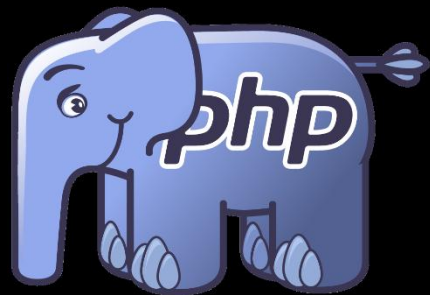


# Manual técnico



# Herramientas utilizadas y tecnologías

**Vscode:** es un editor de código para programadores gratuito, de código abierto multiplataforma. Capaz de adaptarse a cualquier lenguaje de programación

**Comopser ;** es un manejador de paquetes para PHP que proporciona un estándar para administrar, descargar e instalar dependencias y librerías. Similar a NPM en Node.js y Bundler en Ruby, Composer es la solución ideal cuando trabajamos en proyectos complejos que dependen de múltiples fuentes de instalación. En lugar de tener que descargar cada dependencia de forma manual, Composer hace esto de forma automática por nosotros.

**PHP** es un lenguaje de programación para desarrollar aplicaciones y crear sitios web que conquista cada día más seguidores. Fácil de usar y en constante perfeccionamiento es una opción segura para aquellos que desean trabajar en proyectos calificados y sin complicaciones.

**Larave:l** es un framework PHP diseñado para simplificar la creación de aplicaciones PHP modernas . Muchos desarrolladores lo utilizan para optimizar su proceso de desarrollo gracias a su robusto ecosistema, que aprovecha las capacidades integradas de Laravel y sus múltiples paquetes y extensiones compatibles.

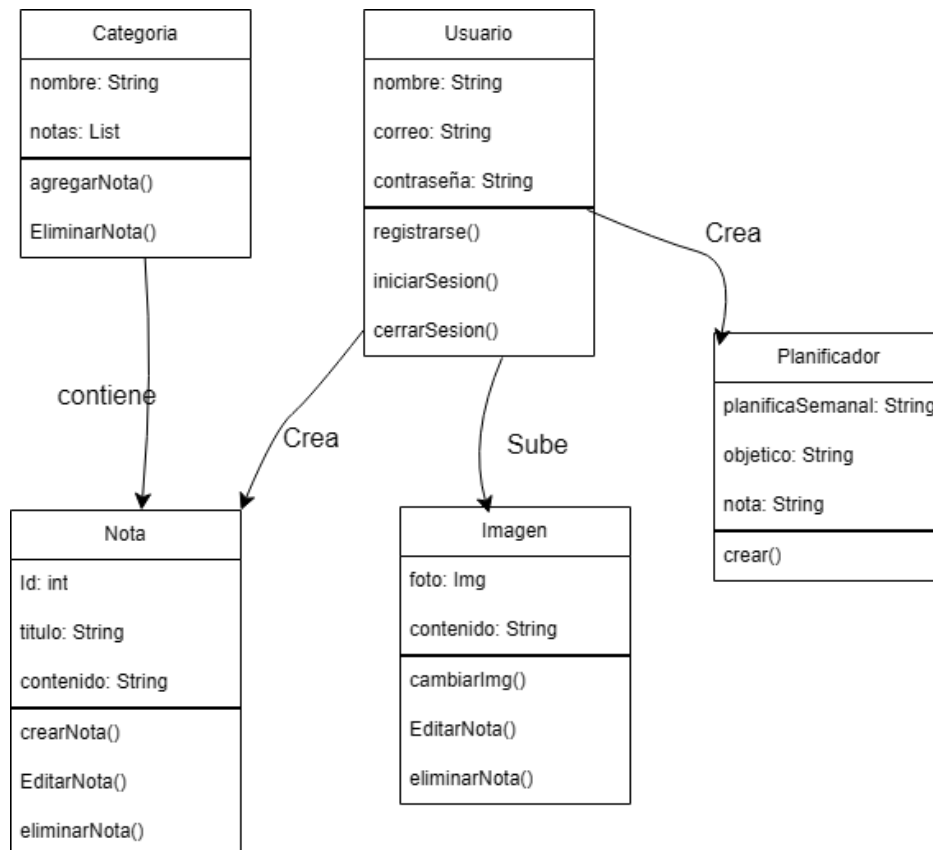
**Node.js** sirve para crear sitios web dinámicos muy eficientes, escritos con el lenguaje de programación para este caso lo ocupamos para darle el estilo con tailwind

**XAMPP** es una distribución de Apache completamente gratuita y fácil de instalar que contiene MariaDB, PHP y Perl. El paquete de instalación de XAMPP ha sido diseñado para ser increíblemente fácil de instalar y usar.

