravel. Una vez codificado no se puede revertir para ver su cadena original, pero se puede comparar si coincide la contraseña de la solicitud con la contraseña cifrada del usuario. (Anexo 3)



El redireccionamiento, inserción, modificación y eliminación de datos siempre se manejan desde el backend, pero la mayor parte del flujo se procesa en el frontend. En desarrollo, todas las vistas se van a encontrar en la carpeta `/resources/js/`, ese directorio se va a dividir en otros para organizar y modular las vistas.

Cada módulo tiene disponible una carpeta en la que se encuentran los PopUps que necesite. Todos excepto el PopupAñadir de ventas, van a procesar los datos de los formularios en el archivo `FuncionesPopUps` que se encuentra en el directorio `Componentes`. Este archivo va a tener muchas funcionalidades: mostrar y esconder los PopUps, guardar los datos del formulario en un estado llamado `formDatos` y realizar las solicitudes al servidor (Anexo 4). A parte de este componente genérico, vamos a trabajar con dos más:

- Header: este componente está pensado para ahorrar código HTML, contiene la parte de navegación y la cabecera de la vista. Solamente contiene enlaces, no tiene lógica. (Anexo 5)
- Buscador: sirve para filtrar los resultados de los registros que llegan de la base de datos de cada módulo. (Anexo 6)

En las vistas de los módulos hay que importar: las funciones de los PopUps, cada componente que simule un PopUp, los componentes del directorio `Componentes` y recibir como parámetros las variables que se envían desde el servidor. (Anexo 7)

La mayoría de las funciones de `FuncionesPopUps` se pasan como parámetros a los componentes de los PopUps para poder llamarlas. (Anexo 8)

Para mostrar los registros de cada módulo se mapea el objeto `datosFiltrados`, el cual proviene del objeto `datosServidor` desde el servidor. `datosFiltrados` va a ser la cadena de caracteres que coincida con algún dato de una columna de cualquier registro, este código se procesa en el componente `Buscador`. Si no se ha llamado al evento `onChange` del `Buscador` o el valor del campo que llama al evento está vacío, `datosFiltrados` será una copia de `datosServidor`. Este se pasa como parámetro a la vista para cargar los registros al renderizar. (Anexo 9)

Cabe resaltar que, para añadir un registro en el módulo de ventas, se trabaja la lógica en un archivo aparte, ya que, al trabajar con varias tablas, hay que validar correctamente y en profundidad el formulario. La dinámica de la lógica es la misma, pero no la incluí en el fichero general `FuncionesPopUps` porque no se usa en ningún otro módulo y son demasiadas líneas de código, preferí separarlo. (Anexo 10)

Cuando se termina de desarrollar el proyecto y se han llevado a cabo las pruebas pertinentes, el ERP se pasa a producción. Esto consiste en codificar el código, comprimirlo y traspasarlo a las carpetas destinadas para ello. Cuando se procesa el código del frontend a producción, los archivos del directorio `/resources/js/` se transforman en la carpeta `/public/build/assets/`.

```
mario@marioUbuntu:/var/www/html/Proyecto-DAW/Proyecto-DAW$ npm run build
> build
> vite build

vite v5.2.7 building for production...
✓ 171 modules transformed.
public/build/assets/compras-pXSbkEg3.jpg 1.70 kB | gzip: 0.41 kB
public/build/assets/proveedores-Cry00hn7.jpg 7.27 kB
public/build/assets/inventario-BKLTjXtC.jpg 9.76 kB
public/build/assets/empleados-Ctq52Wsq.jpg 12.74 kB
public/build/assets/tareas-CtTKb_XH.jpg 12.94 kB
public/build/assets/clientes-zY1DDUJJ.jpg 20.86 kB
public/build/assets/login-Bn14pvpE.png 33.02 kB
public/build/assets/ventas-CY20LMB-.jpg 35.76 kB
public/build/assets/logo-Fvt052q1.png 62.28 kB
public/build/assets/app-B9JW0IF.js 121.25 kB
public/build/assets/app-B9JW0IF.js 7.40 kB | gzip: 2.08 kB
304.15 kB | gzip: 92.83 kB
✓ built in 1.65s
```

```
mario@marioUbuntu:/var/www/html/Proyecto-DAW/Proyecto-DAW$ composer install --optimize-autoloader --no-dev
Installing dependencies from lock file
Verifying lock file contents can be installed on current platform.
Nothing to install, update or remove
Generating optimized autoload files
> illuminateFoundationComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi

 INFO  Discovering packages.

inertiajs/inertia-laravel ..... DONE
laravel/sanctum ..... DONE
laravel/tinker ..... DONE
nesbot/carbon ..... DONE
nunomaduro/termwind ..... DONE

34 packages you are using are looking for funding.
Use the 'composer fund' command to find out more!
```

