Ciclo Formativo Grado Superior

DESARROLLO DE APLICACIONES WEB

Curso 2023-2024

Rubén Torres López

Estudio Pilates

Tutor Individual: EVA MARIA RIVAS OJANGUREN

Tutor Colectivo: EVA MARIA RIVAS OJANGUREN

1. ÍNDICE

Contenido

1. ÍNDICE 2
2. ÍNDICE DE FIGURAS 3
3. INTRODUCCIÓN 3
   3.1. Presentación y Objetivos 3
   3.2. Contexto 3
   3.3. Planteamiento del problema 4
   3.4. Recursos 5
4. ESPECIFICACION DE REQUISITOS 5
   4.1. Introducción 5
   4.2. Descripción general 5
      4.2.1. Requisitos funcionales 6
      4.2.2. Requerimientos de interfaces Externas 7
      4.2.3. Requerimientos de Rendimiento. 7
      4.2.4. Obligaciones del Diseño. 7
      4.2.5. Atributos de la Aplicación Usando Laravel 8
5. ANALISIS 8
   5.1. Introducción 8
   5.2. Diagrama de clases 9
   5.3. Diagramas de uso 9
6. DISEÑO 10
   6.1. Modelo 10
   6.2. Vista 10
   6.3. Controlador 10
   6.4. Funcionamiento del MVC 10
7. IMPLEMENTACION 11
   7.1. Configuración del entorno de desarrollo 11
   7.2. Establecimiento de la base de datos 11
   7.3. Desarrollo del backend 21
   7.4. Desarrollo del front-end 29
      7.4.1. LOGIN 29
      7.4.2. REGISTER 30
      7.4.3. INICIO DE SESION DEL ADMIN 30
      7.4.4. CONTROL ALUMNOS 31
   7.5. EMPLEADOS GESTION 34
   7.6. GESTION GRUPOS Y CLASES 35

7.7. VISTA FACTURAS 35
7.8. HORARIOS 36
7.9. INICIO SESION DE ALUMNO 36
7.10. REGISTRO DE EMPLEADO O ALMUNO 37
7.11. PERFIL DE EMPLEADO 38
7.12. FACTURACION ALUMNO 39
8. EVALUACION 41
8.1. Introducción 41
8.2. Validación de enlaces 41
8.3. Validación de la resolución 42
9. COONCLUSION 44
9.1. Valoración personal 44
9.2. Posibles ampliaciones 44
9.3. BIBLIOGRAFIA 45
9.4. IMPLEMENTACION 45

2. ÍNDICE DE FIGURAS

3. INTRODUCCIÓN

3.1. Presentación y Objetivos

Este proyecto se enfoca en el desarrollo de un centro de control integral para un estudio de pilates, destinado a mejorar la interacción y gestión de alumnos, empleados y administradores. El objetivo principal es crear una plataforma digital que centralice y automatice las operaciones diarias del estudio, desde la administración de clases hasta la gestión de personal y alumnos.

Solo los usuarios registrados tendrán acceso a la aplicación. La aplicación pretende facilitar la forma de realizar reservas de clases, la creación de registros horarios de las clases, pago de la suscripción, generador de facturas, la gestión de alumnos, empleados…

3.2. Contexto

Este proyecto surgió tras una revisión realizada por un familiar que está involucrado en el negocio. Él notó que había muchas áreas de mejora en la gestión del estudio de Pilates. Actualmente, usamos una aplicación que, aunque ha sido útil, está bastante desactualizada. Está llena de código antiguo y no cumple con nuestras necesidades actuales.

Los problemas que enfrentamos son:

- Tecnología Obsoleta: Estamos usando prácticas de programación anticuadas. Esto nos limita mucho, hace que el sistema sea más lento y menos seguro, y nos impide añadir nuevas funcionalidades que son esenciales hoy en día.
- Interfaz de Usuario Anticuada: La app que tenemos es difícil de usar y no tiene un aspecto moderno, lo que complica la experiencia de los usuarios.
- Ineficiencia Operativa: La falta de automatización y características modernas hace que gestionar el estudio sea complicado, afectando tanto a empleados como a alumnos.

Por ello, hemos decidido que es hora de actualizar el sistema del estudio. La idea es reemplazar el sistema antiguo con una solución más moderna, utilizando los avances tecnológicos recientes en desarrollo de aplicaciones web. Buscamos solucionar los problemas actuales y también ofrecer una plataforma que se pueda adaptar a necesidades y mejoras futuras.

Las ventajas de esta actualización son claras:

- Mejora en la Eficiencia Operativa: Con la implementación de código moderno y metodologías de desarrollo actuales, el sistema será más rápido, seguro y fácil de mantener.
- Experiencia de Usuario Mejorada: Actualizaremos la interfaz de usuario para que sea más intuitiva y agradable, tanto para los alumnos como para los empleados.
- Adaptabilidad y Escalabilidad: La nueva aplicación nos permitirá añadir fácilmente nuevas funcionalidades y adaptarnos a los cambios en el negocio o en las tendencias del mercado.

3.3. Planteamiento del problema

Actualmente, el estudio de Pilates se enfrenta a varios desafíos operativos y tecnológicos críticos:

- Tecnología Obsoleta: La aplicación en uso, construida con código y prácticas antiguas, limita nuestra capacidad para integrar nuevas funcionalidades y salvaguardar la seguridad de los datos.
- Interfaz de Usuario Desactualizada: La interfaz actual no cumple con los estándares modernos, resultando en una experiencia de usuario poco intuitiva y atractiva.
- Ineficiencias Operativas: La falta de automatización y funciones avanzadas obstaculiza una gestión eficiente, afectando tanto a empleados como a clientes.

Estos problemas nos impulsan a remodelar y modernizar el centro de control del estudio, buscando no solo solucionar estos inconvenientes sino también adaptarnos a las tendencias actuales y futuras del mercado.

La aplicación contará con tres tipos de usuarios; alumnos, empleados y administrador.