# Diagramas UML.

## Diagrama de clases:

Este diagrama muestra las clases que formar parte de la aplicación y sus relaciones.



**Usuario:** Son nuestros clientes los cuales para hacer uso de nuestra aplicación y por políticas de privacidad necesitan crear una cuenta y los datos que necesitamos es un nombre, un correo y una contraseña; Nuestros usuarios pueden registrar, iniciar sesión y cerrar sesión con los datos que nos proporcionan.

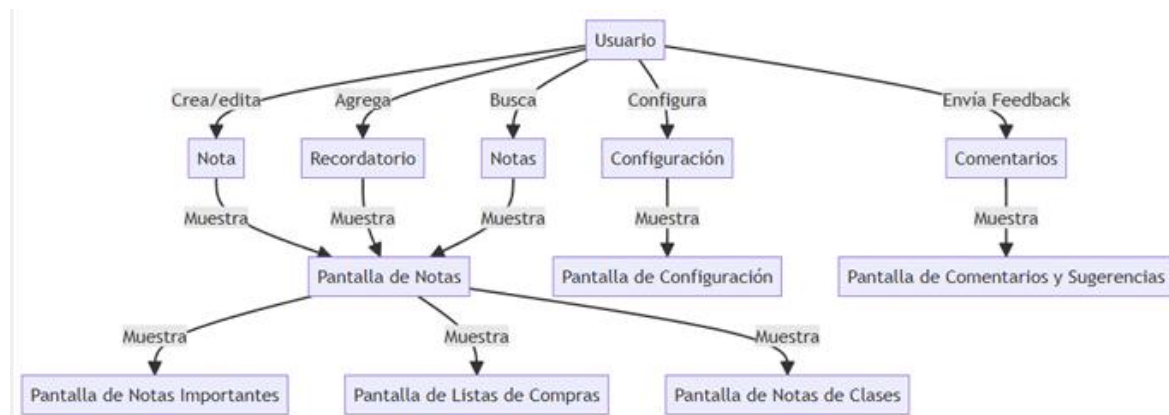
**Categoría:** Las notas están divididas en categoría las categorías son el tipo de nota a utilizar las categorías poseen un nombre y dentro de ella se guardan notas, nuestros usuarios pueden agregar o eliminar notas en nuestras categorías preestablecidas.

**Nota:** Las notas son creadas por nuestro usuario dentro de cada categoría, independiente del tipo de nota que sea las notas contarán con un id único que nos servirá para el registro, también debe tener un título, un contenido, las notas tienen que ser creadas, se pueden editar, eliminar y marcar.

**Plánificador:** La categoría de Planificador Semanal creada para los usuarios que necesitan tener un control sobre su semana, este se divide por cada día de la semana que servida como un título; junto a un objetivo y la nota.

**Imagen:** Proporciona una opción de adjuntar un archivo versión imagen manteniéndolo en un contenedor y la nota de esta.

## Diagrama de flujo del usuario:



## Descripción de los Elementos

- **Usuario:** Es el actor principal que interactúa con la aplicación. Puede crear, editar, buscar notas, agregar recordatorios y configurar la aplicación.
- **Nota:** Representa las notas que el usuario crea o edita. Se muestra en la pantalla de notas.
- **Notas:** Es la colección de todas las notas que el usuario puede buscar. Los resultados de la búsqueda se muestran en la pantalla de notas.
- **Configuración:** Representa las configuraciones de la aplicación que el usuario puede ajustar. Se muestra en la pantalla de configuración.
- **Pantalla de Notas:** Es la interfaz donde se muestran todas las notas, incluyendo las notas importantes, listas de compras y notas de clases.
- **Comentarios:** Representa el feedback que el usuario puede enviar sobre la aplicación. Se muestra en la pantalla de comentarios y sugerencias.
- **Pantalla de Comentarios y Sugerencias:** Es la interfaz donde los usuarios pueden enviar comentarios y sugerencias sobre la aplicación.