Finalizado el proyecto y puesto en producción, programé un archivo ejecutable escrito en bash para Ubuntu, con el fin de programar copias periódicas de la base de datos para guardarlas en un directorio llamado `backups`. (Anexo 11)

8. Conclusiones y proyección a futuro

Realizar este ERP, para mí, ha sido un reto personal. Yo tenía en mente como quería que fuese mi proyecto, pero, no tenía conocimientos previos sobre las tecnologías que trabajan el frontend y el backend. Tengo conocimientos sobre JavaScript y PHP, y sabía que existían frameworks y lo que suponían para los desarrolladores y para las empresas que los empleados conozcan estas tecnologías, pero nunca había trabajado con ningún framework relacionado.

Una alternativa atractiva para mí de realizar este proyecto es desarrollando una aplicación de escritorio con las mismas características, pero con otras tecnologías.

Mi ERP se podría ampliar, por ejemplo, de la siguiente manera:

- Instalar un servidor FTP para compartir las copias de seguridad de la base de datos en otro equipo, esto proporciona una capa adicional de respaldo.
- Separar la base de datos en otro equipo y conectarse a ella por red (local). Esto no se llevó a cabo en mi proyecto por falta de recursos de hardware en mi equipo personal.
- Instalar y configurar un servidor DNS para facilitar a los usuarios el acceso a la aplicación, es más fácil recordar `erp-farmacia` que `localhost:8000`.

9. Bibliografía

Oracle. (n.d.). ¿Qué es ERP?. <https://www.oracle.com/es/erp/what-is-erp/>

SAP. (n.d.). Qué es ERP?. <https://www.sap.com/spain/products/erp/what-is-erp.html>

Composer. (n.d.). <https://getcomposer.org/doc/>

Laravel. (n.d.). Laravel Versión 10.x. <https://laravel.com/docs/10.x/readme>

Laravel. (n.d.). Laravel Versión 9.x. DocumentacionLaravel.com.
<https://documentacionlaravel.com/docs/9.x>

Inertia.js. (n.d.). <https://inertiajs.com/>

React. (n.d.). Aprende React Versión 18.3.1. <https://es.react.dev/learn>

React. (n.d.). Referencias Versión 18.3.1. <https://es.react.dev/reference/react>

Vite. (n.d.). Why Vite Versión 4. <https://v4.vitejs.dev/guide/why.html>

10. Anexos

Anexo 1

Exponer el proyecto en el servidor web (localhost:8000)

Comando para compilar Laravel:

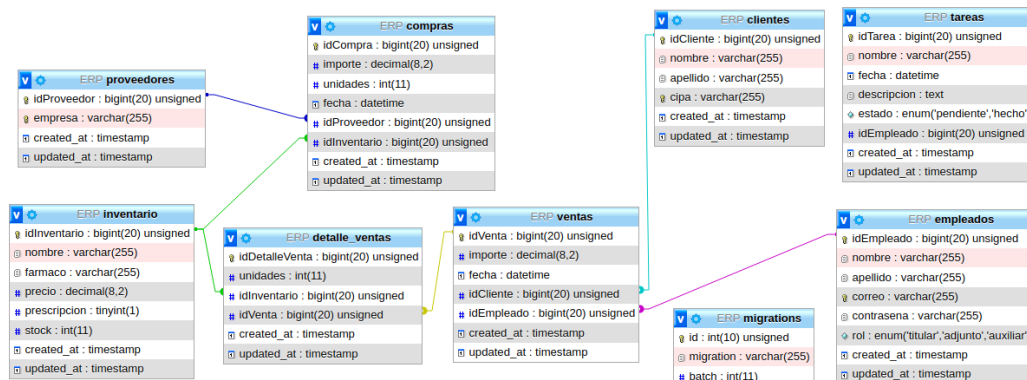
var/www/html/Proyecto-DAW/\$ **php artisan serve**

Comando para compilar React (en producción no es necesario):

var/www/html/Proyecto-DAW/\$ **npm run dev**

Anexo 2

Esquema de BBDD



Anexo 3

VerificarUsuario.php (Middleware)

```
9 class VerificarUsuario {
10
11     public function handle(Request $request, Closure $next): Response {
12
13         // Excluir la ruta de inicio de sesión de la verificación de autenticación
14         $route = $request->route();
15         if ($route && $route->getName() !== 'login') {
16             // Comprobar que existe la variable de sesión
17             if (!$request->session()->has('usuario_autenticado')) {
18                 return redirect()->route('login');
19             }
20         }
21
22         // $next($request) para continuar el flujo
23         return $next($request);
24     }
25 }
```


Anexo 4

LoginController.php

```
10 class LoginController extends Controller {
11
12     // Index del login
13     public function showLogin() {
14         return Inertia::render('Auth/Login');
15     }
16
17     // Autenticación
18     public function auth(Request $request) {
19
20         // Buscar al usuario por su dirección de correo electrónico
21         $usuario = Empleados::where('correo', $request->correo)->first();
22
23         if ($usuario) {
24
25             // Comprobar si existe el usuario y si la contraseña es correcta
26             if (Hash::check($request->contrasena, $usuario->contrasena)) {
27
28                 // Iniciar variable de sesión con los valores del usuario
29                 $request->session()->put('usuario_autenticado', $request->correo);
30
31                 return redirect()->route('home');
32             } else {
33                 // Contraseña incorrecta
34                 return Inertia::render('Auth/Login', ['errores' => ['contrasena' => 'La contraseña proporcionada es incorrecta.']] );
35             }
36         } else {
37             // Usuario incorrecto
38             return Inertia::render('Auth/Login', ['errores' => ['correo' => 'No se encontró ningún usuario con este correo electrónico.']] );
39         }
40     }
41 }
42
43
44 }
```