3.4. Recursos

En este proyecto usare las tecnologías siguientes:

- PHP con el framework Laravel
- JavaScript
- HTML
- CSS
- Eloquent de Laravel para manejo de base de datos
- Phpmyadmin
- Stripe

4. ESPECIFICACION DE REQUISITOS

4.1. Introducción

En esta sección explico con detalle cada una de las funcionalidades de esta aplicación.

4.2. Descripción general

El usuario alumno debe ser capaz de reservar la clase, ver que clases tiene reservadas, y ver la lista de clases asistidas. Además, este podrá realizar el pago de la suscripción tanto mensual como anual. Cuando se realice el pago, se generará un documento pdf que se descargara en el navegador y se almacenara en el servidor de manera local.

El usuario empleado accederá a su perfil donde podrá ver el listado de alumnos asignados a su grupo.

El usuario admin podrá realizar las siguientes tareas.

- Gestionar tanto alumnos como empleados pudiendo añadir, editar o eliminar cualquiera d ellos.
- Podrá gestionar tanto grupos como clases al igual que los alumnos y empleados, podrá crear editar y borrar.
- Podrá hacer lo mismo con los registros horarios, necesarios para mostrar después en el alumno
- Podrá acceder al generador de código de invitación para el alumno, el cual es un formulario con el correo electrónico del alumno.

- Podrá ver el listado de facturas generadas por los pagos de los alumnos.

Aquí un desglose de como seria por cada usuario los requisitos funcionales:

4.2.1. Requisitos funcionales

4.2.1.1. Usuario 'alumno'

Tipo	*Identificador*	*Prioridad*	*Descripción Corta*	*Descripción Detallada*	*Entrada*	*Proceso*	*Salida*
Funcional	RF-A1	Alta	Registro con Código	Los alumnos que han visitado la tienda y tienen un código, lo usan para registrarse.	Código generado en la tienda.	Introducir código en el formulario de registro.	Usuario registrado.
Funcional	RF-A2	Media	Registro sin Código	Los alumnos sin código se registran, pero quedan pendientes de pago y son redirigidos a facturación.	Datos personales.	Rellenar formulario alternativo.	Usuario registrado como pendiente de pago.
Funcional	RF-A3	Alta	Generación de Factura	Al realizar el pago, se genera una factura en PDF que se guarda localmente y se descarga en el navegador.	Información de pago.	Proceso de pago y generación de factura.	Factura en PDF.
Funcional	RF-A4	Alta	Reserva de Clase	Los alumnos pueden reservar la primera clase disponible según su grupo y nivel.	Selección de grupo y nivel.	Elegir y reservar clase disponible.	Clase reservada.

4.2.1.2. Usuario 'empleado'

Tipo	*Identificador*	*Prioridad*	*Descripción Corta*	*Descripción Detallada*	*Entrada*	*Proceso*	*Salida*
Funcional	RF-E1	Alta	Acceso a Perfil	Los empleados acceden a su perfil para ver el listado de alumnos asignados a su clase.	Credenciales de empleado.	Iniciar sesión y acceder al perfil.	Lista de alumnos asignad
Funcional	RF-E2	Alta	Inicio de Sesión de Empleado	Los empleados pueden iniciar sesión en el sistema para acceder a su perfil y funcionalidades relacionadas.	Credenciales de empleado.	Autenticación en el sistema.	Acceso a perfil y funcionalidades de empleado.

4.2.1.3. Usuario 'admin'

Tipo	*Identificador*	*Prioridad*	*Descripción Corta*	*Descripción Detallada*	*Entrada*	*Proceso*	*Salida*
Funcional	RF-AD1	Alta	Gestión de Usuarios	El admin puede gestionar alumnos y empleados.	Datos de usuarios.	Crear, editar, borrar usuarios.	Usuarios gestionados.
Funcional	RF-AD2	Alta	Gestión de Horarios	El admin gestiona los horarios de clases y grupos.	Información de clases y grupos.	Crear, editar, borrar horarios.	Horarios actualizados.
Funcional	RF-AD3	Alta	Visualización de Facturas	El admin puede ver y descargar facturas generadas.	Facturas generadas.	Acceder y descargar facturas.	Facturas descargadas.
Funcional	RF-AD4	Media	Generación de Código	El admin genera códigos de invitación para alumnos y los envía por correo.	Información del alumno.	Crear y enviar código.	Código enviado.
Funcional	RF-AD5	Alta	Cambio de Grupos y Clases por Admin	El admin puede reasignar alumnos a diferentes grupos y cambiar la asignación de clase de los empleados, usando el formulario de edición o directamente desde el menú de grupos y clases.	Selección de alumno/empleado y nuevo grupo/clase.	Modificar asignaciones en el sistema.	Alumnos y empleados reasignados a nuevos grupos y clases.
Funcional	RF-AD6	Alta	Inicio de Sesión de Admin	Los administradores pueden iniciar sesión en el sistema para acceder a sus funcionalidades exclusivas.	Credenciales de admin.	Autenticación en el sistema.	Acceso a funcionalidades de admin.
Funcional	RF-AD7	Alta	Gestión de Grupos y Clases	El admin tiene la capacidad de crear, editar y borrar grupos y clases dentro del sistema.	Información de grupo o clase.	El admin accede al módulo de gestión de grupos y clases, donde puede añadir nuevos grupos o clases, modificar los existentes o eliminarlos según sea necesario.	Grupos y clases creados, actualizados o eliminados en el sistema.

Funcional	RF-AD4	Alta	Generación y Envío de Códigos	El admin puede generar códigos de invitación para nuevos alumnos y enviarlos directamente a través de correo electrónico.	Información del destinatario del código.	El admin crea un código de invitación en el sistema y lo envía al correo electrónico del destinatario.	Código de invitación generado y enviado por email.