## Flujo de datos

**Creación y Edición:** El usuario crea o edita notas, que son gestionadas por el sistema.

**Búsqueda de Notas:** El usuario busca notas específicas, y los resultados se muestran en la pantalla de notas.

## Diagrama de flujo del administrador:



## Descripción de los Elementos

- **Administrador:** Es el actor principal que gestiona la aplicación, incluyendo usuarios, comentarios y reportes.
- **Usuarios:** Representa la base de datos de usuarios de la aplicación. El administrador puede gestionar (agregar, editar o eliminar) usuarios.
- **Comentarios:** Representa los comentarios y sugerencias enviados por los usuarios. El administrador puede revisar y gestionar estos comentarios.
- **Reportes:** Representa los reportes generados por el administrador sobre el uso de la aplicación y la actividad de los usuarios.
- **Configuración de la Aplicación:** Representa las configuraciones generales de la aplicación que el administrador puede ajustar.
- **Pantalla de Usuarios:** Es la interfaz donde se gestionan los datos de los usuarios.
- **Pantalla de Comentarios:** Es la interfaz donde se muestran los comentarios y sugerencias de los usuarios.
- **Pantalla de Reportes:** Es la interfaz donde se muestran los reportes generados por el administrador.
- **Pantalla de Configuración:** Es la interfaz donde se gestionan los parámetros de configuración de la aplicación.

## Flujo de Datos

- **Gestión de Usuarios:** El administrador gestiona la base de datos de usuarios, incluyendo la modificación de datos de usuario.
- **Revisión de Comentarios:** El administrador revisa los comentarios enviados por los usuarios y puede tomar acciones sobre ellos.
- **Generación de Reportes:** El administrador genera reportes sobre la actividad de la aplicación y los usuarios, que se muestran en la pantalla de reportes.
- **Configuración de la Aplicación:** El administrador ajusta la configuración de la aplicación, que se refleja en la pantalla de configuración.

## Uso de MVC

Capa	Objetivos	Funcionalidad
Modelo	Gestiona los datos y reglas de negocio	Interactúa con la base de datos (Ej. modelos User, Nota)
Vista	Muestra la información al usuario	Son los archivos Blade (.blade.php) que ves en el navegador
Controlador	Procesa la lógica y conecta Modelo y Vista	Recibe peticiones, procesa datos con Modelos y devuelve Vistas o JSON

**MVC (Modelo-Vista-Controlador)** utilizando PHP, con el objetivo de organizar el código de forma estructurada, separando la lógica de negocio, la presentación visual y el control de flujo de la aplicación.

## Flujo MVC

El usuario abre el formulario de login → Vista (login.blade.php)

Envía el formulario → AuthenticatedSessionController@store

El controlador valida los datos y autentica al usuario (usa modelo User)

Redirige a otra vista (ej. /dashboard)

## Modelo (Model)

Los modelos representan los datos y la lógica de negocio de la aplicación. Los modelos están representados como:

- **Usuario:** gestiona el registro, inicio de sesión y cierre de sesión.
- **Nota:** maneja la creación, edición, eliminación y marcado de notas importantes.
- **Categoría:** agrupa las notas por tipo.
- **Comentario:** permite a los usuarios enviar sugerencias.
- **Configuracion:** gestiona las preferencias del usuario.
- (Aunque se planteó el modelo Recordatorio, no fue implementado.)

Cada modelo se conecta con la base de datos MySQL para almacenar y recuperar información.

## Modelos (Carpeta: App\Models)

Los modelos representan la estructura de los datos y encapsulan la lógica relacionada con la base de datos. Se está usando Eloquent ORM de Laravel, facilitando la comunicación con la base de datos usando clases PHP.

Ejemplos:

Nota.php



- Representa una nota común.
- Tiene campos: titulo, contenido, y user\_id.
- Usa fillable para proteger contra asignación masiva y permitir solo estos campos.
- Hereda de Model, lo que lo conecta automáticamente con la tabla notas (por convención).

### NotaCompra.php

- Representa una nota de compra.
- Campos: titulo, checklist, user\_id.
- Tiene una relación **belongsTo** con User, indicando que cada nota de compra pertenece a un usuario.