Anexo 5

FuncionesPopUps.jsx

```
1 import { useState } from 'react';
2 import { router } from '@inertiajs/react';
3
4 export default function FuncionesPopUps() {
5
6     // Estados
7     const [popupAñadir, setPopupAñadir] = useState(false);
8     const [popupEditar, setPopupEditar] = useState(false);
9     const [popupEliminar, setPopupEliminar] = useState(false);
10    const [popupInfo, setPopupInfo] = useState(false);
11    const [datoEliminar, setDatoEliminar] = useState('');
12    const [formDatos, setFormDatos] = useState({});
13
14    // Llamadas para iniciar las acciones
15    function añadir() {
16        mostrarPopupAñadir();
17        setFormDatos({});
18    }
19
20    function editar(registro) {
21        mostrarPopupEditar();
22        setFormDatos(registro);
23    };
24
25    function eliminar(dato) {
26        mostrarPopupEliminar();
27        setDatoEliminar(dato);
28    };
29
30    function info(registro) {
31        mostrarPopupInfo();
32        setFormDatos(registro);
33    };
34
35    // Funciones para mostrar los PopUps
36    function mostrarPopupAñadir() { setPopupAñadir(!popupAñadir) };
37    function mostrarPopupEditar() { setPopupEditar(!popupEditar) };
38    function mostrarPopupEliminar() { setPopupEliminar(!popupEliminar) };
39    function mostrarPopupInfo() { setPopupInfo(!popupInfo) };
40
41    // Setear los valores de los formularios de Añadir y Editar
42    function handleChange(e) {
43        const { name, value } = e.target;
44        setFormDatos(prev => ({
45            ...prev,
46            [name]: value
47        }));
48    };
49
50    // Solicitudes POST al servidor
51    function confirmarAñadir(e, url) {
52        e.preventDefault();
53        mostrarPopupAñadir();
54        router.post(url, formDatos);
55        setFormDatos({});
56        window.location.reload();
57    };
58
59    function confirmarAñadirVenta(e, url, venta) {
60        e.preventDefault();
61        mostrarPopupAñadir();
62        router.post(url, venta);
63        window.location.reload();
64    };
65
66    function confirmarEditar(e, url) {
67        e.preventDefault();
68        mostrarPopupEditar();
69        router.post(url, formDatos);
70        setFormDatos({});
71        window.location.reload();
72    };
73
74    function confirmarEliminar(e, url) {
75        e.preventDefault();
76        mostrarPopupEliminar();
77        router.post(url, { dato: datoEliminar });
78        window.location.reload();
79    };
80 }
```

Anexo 6

Header.jsx

```
1 import { router } from '@inertiajs/react';
2 import logo from '../Img/logo.png';
3
4 export default function Header({ session }) {
5
6   // Función para cerrar sesión
7   const logout = () => {
8     router.post('/logout');
9   };
10
11   return (
12     <header>
13       <div className="divLogo">
14         <a href="/"><img src={ logo } alt="" /></a>
15       </div>
16       <div className="divSesion">
17         { session }
18       </div>
19       <div className="divLogout">
20         <button onClick={ logout }>Desconectar</button>
21       </div>
22     </header>
23     <nav>
24       <ul>
25         <a href="/inventario"><li>Inventario</li></a>
26         <a href="/ventas"><li>Ventas</li></a>
27         <a href="/compras"><li>Compras</li></a>
28         <a href="/proveedores"><li>Proveedores</li></a>
29         <a href="/clientes"><li>Clientes</li></a>
30         <a href="/empleados"><li>Empleados</li></a>
31         <a href="/tarefas"><li>Tareas</li></a>
32       </ul>
33     </nav>
34   )
35 }
36
37 }
```

Anexo 7

Buscador.jsx

```
1 import { useState } from 'react';
2
3 export default function Buscador({ datosServidor, setDatosFiltrados, campos }) {
4
5   // Estado del buscador
6   const [buscar, setBuscar] = useState('');
7
8   // Actualizar el valor del buscador
9   const handleChangeBuscador = (e) => {
10
11     const valor = e.target.value.trim();
12     setBuscar(valor);
13
14     if (datosServidor && campos) {
15       // Filtrar los resultados
16       const datosFiltrados = datosServidor.filter(datos => {
17         campos.some(campo => {
18           // Para comprobar si se le pasan objetos anidados (en caso de acceder a un campo de otra tabla) y lo descompone para poder filtrarlo
19           const valorCampo = campo.split('.').reduce((obj, key) => (obj && obj[key] !== undefined) ? obj[key] : undefined, datos);
20           return valorCampo && valorCampo.toString().toLowerCase().includes(valor.toLowerCase());
21         });
22       });
23
24       // Actualizar el estado de los datosServidor filtrados
25       setDatosFiltrados(datosFiltrados);
26     }
27   };
28
29   return (
30     <input type="text" name="buscador" className="buscador" placeholder="Buscar..." value={ buscar } onChange={ handleChangeBuscador } />
31   );
32 }
```