4.2.1.4. *Resumen de requisitos*

Tipo de Usuario	*Tipo de Requisito*	*Identificador*	*Descripción Corta*
Alumno	Funcional	RF-A1	Registro con Código
Alumno	Funcional	RF-A2	Registro sin Código
Alumno	Funcional	RF-A3	Generación de Factura
Alumno	Funcional	RF-A4	Reserva de Clase
Empleado	Funcional	RF-E1	Acceso a Perfil
Empleado	Funcional	RF-E2	Inicio de Sesión de Empleado
Admin	Funcional	RF-AD1	Gestión de Usuarios
Admin	Funcional	RF-AD2	Gestión de Horarios
Admin	Funcional	RF-AD3	Visualización y Descarga de Facturas
Admin	Funcional	RF-AD4	Generación de Código
Admin	Funcional	RF-AD5	Cambio de Grupos y Clases
Admin	Funcional	RF-AD6	Inicio de Sesión de Admin
Admin	Funcional	RF-AD7	Gestión de Grupos y Clases
Todos	No Funcional	RNF-1	Seguridad de Datos
Todos	No Funcional	RNF-2	Rendimiento
Todos	No Funcional	RNF-3	Usabilidad
Todos	No Funcional	RNF-4	Escalabilidad

4.2.1.5. *Requisitos no funcionales*

Tipo	*Identificador*	*Prioridad*	*Descripción Corta*	*Descripción Detallada*
No Funcional	RNF-1	Alta	Seguridad de Datos	El sistema debe asegurar la protección de los datos personales y financieros.
No Funcional	RNF-2	Alta	Rendimiento	El sistema debe ser rápido y eficiente en el procesamiento de datos y transacciones.
No Funcional	RNF-3	Alta	Usabilidad	La interfaz de usuario debe ser intuitiva y fácil de navegar para todos los usuarios.
No Funcional	RNF-4	Media	Escalabilidad	El sistema debe ser capaz de adaptarse a un creciente número de usuarios y datos.

4.2.2. Requerimientos de interfaces Externas

4.2.2.1. *Interfaces de los usuarios*

Se intenta respetar una misma interfaz en todos los módulos, aunque puede variar de una a otra ventana. El encabezado será común para todas las páginas de la aplicación.

4.2.2.2. *Interfaces hardware*

Al ser una aplicación web se podrá visualizar desde cualquier sistema operativo.

4.2.2.3. *Interfaces Software*

La aplicación funcionará en cualquier equipo que tenga navegador web y

conexión a Internet.

4.2.2.4. *Interfaces de Comunicaciones.*

Si se quisiera publicar la aplicación se realizaría utilizando el protocolo HTTP mediante TCP/IP. Al no tener que publicarse se muestra en entorno local.

4.2.3. Requerimientos de Rendimiento.

La aplicación está diseñada para funcionar con un rendimiento óptimo en cualquier tipo de software o hardware. Los únicos aspectos que podrían influir en su eficiencia son la velocidad de conexión a Internet del usuario y la del servidor.

4.2.4. Obligaciones del Diseño.

4.2.4.1. *Estándares Cumplidos.*

Nuestra aplicación está diseñada para seguir los estándares de seguridad de sitios web con acceso protegido. Implementaremos un sistema de autenticación para asegurar que solo los administradores tengan acceso a las áreas restringidas del sitio. Además, hemos elegido el

español como idioma principal para la interfaz de la web, considerando que está dirigida principalmente a clientes en España.

4.2.4.2. Limitaciones Hardware.

Dado que es una aplicación web local, no existen restricciones significativas en cuanto al hardware necesario para ejecutarla. La única situación en la que podrían surgir limitaciones es si el dispositivo es muy antiguo y no es capaz de soportar un entorno de desarrollo, como XAMPP, que se requiere para su funcionamiento.

4.2.5. Atributos de la Aplicación Usando Laravel

4.2.5.1. Seguridad

La seguridad es crucial en nuestra aplicación, especialmente en la administración del sitio web. Gracias a Laravel, la gestión de la base de datos es segura y robusta. Solo los usuarios con el rol de administrador tendrán acceso a funciones críticas como ver, editar, borrar o registrar usuarios, aprovechando las capacidades de autenticación y autorización de Laravel.

4.2.5.2. Facilidades de Mantenimiento

El mantenimiento de la aplicación se facilita con Laravel, que ofrece un manejo eficiente de la base de datos y conexiones con integraciones externas. Las actualizaciones y la adición de nuevas funcionalidades son más sencillas y seguras, gracias a la arquitectura y las herramientas que proporciona Laravel.

4.2.5.3. Portabilidad

Desarrollada con Laravel, una tecnología basada en PHP, nuestra aplicación es altamente portable. Es compatible con cualquier plataforma que soporte PHP y accesible desde cualquier navegador web, garantizando una amplia disponibilidad y flexibilidad.

4.2.5.4. Otros Requerimientos

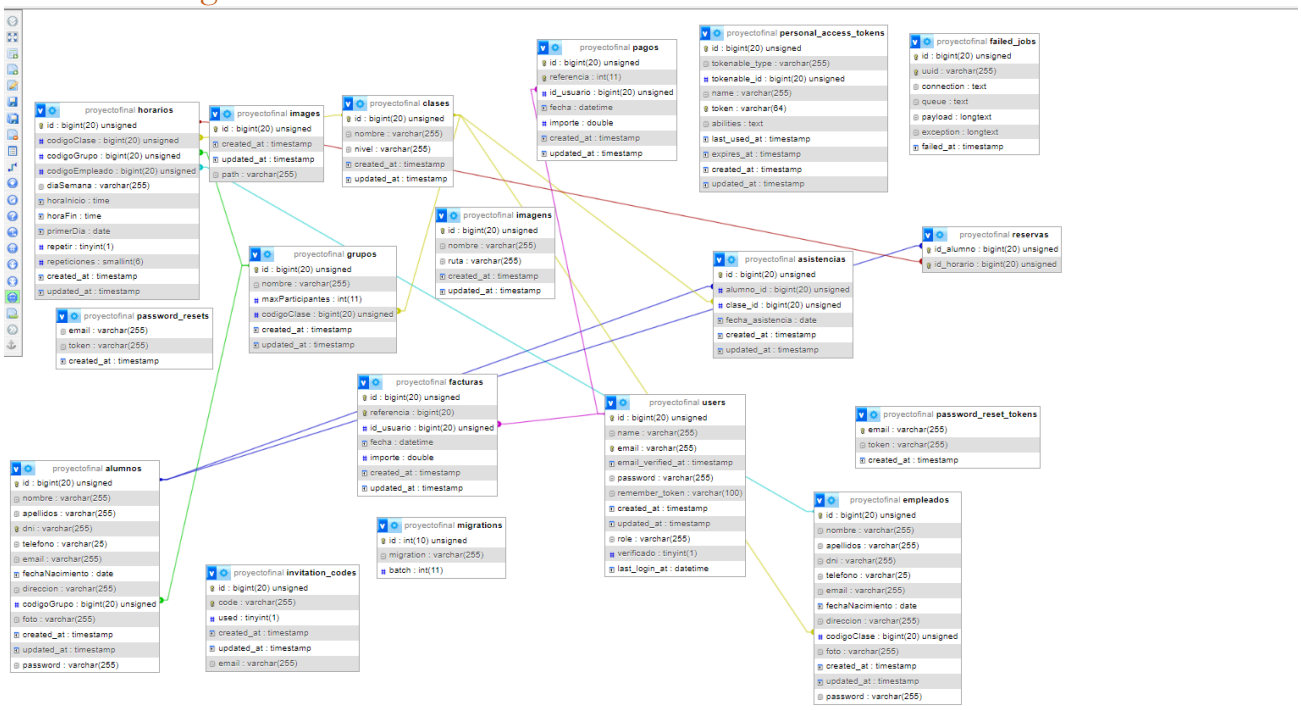
La aplicación utiliza una base de datos MySQL (phpMyAdmin) para almacenar toda la información relevante. Laravel facilita la realización de consultas a la base de datos de manera eficiente y segura a través de PHP, aprovechando su ORM Eloquent para una mejor gestión de los datos.