### NotaFoto.php

- Representa una nota con una imagen asociada.
- Campos: user\_id, nota, foto.
- No hay relaciones definidas explícitamente (pero podría tener belongsTo o hasOne en una implementación más completa).

### Planificador.php

- Representa un plan semanal del usuario.
- Campos relacionados con los días de la semana (lunes a viernes), objetivos, notas, y user\_id.
- Ideal para un sistema de planificación personal o académica.

### User.php

- Modelo que representa al usuario del sistema.
- Hereda de Authenticatable, por lo tanto, es compatible con el sistema de autenticación de Laravel.
- Define campos protegidos (fillable, hidden, y casts) y activa notificaciones.
- También podría tener relaciones con otros modelos como hasMany(Nota::class) si se define.

## Controlador (Controllers)

- `RegisteredUserController`: Registrar usuarios.
- `AuthenticatedSessionController`: Iniciar/cerrar sesión.
- `VerifyEmailController`: Verificar el email del usuario.
- `NewPasswordController`: Guardar nueva contraseña.
- `ConfirmablePasswordController`: Confirmar la contraseña antes de una acción importante.

## Vistas(Views)

Uso de Blade en las vistas; es el motor de plantillas de Laravel. Permite mezclar HTML con código PHP utilizando una sintaxis limpia y sencilla.

Las instrucciones comienzan con `@`, por ejemplo: `@if`, `@foreach`, `@csrf`, `@extends`, etc.

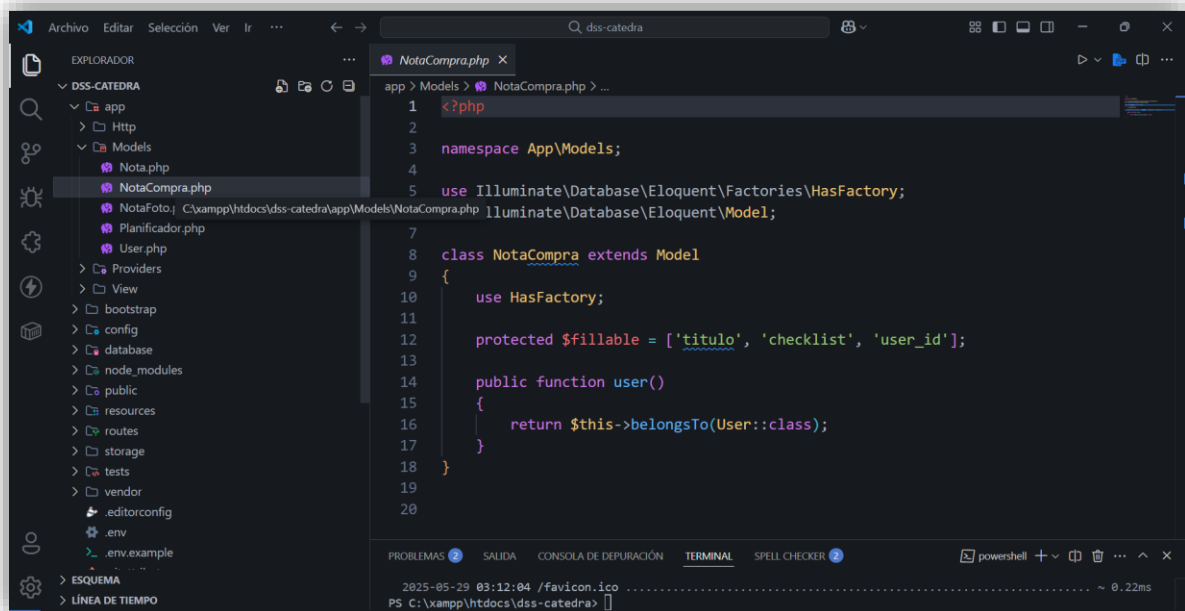
Permite reutilización de plantillas mediante `@extends`, `@section`, `@yield` y componentes (`<x-component>`).

Blade es compilado en PHP puro y cacheado automáticamente por Laravel, por lo que es rápido y seguro.

- `Login.blade.php`: Vista para iniciar sesión en el sistema e insertar rutas y tokens CSRF.
- `Register.blade.php`: Formulario de registro de nuevos usuarios con campos para nombre, email, contraseña y confirmación.
- `editar.blade.php`: Vista para editar una nota existente reutilizar layout, insertar datos dinámicos y rutas.
- `Fotos.blade.php`: Crear notas acompañadas de imágenes y mostrarlas en una lista dinamida de imágenes y notas.