Anexo 8

Importaciones en las vistas

```
1 import Header from "../Componentes/Header";
2 import FuncionesPopUps from "../Componentes/FuncionesPopUps";
3 import PopupAñadir from "../Popups/PopupAñadir";
4 import PopupEditar from "../Popups/PopupEditar";
5 import PopupEliminar from "../Popups/PopupEliminar";
6 import Buscador from "../Componentes/Buscador";
7 import { useState } from "react";
8
9 export default function Tareas({ sesionUsuario, datosServidor, empleados, mensaje }) {
10
11     const [datosFiltrados, setDatosFiltrados] = useState(datosServidor);
12
13     // Usa las funciones de popup
14     const {
15         popupAñadir,
16         popupEditar,
17         popupEliminar,
18         mostrarPopupAñadir,
19         mostrarPopupEditar,
20         mostrarPopupEliminar,
21         añadir,
22         editar,
23         eliminar,
24         handleChange,
25         confirmarAñadir,
26         confirmarEditar,
27         confirmarEliminar,
28         formDatos
29     } = FuncionesPopUps();
```

Anexo 9

Ejemplo de uso de componentes

```
9 export default function Tareas({ sesionUsuario, datosServidor, empleados, mensaje }) {
31     return (
32         <>
33         <Header sesion={ sesionUsuario }/>
34         <main>
35             <h1>Tareas</h1>
36             { popupAñadir && <PopupAñadir mostrarPopupAñadir={ mostrarPopupAñadir } confirmarAñadir={ confirmarAñadir }
37             formDatos={ formDatos } handleChange={ handleChange } empleados={ empleados } /> }
38             { popupEditar && <PopupEditar mostrarPopupEditar={ mostrarPopupEditar } confirmarEditar={ confirmarEditar }
39             formDatos={ formDatos } handleChange={ handleChange } empleados={ empleados } /> }
40             { popupEliminar && <PopupEliminar mostrarPopupEliminar={ mostrarPopupEliminar }
41             confirmarEliminar={ confirmarEliminar } /> }
42             {mensaje && (
43                 <div>
44                     {mensaje.exito && <p className="mensaje exito">{mensaje.exito} &#x2714;</p>}
45                     {mensaje.error && <p className="mensaje error">&#x274C; {mensaje.error}</p>}
46                 </div>
47             )}
48             { datosServidor &&
49                 <>
50                     <div className="cabecera-tabla">
51                         <div>
52                             <button className="añadir" onClick={ añadir }>Añadir tarea</button>
53                         </div>
54                         <div className="div-buscador">
55                             <Buscador datosServidor={ datosServidor } setDatosFiltrados={ setDatosFiltrados }
56                             campos={['nombre', 'empleados.correo', 'fecha', 'descripcion', 'estado' ]} />
57                         </div>
58                     </div>
59                 </>
60             )}
61         </main>
62         </>
63     );
64 }
```

Anexo 10

Mostrar registros de las tablas

```

59 <table className="tablaDatos">
60 <tbody>
61 <tr>
62 <th>Nombre</th>
63 <th>Empleado</th>
64 <th>Fecha</th>
65 <th>Descripción</th>
66 <th>Estado</th>
67 <th>Creado</th>
68 <th>Actualizado</th>
69 <th></th>
70 <th></th>
71 </tr>
72 { datosFiltrados.length === 0 ? (
73 <tr>
74 <td colspan="9"><p className="sin-resultados">No se encontraron resultados</p></td>
75 </tr>
76 ) : ( datosFiltrados.map(tarea => (
77 <tr key={tarea.idTarea}>
78 <td>{tarea.nombre}</td>
79 <td>{tarea.empleados.correo}</td>
80 <td>{new Date(tarea.fecha).toLocaleString()}</td>
81 <td><textarea value={tarea.descripcion} cols="30" rows="3" readOnly /></td>
82 <td>{tarea.estado}</td>
83 <td>{new Date(tarea.created_at).toLocaleString()}</td>
84 <td>{new Date(tarea.updated_at).toLocaleString()}</td>
85 <td className="botones editar"><button onClick={() => editar(tarea)} >Editar</button></td>
86 <td className="botones eliminar"><button onClick={() => eliminar(tarea.idTarea)}>Eliminar</button></td>
87 </tr>
88 ))
89 </tbody>
90 </table>
91 </>
92 }
93