5. ANALISIS

5.1. Introducción

En el análisis de nuestra aplicación web, hemos adoptado las directrices del UML (Lenguaje Unificado de Modelado). UML nos brinda la posibilidad de emplear diversos tipos de diagramas para entender a fondo nuestra aplicación. Esto nos permite representar de manera estandarizada las funcionalidades, requisitos y otras características identificadas en el sistema. Para nuestro análisis, hemos decidido utilizar principalmente los diagramas de clases y de casos de uso.

5.2. Diagrama de clases



La base de datos consta de las siguientes tablas:

- Alumnos
- Empleados
    - Estas dos tablas son iguales, y tienen los campos básicos para almacenar la información necesaria del usuario
- Horarios
    - Esta tabla almacena los registros horarios relacionando así los grupos los usuarios las reservas y las clases
- Grupos
- Clases
    - Dos clases básicas que sirven para almacenar la información de cada uno
- Users
    - Tabla generada por laravel que sirve para la autenticacion
- Migrations
    - Tabla en la cual están las migraciones necesarias para que la app funcione con su base de datos correctamente
- Invitation_codes
    - Donde se almacenan los códigos de invitación del alumno
- Imágenes
    - Para las imágenes que se suban en cada perfil de usuario
- Pagos
    - Se almacenan los datos de los pagos de las suscripciones del alumno
- Reservas
    - Las reservas realizadas por cada alumno
- Asistencias
    - Para almacenar cuantas veces asiste un alumno

5.3. Diagramas de uso

6. DISEÑO

6.1. Modelo

El Modelo es el núcleo de la lógica de negocio y el manejo de datos. Se encarga de la gestión y almacenamiento de la información, típicamente a través de una base de datos. Incluye las clases y métodos que interactúan directamente con la base de datos y que implementan las reglas de negocio de la aplicación.

6.2. Vista

La Vista es la capa de presentación de la aplicación. Su función es presentar los datos al usuario de manera comprensible y permitir la interacción con el sistema. Es la interfaz gráfica que el usuario utiliza para interactuar con la aplicación.

6.3. Controlador

El Controlador actúa como mediador entre la Vista y el Modelo. Procesa las entradas del usuario, enviadas a través de la Vista, solicitando datos al Modelo y luego pasando esos datos de vuelta a la Vista para su presentación. Esencialmente, controla el flujo de datos en la aplicación y las interacciones del usuario.

6.4. Funcionamiento del MVC

El proceso del patrón MVC funciona de la siguiente manera:

- El usuario realiza una acción que genera una petición.

- El Controlador captura esta petición y llama al Modelo apropiado para manejar la solicitud.
- El Modelo procesa la petición, interactuando con la base de datos si es necesario.
- Una vez que el Modelo procesa la información, se la pasa al Controlador.
- El Controlador, a su vez, envía estos datos a la Vista.
- Finalmente, la Vista formatea los datos y los muestra al usuario.

7. IMPLEMENTACION

7.1. Configuración del entorno de desarrollo

Para el desarrollo del proyecto he utilizado el servidor local XAMPP, para editar el código he usado Visual Studio Code y he utilizado el framework de Laravel-php para el desarrollo del crud

Para el uso de versiones he usado git junto con gitlab

7.2. Establecimiento de la base de datos

Para configurar la base de datos he usado la siguiente configuración en el archivo .env

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=proyectoFinal
DB_USERNAME=root
DB_PASSWORD=

STRIPE_KEY=pk_test_51OIbF1HiCTgEgoJNeVItAh0vUIFk2nRE5swppQc4sLlEfxdQow9aZEqrJxannShi8adMbCemBzBv8RaxKvnqiRXz00
STRIPE_SECRET=sk_test_51OIbF1HiCTgEgoJNHYzbZLdenn0kBBOKxSBpPMay9qnAeVHkRam4Tl5Q2w3eCugdp2hgvB8H5gZVjhWrHdCQOXd

BROADCAST_DRIVER=log
CACHE_DRIVER=file
FILESYSTEM_DISK=local
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120

MEMCACHED_HOST=127.0.0.1

REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379
```

Aquí muestro algunas de las migraciones necesarias para el funcionamiento de la bd