- **index.blade.php:** Mostrar una lista dinámica de todas las notas del usuario con botones de edición y eliminación.
- **planificador.blade.php:** Permitir al usuario planificar su semana y guardar objetivos y notas; al final, se muestra un resumen del planificador.

## Capturas del código



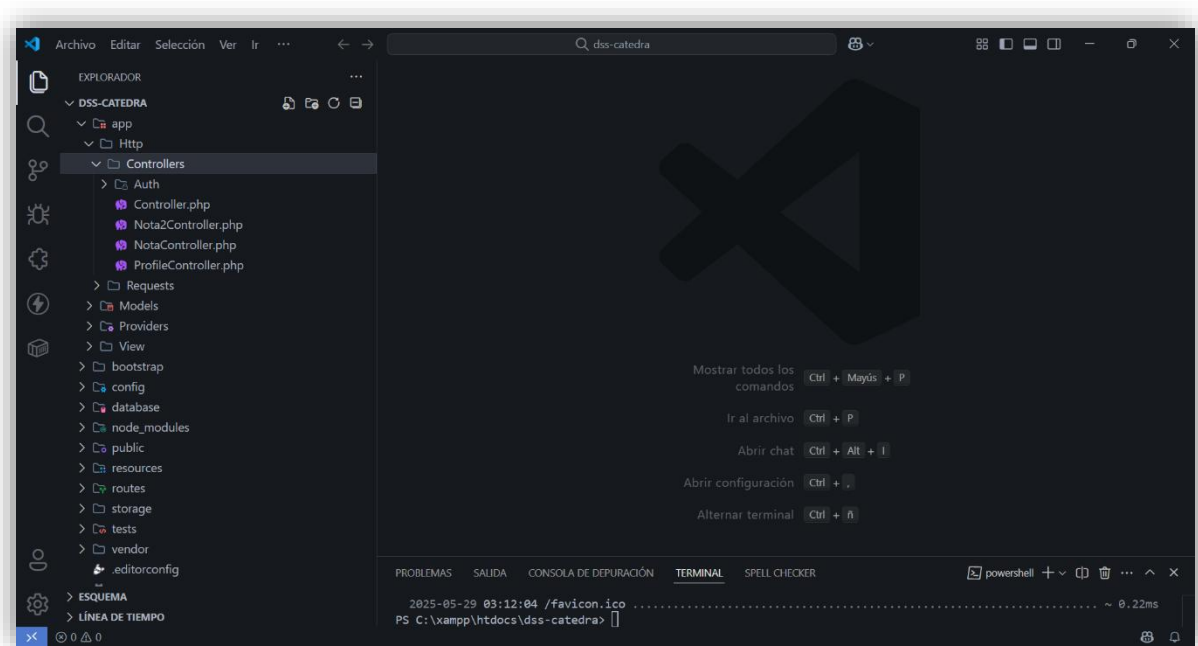
The screenshot shows the Visual Studio Code editor with the file `NotaCompra.php` open. The file explorer on the left shows the project structure, including the `Models` directory. The code in the editor is as follows:

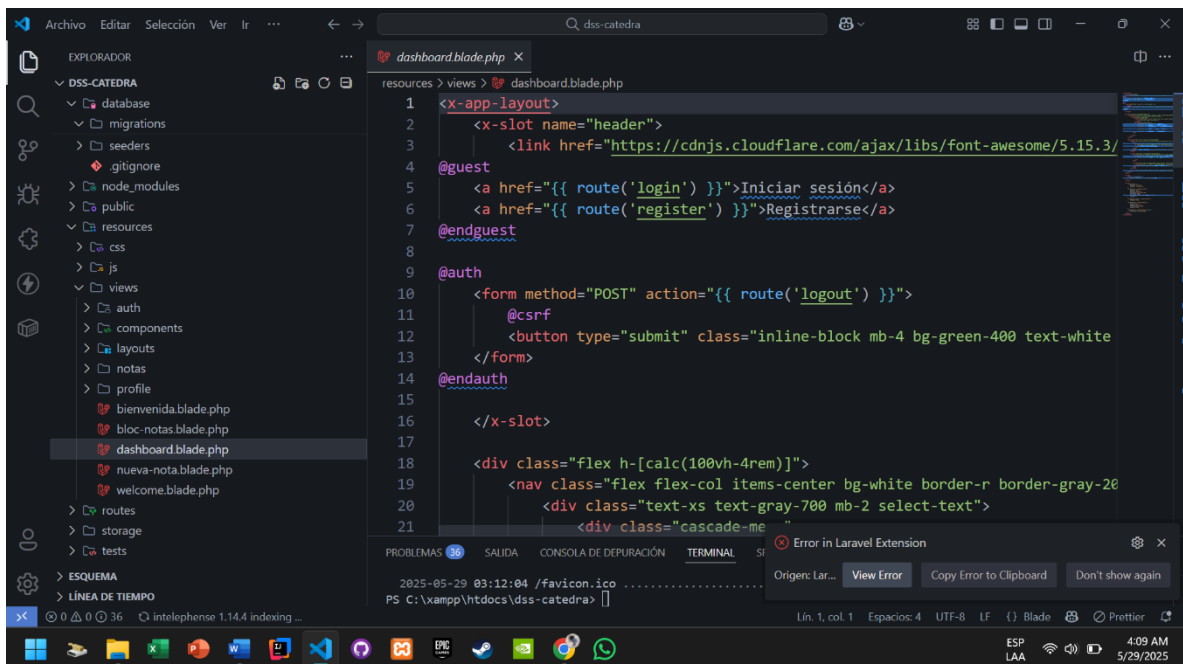
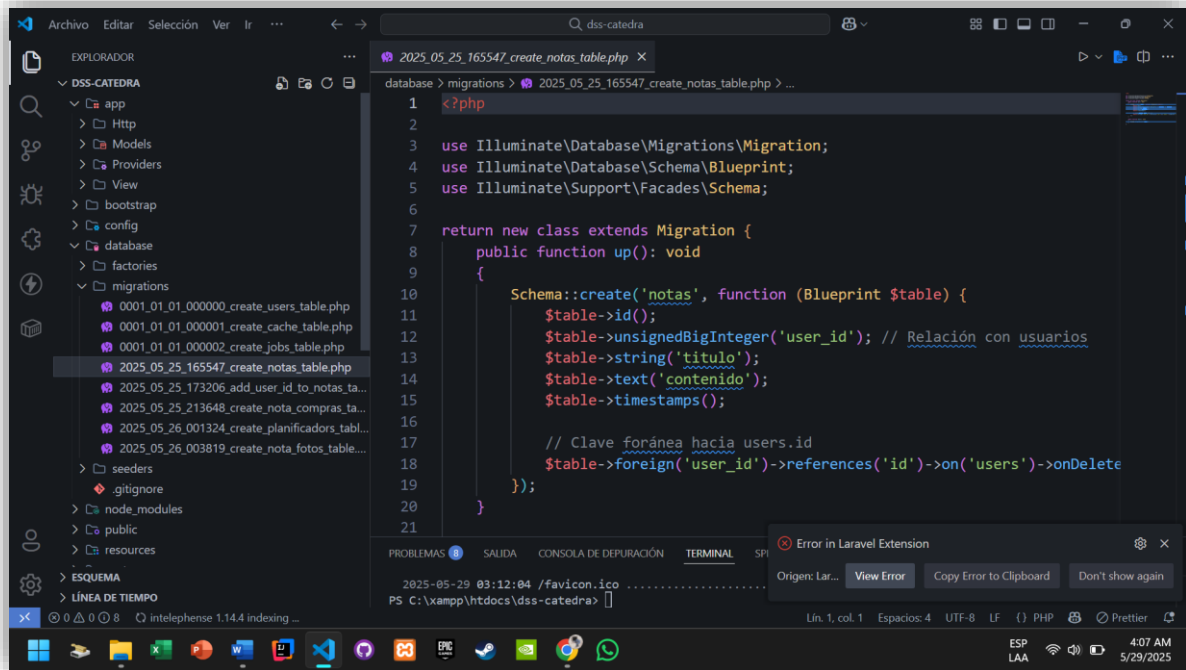
```

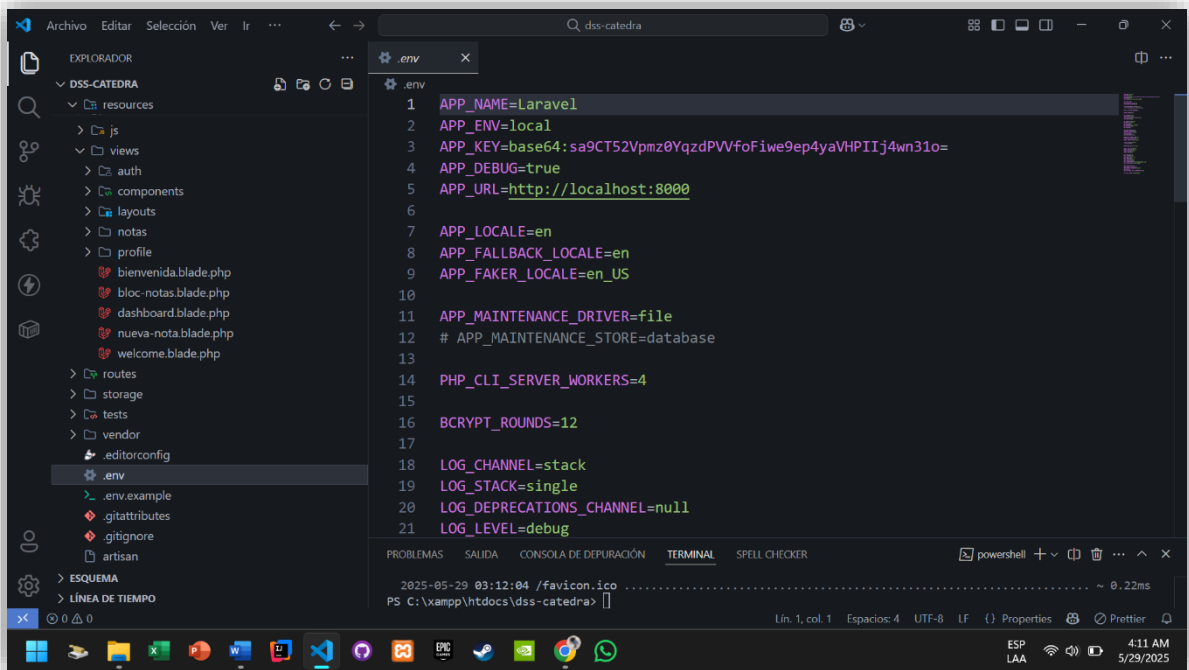
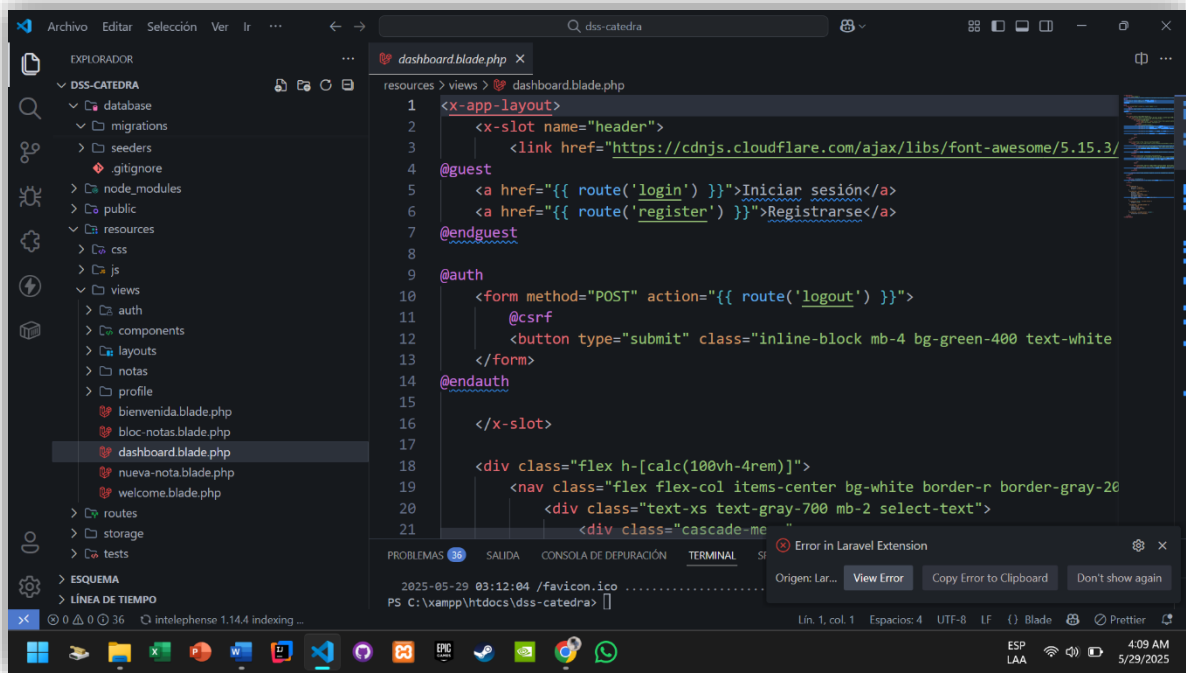
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  class NotaCompra extends Model
9  {
10     use HasFactory;
11
12     protected $fillable = ['titulo', 'checklist', 'user_id'];
13
14     public function user()
15     {
16         return $this->belongsTo(User::class);
17     }
18 }
19
20