```

Anexo 11

FuncionesPopupAñadir.jsx (Módulo Ventas)

```

1 import { useEffect, useState } from "react";
2
3 export default function FuncionesPopupAñadir() {
4
5   // Estados
6   const [venta, setVenta] = useState({});
7   const [errores, setErrores] = useState({});
8
9   // Cada vez que cambie errores se actualiza en el párrafo
10  useEffect(() => {
11
12    // Limpia el contenido existente en "avisos"
13    let avisos = document.getElementById("avisos");
14    avisos.innerHTML = "";
15
16    // Agrega cada error como un párrafo
17    Object.keys(errores).forEach((errorName) => {
18      let parrafo = document.createElement("p");
19      parrafo.textContent = errores[errorName];
20      avisos.appendChild(parrafo);
21    });
22
23  }, [errores]);
24
25  // Añadir los errores
26  function err(name, value) {
27    setErrores(prev => ({
28      ...prev,
29      [name]: value
30    }));
31  }
32
33  // Eliminar errores
34  function eliminarErr(name) {
35
36    if (errores[name]) {
37      const nuevo = { ...errores };
38      delete nuevo[name];
39      setErrores(nuevo);
40    }
41  }

```

```

45 // Añadir datos a venta
46 function set(name, value) {
47   setVenta(prev => ({
48     ...prev,
49     [name]: value
50   }));
51 }
52
53 // Eliminar dato de la venta
54 function borrar(name) {
55   let nuevo = { ...venta };
56   delete nuevo[name];
57   setVenta(nuevo);
58 }
59
60 // Procesar los campos de cipa, empleado, fecha, nombre y apellido pasando la lista de clientes
61 function handleChangeDatos(e, clientes) {
62
63   const cipa = document.getElementsByName('cipa')[0];
64
65   const { name, value } = e.target;
66
67   // Comprobar si se ha rellenado el cipa y se le llama a la funcion pasando clientes
68   if (clientes && cipa.value.length == 10) {
69
70     // Si el cipa está en la lista de clientes
71     if (clientes.includes(value)) {
72       document.getElementsByClassName('cliente')[0].classList.add('oculto');
73       document.getElementsByClassName('cliente')[1].classList.add('oculto');
74
75       set(name, value);
76
77       eliminarErr("errorCipa")
78     }