```php
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('asistencias', function (Blueprint $table) {
            $table->id();
            $table->foreignId('alumno_id')->constrained()->onDelete('cascade');
            $table->foreignId('clase_id')->constrained()->onDelete('cascade');
            $table->date('fecha_asistencia');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('asistencias');
    }
};
```

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('horarios', function (Blueprint $table) {
            $table->id();
            $table->unsignedBigInteger('codigoClase');
            $table->unsignedBigInteger('codigoGrupo');
            $table->unsignedBigInteger('codigoEmpleado');
            $table->string('diaSemana');
            $table->time('horaInicio');
            $table->time('horaFin');
            $table->date('primerDia');
            $table->boolean('repetir')->nullable();
            $table->smallInteger('repeticiones')->nullable();
            $table->foreign('codigoClase')->references('id')->on('clases');
            $table->foreign('codigoGrupo')->references('id')->on('grupos');
            $table->foreign('codigoEmpleado')->references('id')->on('empleados');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
```

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('empleados', function (Blueprint $table) {
            $table->id();
            $table->string('nombre');
            $table->string('apellidos');
            $table->string('dni');
            $table->string('telefono', 25);
            $table->string('email');
            $table->date('fechaNacimiento');
            $table->string('direccion');
            $table->unsignedBigInteger('codigoClase');
            $table->string('foto');
            $table->foreign('codigoClase')->references('id')->on('clases');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('empleados');
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('grupos', function (Blueprint $table) {
            $table->id();
            $table->string('nombre');
            $table->integer('maxParticipantes');
            $table->unsignedBigInteger('codigoClase');
            $table->foreign('codigoClase')->references('id')->on('clases');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('grupos');
    }
};
```

database > migrations > 2023_08_06_200000_create_clases_table.php

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('clases', function (Blueprint $table) {
            $table->id();
            $table->string('nombre');
            $table->string('nivel');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('clases');
    }
};
```

database > migrations > 2023_08_06_200000_create_clases_table.php

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('clases', function (Blueprint $table) {
            $table->id();
            $table->string('nombre');
            $table->string('nivel');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('clases');
    }
};
```

Ademas he hecho uso de los seeders y factorys para la creacion de datos aleatorios en la bd y facilitar asi el funcionamiento de la bd en el desarrollo.

```php
 */
0 references | 0 overrides
public function definition(): array
{

    // Crear una instancia de Faker
    $faker = FakerFactory::create();

    // Expresion regular
    $regexPattern = '/^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]+$/';

    // Inicializacion de la variable para guardar el valor correcto
    $randomValue = '';

    // genera el valor hasta que cumpla la condicion
    do {
        $randomValue = $faker->asciify('**********'); // Genera un valor de 10 caracteres aleatorios
    } while (!preg_match($regexPattern, $randomValue));

    return [
        'nombre' => $this->faker->name,
        'apellidos' => $this->faker->lastName,
        'dni' => $this->faker->unique()->numerify('########A'),
        'telefono' => $this->faker->phoneNumber,
        'email' => $this->faker->unique()->safeEmail,
        'fechaNacimiento' => $this->faker->dateTime,
        'direccion' => $this->faker->address,
        'codigoGrupo' => '1',
        'foto' => $this->faker->imageUrl(640, 480, 'people'),
        'password' => $faker->regexify('/([A-Za-z0-9]+(_[A-Za-z0-9]+)+)/i'),
    ];
}
```

```php

use Illuminate\Database\Eloquent\Factories\Factory;

/**
 * @extends \Illuminate\Database\Eloquent\Factories\Factory<\App\Mod
 */
0 references | 0 implementations
class GrupoCustomFactory extends Factory
{
    /**
     * Define the model's default state.
     *
     * @return array<string, mixed>
     */
    0 references | 0 overrides
    public function definition(): array
    {
        return [
            'id' => 0,
            'nombre' => 'unassigned',
            'maxParticipantes' => 100,
            'codigoClase' => 0,
        ];
    }
}

```

table.php
ord_res...
ord_res...
jobs_ta...
al_acce...
table.p...

```php
0 references | 0 overrides
public function definition(): array
{

    do {
        $primerDia = $this->faker->dateTimeBetween($startDate = '-1 month', $endDate = '+2 month');
    } while ($primerDia->format('N') >= 6); // 6 y 7 representan sábado y domingo respectivamente

    $tramos = [
        ['10:00', '11:20'],
        ['11:30', '12:50'],
        ['13:00', '14:20'],
        ['15:00', '16:20'],
        ['16:30', '17:50'],
        ['18:00', '19:20'],
        ['19:30', '20:50']
    ];

    $tramoAleatorio = $this->faker->randomElement($tramos);

    return [
        'codigoClase' => $this->faker->numberBetween(1, 10),
        'codigoGrupo' => $this->faker->numberBetween(1, 10),
        'codigoEmpleado' => $this->faker->numberBetween(1, 10),
        'diaSemana' => $this->faker->randomElement(['Lunes', 'Martes', 'Miércoles', 'Jueves', 'Viernes']
        'horaInicio' => $tramoAleatorio[0],
        'horaFin' => $tramoAleatorio[1],
        'primerDia' => $primerDia,
        'repetir' => $this->faker->boolean,
        'repeticiones' => $this->faker->numberBetween(1, 6),
    ];
```

```php
1 reference | 0 implementations
class GrupoClaseCustomSeeder extends Seeder
{
    0 references | 0 overrides
    public function run(): void
    {
        $clase = Clase::factory()->create([

                'id' => 0,
                'nombre' => 'unasigned',
                'nivel' => 'unasigned',

        ]);
        Grupo::factory()->create([
            'id' => 0,
            'nombre' => 'unassigned',
            'maxParticipantes' => 100,
            'codigoClase' => $clase->id,
        ]);
    }
}
```

```
0 references | 0 implementations
class HorarioTableSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    0 references | 0 overrides
    public function run(): void
    {
        Horario::factory(60)->create();
    }
}
```

7.3. Desarrollo del backend

Aquí muestro algunas de las funciones desarrolladas en los controladores de la app:

Por ejemplo,