```

The terminal at the bottom shows the command prompt with the path `C:\xampp\htdocs\dss-catedra`.







localhost/phpmyadmin/index.php?route=/sql&pos=0&db=dss\_catedra&table=users

phpMyAdmin

Reciente Favoritas

Nueva

- dss\_catedra
  - Nueva
  - cache
  - cache\_locks
  - failed\_jobs
  - jobs
  - job\_batches
  - migrations
  - notas
  - nota\_compras
  - nota\_fotos
  - password\_reset\_tokens
  - planificadors
  - sessions
  - users
- information\_schema
- mysql
- performance\_schema
- phpmyadmin

Examinar Estructura SQL Buscar Insertar Exportar Importar Privilegios Operaciones Más

Perfilando [ Editar en línea ] [ Editar ] [ Explicar SQL ] [ Crear código PHP ] [ Actualizar ]

☐ Mostrar todo | Número de filas: 25 | Filtrar filas: Buscar en esta tabla | Ordenar según la clave: Ninguna

Opciones extra

		id	name	email	email_verified_at	password
<input type="checkbox"/>	Editar Copiar Borrar	1	omar arturo	omar@gmail.com	NULL	\$2y\$12\$aBRxcnTvhGbrwcmSZT3GhOcMUIOv/V1/akjeBOKnI2L...
<input type="checkbox"/>	Editar Copiar Borrar	2	juan	juan@gmail.com	NULL	\$2y\$12\$opUjgRkAP8jmjBdoDZ9iZOCrL.G2pTilagHAcu9wbXRF...
<input type="checkbox"/>	Editar Copiar Borrar	3	link	link@gmail.com	NULL	\$2y\$12\$k8MBEv4RVbCL6SnlbKEroOzE3IHWvum6ZNII7u.VU/...
<input type="checkbox"/>	Editar Copiar Borrar	4	nelson	nelson@gmail.com	NULL	\$2y\$12\$RhUEdzVT70hNdAGwX6sTw.naOD77pBXeUABs6DwtOc...
<input type="checkbox"/>	Editar Copiar Borrar	5	sebastian	sebastian@gmail.com	NULL	\$2y\$12\$VklHcdACIEPotVEXC7I7mOR5n5BvucJ3eHzjh0GW6l...
<input type="checkbox"/>	Editar Copiar Borrar	6	pedro	pedro@gmail.com	NULL	\$2y\$12\$dNIZ8/X6hSIWNubTt3ZFhOChZCbq5zgDWEeRnnkQE...
<input type="checkbox"/>	Editar Copiar Borrar	7	sara	sara21@gmail.com	NULL	\$2y\$12\$9d.ImIEFIUGNOYHkgWP9qOt.cZ0SP1.OBQOxS8IRG2O...
<input type="checkbox"/>	Editar Copiar Borrar	8	carlos	carlos231@gmail.com	NULL	\$2y\$12\$LvWwPh.T0EujcCSxA9blewCkk9VPi82ayASxwdAx2M...
<input type="checkbox"/>	Editar Copiar Borrar	9	odalez	odalez@gmail.com	NULL	\$2y\$12\$OhtNRqndQj3QDnXiwzFUOO5xUjnwBf4qj3hujNUzCbz...

Consola