```

Desarrollo de un ERP

```
81 function handleChangeDatos(e, clientes) {
82   // Mostrar el error y los campos
83   document.getElementsByClassName('cliente')[0].classList.remove('oculto');
84   document.getElementsByClassName('cliente')[1].classList.remove('oculto');
85   err("ErrorCipa", "El cliente con el cipa ('+cipa.value+') no está registrado");
86   set(name, value);
87
88   const nombre = document.getElementsByName('nombre')[0];
89   const apellido = document.getElementsByName('apellido')[0];
90
91   // Si se rellenan nombre y apellido añadirlo
92   if (nombre.value.length >= 2 || apellido.value.length >= 3) {
93     set(name, value);
94   }
95 } else {
96   set(name, value);
97 }
98 }
99
100 // Función para procesar los selects
101 function handleChangeProductos(e, productos) {
102   const { name, value } = e.target;
103
104   // Buscar el producto seleccionado en el select
105   const productoObj = productos.find(producto => producto.nombre === value);
106
107   const farmaco = document.getElementsByName(name)[0];
108 }
```

```
108 function handleChangeProductos(e, productos) {
109   function añadir() {
110     // Capturar el campo del cipa
111     const cipa = document.getElementsByName('cipa')[0];
112
113     // Verificar si necesita prescripción médica
114     if (productoObj.prescripcion === 1) {
115       if (cipa.value.length >= 10) {
116         farmaco.value = '';
117         err("errorPrescripcion", "El fármaco " + productoObj.nombre + " necesita prescripción médica");
118         cipa.focus();
119
120         if (venta[name]) {
121           borrar(name);
122         }
123
124         return;
125       } else {
126         // Eliminar el producto anterior
127         if (venta[name]) {
128           borrar(name);
129           const unidades = 'unidades-' + name.split('-')[1];
130           document.getElementsByName(unidades)[0].value = '';
131         }
132
133         // Agregar el producto seleccionado a la venta
134         setVenta(prev => ({
135           ...prev,
136           [name]: {
137             producto: productoObj,
138             unidades: 0
139           }
140         }));
141
142         eliminarErr("errorPrescripcion");
143       }
144     }
145   }
146 }
```

```
100 function handleChangeProductos(e, productos) {
101   // Si la venta no está vacía
102   if (Object.keys(venta).length !== 0) {
103     // Verificar si el producto ya está en la venta
104     let productoExistente = false;
105
106     for (const i in venta) {
107       if (i.startsWith('productos-')) {
108         const nombreProducto = venta[i].producto.nombre;
109
110         if (nombreProducto === value) {
111           productoExistente = true;
112           break;
113         }
114       }
115     }
116
117     if (productoExistente) {
118       // Vaciar el select y las unidades
119       farmaco.value = '';
120       const unidades = 'unidades-' + name.split('-')[1];
121       document.getElementsByName(unidades)[0].value = '';
122       err("errorRepetido", "El producto "+value+" ya está en la venta");
123
124       if (venta[name]) {
125         borrar(name);
126       }
127       return;
128     } else {
129       añadir();
130       eliminarErr("errorRepetido");
131     }
132   } else {
133     añadir();
134   }
135 }
```

```
108   } else {
109     // Eliminar el producto anterior
110     if (venta[name]) {
111       borrar(name);
112       const unidades = 'unidades-' + name.split('-')[1];
113       document.getElementsByName(unidades)[0].value = '';
114     }
115
116     // Agregar el producto seleccionado a la venta
117     setVenta(prev => ({
118       ...prev,
119       [name]: {
120         producto: productoObj,
121         unidades: 0
122       }
123     }));
124   }
125
126   // Ocultar o mostrar el campo de las unidades si hay un producto seleccionando
127   const unidades = document.getElementsByName('unidades-' + name.split('-')[1])[0];
128
129   if (farmaco.value !== '') {
130     unidades.style.display = 'block';
131   } else {
132     unidades.style.display = 'none';
133   }
134 }
```

```

218 function handleChangeUnidades(e) {
219     const { name, value } = e.target;
220     const n = name.split('-')[1];
221
222     setVenta(prev => ({
223         ...prev,
224         ['productos-' + n]: {
225             ...prev['productos-' + n],
226             unidades: parseInt(value)
227         }
228     }));
229 }
230
231 // Agregar los campos de select para añadir más productos
232 function agregarSelect(productos) {
233
234     let ultimoSelect = null;
235
236     for (let i = 10; i >= 0; i--) {
237         const selects = document.getElementsByName('productos-' + i);
238
239         if (selects.length > 0) {
240             ultimoSelect = selects[0];
241             break;
242         }
243     };
244
245     // Sacar el último select que hay en el html por el nº
246     ultimoSelect = ultimoSelect.name;
247     const ultimoCaracter = ultimoSelect.slice(-1);
248     const ultimoNumero = parseInt(ultimoCaracter, 10);
249     const nuevoSelect = ultimoNumero + 1;
250
251     const contenedor = document.createElement('p');
252     contenedor.id = nuevoSelect;
253