```
1 reference | 0 overrides
public function mostrarPerfil($alumno)
{
    $alumno = Alumno::where('email', '=', $alumno)->first();
    //dd($alumno);
    $asistencias = $alumno->obtenerAsistencias();
    //dd($asistencias);
    //dd($alumno->codigoGrupo);
    if ($alumno->codigoGrupo === 0) {
        $horarioMañana = 0;
        $horarioTarde = 0;
    } else {
        $horarioMañana = Horario::where('codigoGrupo', '=', $alumno->codigoGrupo)
            ->where('horaInicio', '<', '14:00:00')
            ->orderBy('primerDia', 'asc')
            ->orderBy('horaInicio', 'asc')
            ->first();

        if (!$horarioMañana) {
            $horarioMañana = 0;
        }
        $horarioTarde = Horario::where('codigoGrupo', '=', $alumno->codigoGrupo)
            ->where('horaInicio', '>', '14:00:00')
            ->orderBy('primerDia', 'asc')
            ->orderBy('horaInicio', 'asc')
            ->first();

        if (!$horarioTarde) {
            $horarioTarde = 0;
        }
    }

    $reservas = Reserva::where('id_alumno', '=', $alumno->id)->get();
    $horariosReservados = [];
    foreach ($reservas as $reserva) {
        $horario = Horario::where('id', '=', $reserva->id_horario)->first();
        $horariosReservados[] = $horario;

    }
    //dd($horariosReservados);
    return view('Alumno.perfil', compact('alumno', 'asistencias', 'horarioMañana', 'horarioTarde', 'horariosReservados'));
}
```

```php
public function registrarAlumno(Request $request)
{
    if (!$request->session()->has('user')) {
        return redirect()->route('rutaAdecuada')->with('error', 'No estás autenticado.');
    }

    $user = $request->session()->get('user');

    // Calcula la fecha de hace 16 años
    $fechaHace16Años = now()->subYears(16)->format('Y-m-d');

    // Validación de la solicitud
    $validatedData = $request->validate([
        'apellidos' => 'required',
        'dni' => 'required|unique:alumnos,dni|regex:/^[0-9]{8}[A-Z]$/',
        'telefono' => 'required|regex:/^[679][0-9]{8}$/',
        'fechaNacimiento' => 'required|date|before_or_equal:' . $fechaHace16Años,
        'direccion' => 'required',
        'foto' => 'required|image',
        'password' => ['required', 'regex:/^(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]{8,}$/'],
    ], [
         'telefono.regex' => 'El número de teléfono debe comenzar con 6, 7, o 9 y tener un total de 9 dígitos.',
        'password.regex' => 'La contraseña debe tener al menos 8 caracteres, incluyendo una letra mayúscula, un número y un signo especial.',
        'dni.regex' => 'El DNI debe tener 8 dígitos seguidos de una letra mayúscula.',
        'fechaNacimiento.before_or_equal' => 'Debes tener al menos 16 años de edad.'
    ]);
    // Manejo de la foto
    $fotoPath = $request->file('foto')->store('uploads', 'public');

    // Creación del alumno
    $alumno = new Alumno([
        'nombre' => $user->name,
        'apellidos' => $validatedData['apellidos'],
        'dni' => $validatedData['dni'],
        'telefono' => $validatedData['telefono'],
        'email' => $user->email,
        'fechaNacimiento' => $validatedData['fechaNacimiento'],
        'direccion' => $validatedData['direccion'],
        'foto' => 'storage/' . $fotoPath,
        'codigoGrupo' => 0,
        'password' => $user->password,
    ]);

    $alumno->save();

    return redirect()->route('inicioAlumno', ['alumno' => $user->email])->with('success', 'Alumno registrado con éxito.');
}
```

```php
0 references | 0 overrides
public function store(Request $request)
{


    $user = $request->session()->get('user');

    // Calcula la fecha de hace 16 años
    $fechaHace16Años = now()->subYears(16)->format('Y-m-d');
    //dd($request->all());
    $campos = [
        'apellidos' => 'required',
        'dni' => 'required|unique:alumnos,dni|regex:/^[0-9]{8}[A-Z]$/',
        'telefono' => 'required|regex:/^[679][0-9]{8}$/',
        'fechaNacimiento' => 'required|date|before_or_equal:' . $fechaHace16Años,
        'direccion' => 'required',
        'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],
        'foto' => 'required|image',
        'password' => ['required', 'regex:/^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[$@$!%*?&])([A-Za-z\d$@$!%*?&]|[^ ]){8,15}$/'],


    ];

    $mensaje = [
        'email' => 'El formato del correo electrónico no es válido.',
        'required' => 'El :attribute es obligatorio',
        'foto' => 'La foto es requerida'
    ];
    echo "hola";
    $this->validate($request, $campos, $mensaje);
    $datosalumno = request()->except('_token', 'password_confirmation');
    /*
     *Comprobamos si existe un archivo en el formulario
     */

    if ($request->hasFile('foto')) {
        //dd($request->file('foto'));
        $datosalumno['foto'] = 'storage/' . $request->file('foto')->store('uploads', 'public');
    }

    Alumno::insert($datosalumno);

    // return response()->json($datosalumno);
    return redirect('alumnos')->with('mensaje', 'alumno agregado con exito');
}
```

O aquí como se realiza el pago por Stripe

```php
public function verificarDatos(Request $request)
{
    \Stripe\Stripe::setApiKey(env('STRIPE_SECRET'));
    //dd($request->all());
    try {
        $producto = $request->input('producto');
        //dd($producto);
        $validacionDatos = $request->validate([
            'nombre' => 'required|max:255',
            'precio' => 'required|numeric',
            'stripeToken' => 'required'
        ]);

    } catch (ValidationException $e) {
        return view('facturaciones.checkout')
            ->withErrors($e->validator)
            ->with([
                'producto' => $producto,
                'stripeKey' => env('STRIPE_KEY'),
            ]);
    }
    $precio = $request->input('precio') * 100;
    try {
        $charge = \Stripe\Charge::create([
            'amount' => $precio,
            'currency' => 'eur',
            'description' => 'Pago por ' . $request->input('nombre'),
            'source' => $request->input('stripeToken'),
        ]);
        //dd($charge->status);
        if ($charge->status == 'succeeded') {

            $user = Auth::user();
            $user->verificado = 1;
            $user->save();
            do {
                $fecha = date('Ymd');
                $numeroAleatorio = rand(1000, 9999);
                $numeroReferencia = $fecha . $numeroAleatorio;

                $existe = Factura::where('referencia', $numeroReferencia)->exists();
            } while ($existe);
            Factura::insert([
                'referencia' => $numeroReferencia
```

```php
$precio = $request->input('precio') * 100;
try {
    $charge = \Stripe\Charge::create([
        'amount' => $precio, // El monto aquí es un ejemplo, convierte tu precio a cent
        'currency' => 'eur', // Cambia a tu moneda según sea necesario
        'description' => 'Pago por ' . $request->input('nombre'),
        'source' => $request->input('stripeToken'), // El token de Stripe
    ]);
    //dd($charge->status);
    if ($charge->status == 'succeeded') {
        // Redirigir a la página de éxito
        $user = Auth::user();
        $user->verificado = 1;
        $user->save();
        do {
            $fecha = date('Ymd'); // Fecha actual en formato año-mes-día
            $numeroAleatorio = rand(1000, 9999); // Número aleatorio entre 1000 y 9999
            $numeroReferencia = $fecha . $numeroAleatorio;

            $existe = Factura::where('referencia', $numeroReferencia)->exists();
        } while ($existe);
        Factura::insert([
            'referencia' => $numeroReferencia,
            'id_usuario' => $user->id,
            'fecha' => now(),
            'importe' => $precio,
        ]);
        $datosFactura = [
            'referencia' => $numeroReferencia,
            'id_usuario' => $user->id,
            'fecha' => now(),
            'importe' => $precio,
            'producto' => $producto,
        ];
        return $this->descargarFactura($datosFactura);
```

También aplico el uso de modelos

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

40 references | 0 implementations
class Alumno extends Model
{
    use HasFactory;
    0 references
    public $timestamps = true;
    0 references | 0 overrides
    public function asistencias()
    {
        return $this->hasMany(Asistencia::class);
    }

    0 references | 0 overrides
    public function obtenerAsistencias()
    {
        //dd($this);
        $asistencia = Asistencia::where('alumno_id', '=', $this->id)->get();
        //dd($asistencia);
        $numeroAsistencias = $asistencia->count();

        return ['numeroAsistencias' => $numeroAsistencias, 'asistencias' => $asistencia];
    }
}
```

```php
19 references | 0 implementations
class Horario extends Model
{
    use HasFactory;

    0 references
    protected $fillable = ['codigoClase', 'codigoEmpleado', 'codigoGrupo', 'diaSemana', 'horaInicio', 'horaFin', 'primerDia', 'repetir', 'repeticiones'];

    0 references | 0 overrides
    public function obtenerCodigoClase()
    {
        return $this->belongsTo(Clase::class, 'nombre');
    }

    0 references | 0 overrides
    public function obtenerCodigoEmpleado()
    {
        return $this->belongsTo(Empleado::class, 'nombre');
    }
    0 references | 0 overrides
    public function obtenerNombreClase()
    {
        echo "hola";
    }

    0 references | 0 overrides
    public function clase()
    {
        return $this->belongsTo('App\Models\Clase', 'codigoClase', 'id');
    }

}
```

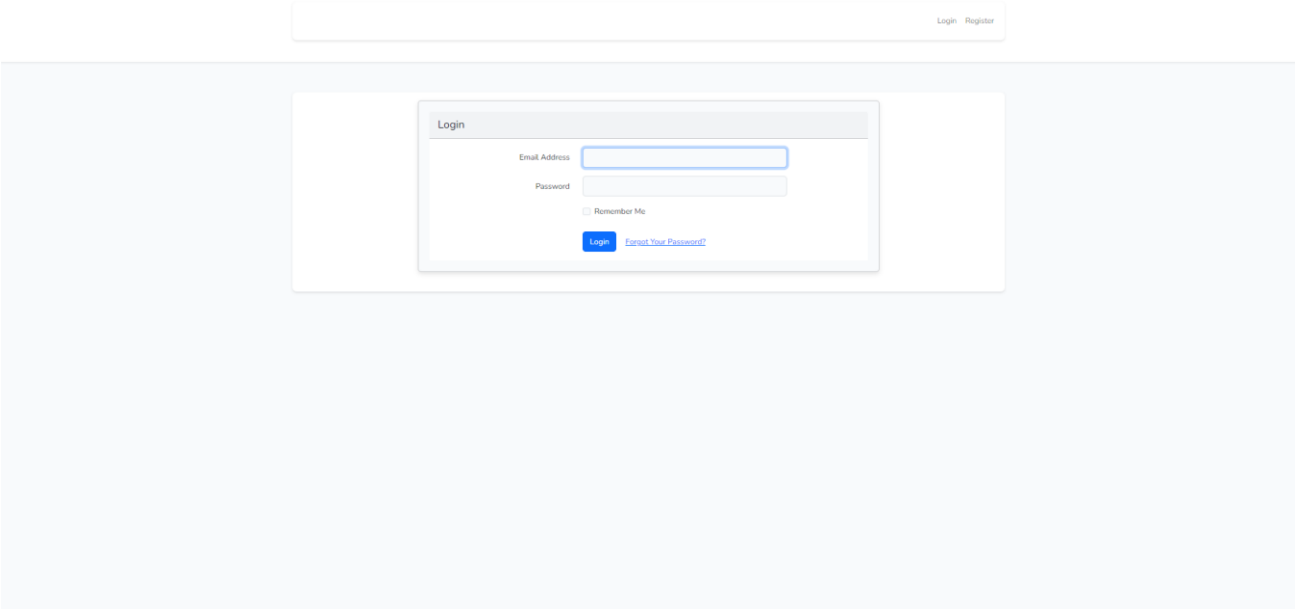
Así es la autenticación

```php
protected function validator(array $data): ValidatorContract
{
    $opcionCodigo = "";
    $code = $data['code'];
    $invitation = InvitationCode::where('code', $code)
        ->where('used', 0)
        ->first();
    if (isset($data['codigoUse'])) {
        $opcionCodigo = $data['codigoUse'];
        if (!$invitation) {
            throw ValidationException::withMessages([
                'code' => ['El código de invitación no es válido.'],
            ]);
        }
        $rules = [
            'code' => ['required', 'string', 'exists:invitation_codes,code,used,0'],
            'name' => ['required', 'string', 'max:255'],
            'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],
            'password' => ['required', 'string', 'min:8', 'confirmed'],
            'role' => ['required', 'string', new MatchInvitationRole($invitation->role)]
        ];
        $messages = [
            'required' => 'El campo :attribute es obligatorio.',
            'string' => 'El campo :attribute debe ser una cadena de texto.',
            'exists' => 'El código no es válido.',
            'max' => 'El campo :attribute no puede tener más de :max caracteres.',
            'email' => 'El formato del correo electrónico no es válido.',
            'unique' => 'El correo electrónico ya está registrado.',
            'confirmed' => 'La confirmación de la contraseña no coincide.',
        ];

        $validator = Validator::make($data, $rules, $messages);

        return $validator;
    } else {
        //dd($invitation);
        $rules = [
            'name' => ['required', 'string', 'max:255'],
            'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],
            'password' => ['required', 'string', 'min:8', 'confirmed'],
            'role' => ['required', 'string', 'in:admin,alumno,empleado'],

        ];
        $messages = [
            'required' => 'El campo :attribute es obligatorio.',
            'string' => 'El campo :attribute debe ser una cadena de texto.',
            'max' => 'El campo :attribute no puede tener más de :max caracteres.',
            'email' => 'El formato del correo electrónico no es válido.',
            'unique' => 'El correo electrónico ya está registrado.',
            'confirmed' => 'La confirmación de la contraseña no coincide.',
        ];

        return Validator::make($data, $rules, $messages);
```