```

```

233 function agregarSelect(productos) {
234
235     // Crear el botón de eliminar
236     const botonEliminar = document.createElement('button');
237     botonEliminar.id = 'eliminar' + nuevoSelect;
238     botonEliminar.textContent = '-';
239     botonEliminar.onclick = function() {
240         eliminarSelect(contenedor.id);
241     };
242
243     // Agregar los botones al contenedor
244     contenedor.appendChild(botonEliminar);
245
246     // Agregar el select al contenedor
247     contenedor.appendChild(select);
248
249     // Agregar unidades al contenedor
250     contenedor.appendChild(unidades);
251
252     // Agregar el contenedor al contenedor-selects
253     document.getElementById('contenedor-selects').appendChild(contenedor);
254 }

```

```

233 function agregarSelect(productos) {
234     // Crear el select
235     const select = document.createElement('select');
236     select.id = 'productos' + nuevoSelect;
237     select.name = 'productos-' + nuevoSelect;
238     select.value = venta.productos;
239     select.onChange = (e) => {
240         handleChangeProductos(e, productos);
241     };
242     select.required = true;
243
244     // Agregar opciones al select
245     const opcionVacia = document.createElement('option');
246     opcionVacia.value = '';
247     opcionVacia.textContent = '';
248     select.appendChild(opcionVacia);
249
250     // Agregar los options con los productos al select
251     productos.forEach(producto => {
252         const option = document.createElement('option');
253         option.key = producto.idInventario;
254         option.value = producto.nombre;
255         option.textContent = producto.nombre;
256         select.appendChild(option);
257     });
258
259     // Crear las unidades para el select
260     const unidades = document.createElement('input');
261     unidades.type = 'number';
262     unidades.id = 'unidades' + nuevoSelect;
263     unidades.name = 'unidades-' + nuevoSelect;
264     unidades.value = '';
265     unidades.onChange = function(e) {
266         handleChangeUnidades(e);
267     };
268     unidades.min = 1;
269     unidades.required = true;
270

```

```

313 // Para eliminar el select seleccionado con sus unidades
314 function eliminarSelect(id) {
315
316     const select = 'productos-' + id
317
318     // Obtener el valor del select
319     const valorSelect = document.getElementsByName(select)[0].value;
320
321     // Eliminar el atr del DOM
322     const contenedor = document.getElementById(id);
323     contenedor.parentNode.removeChild(contenedor);
324
325     let nuevo = { ...venta };
326
327     // Eliminar el registro del objeto venta si el valor no está vacío
328     if (valorSelect !== '') {
329         delete nuevo[select];
330         setVenta(nuevo);
331     }
332
333     return {
334         handleChangeDatos,
335         handleChangeProductos,
336         handleChangeUnidades,
337         agregarSelect,
338         eliminarSelect,
339         venta
340     };
341 }
342

```

Anexo 12

Backup.sh

```

1  #!/bin/bash
2
3  # Configuración de la base de datos
4  DB_HOST="localhost"
5  DB_USER="root"
6  DB_PASSWORD="rootroot"
7  DB_NAME="ERP"
8  BACKUP_DIR="./backups/"
9  TIMESTAMP=$(date +%d-%m-%Y)
10
11 # Dump a la base de datos (copia)
12 mysqldump -h $DB_HOST -u $DB_USER -p$DB_PASSWORD $DB_NAME > $BACKUP_DIR/$DB_NAME-$TIMESTAMP.sql
13
14 # Eliminar respaldos antiguos de más de 3 días
15 find $BACKUP_DIR -type f -name "*.sql" -mtime +3 -exec rm {} \;
16
17 # Pasos para automatizar
18 # El demonio cron de Ubuntu para automatizar tareas
19 # crontab -e
20 # Se ejecuta todos los días a la 1 am
21 # 0 1 * * * /var/www/html/ruta del proyecto/backup.sh

```