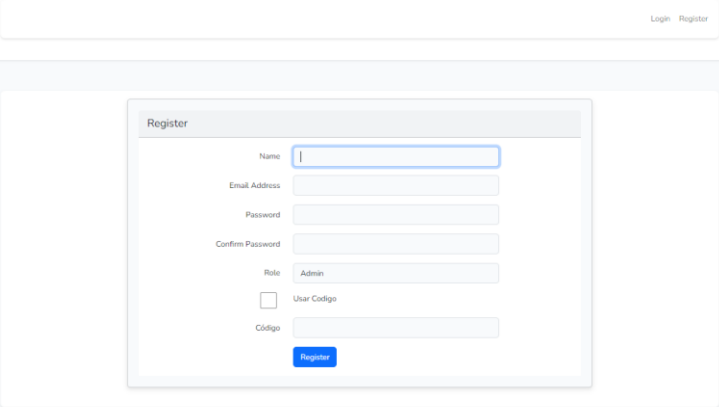
```
0 references | 0 overrides
protected function registered(Request $request, $user)
{


    $clases = Clase::all();
    $existeUsuario = $this->comprobarExistencia($user);
    if ($existeUsuario && $user->verificado === 'true') {
        if ($user->role === 'admin') {
            $redirectPath = '/admin';
        } elseif ($user->role === 'alumno') {
            $redirectPath = '/alumno';
        } elseif ($user->role === 'empleado') {
            $redirectPath = '/empleado';
        } else {
            $redirectPath = $this->redirectTo;
        }
    } else {
        if ($user->role === 'admin') {
            $redirectPath = '/admin';
        } elseif ($user->role === 'alumno') {
            $redirectPath = route('registroAlumno');
        } else {
            $redirectPath = route('mostrarRegistro');
        }

    }


    $request->session()->flash('registro_exitoso', 'Registro exitoso. Bienvenido.');
    $request->session()->put('user', $user);

    return redirect($redirectPath)->with('clases', $clases);
}
```

7.4. Desarrollo del front-end

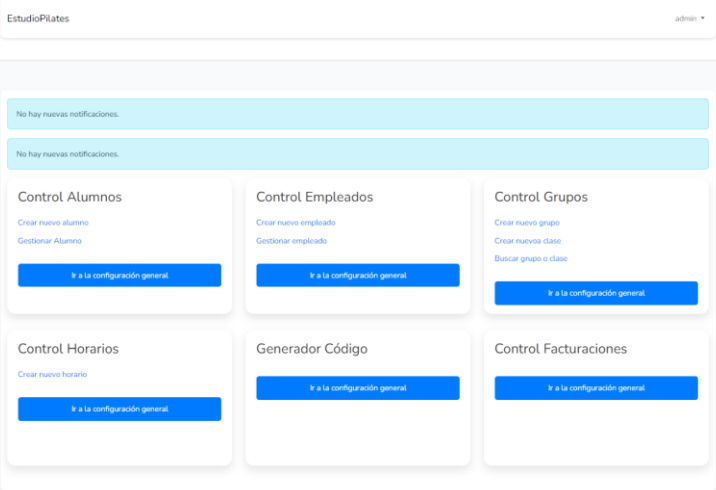
7.4.1. LOGIN



7.4.2. REGISTER



7.4.3. INICIO DE SESION DEL ADMIN



7.4.4. CONTROL ALUMNOS

EstudioPilates

admin

Crear nuevo alumno

Foto	ID	Nombre	Apellidos	DNI	Telefono	Email	fechaNacimiento	Direccion	codigoGrupo	Opciones
	11	ruben	Torres	71778865M	342	infotrabajo97@gmail.com	1997-08-18	trrr	0	
	13	ruben	torres	717788	333333	pabloterror99@gmail.com	1997-08-18	turonfafdsdgfsgfg	0	
	14	Mr. Brad Gutmann	Kuhlman	58150921A	(669) 514-5683	schiller.marcelina@example.org	1996-07-12	91238 Shaun Freeway Daijachester, CO 73929-1667	1	
	15	Dr. Tessie Labadie MD	Feeney	35319814A	331.916.8486	markus.maggio@example.net	1989-07-05	6746 Melyssa Motorway Suite 648 Rogeliotown, DC 77418-9140	1	
	16	Sammie Gerlach III	Buckridge	05777545A	562-451-5668	clement40@example.com	1976-10-25	52802 Smith Wells Aleenland, NV 90832-8272	1	

‹ 1 2 3 4 5 ›

Crear alumno

Nombre:

Apellidos:

Seleccionar archivo N...

DNI:

Telefono:

Crear datos alumno

Regresar

Email:

Contraseña:

Confirmar contraseña:

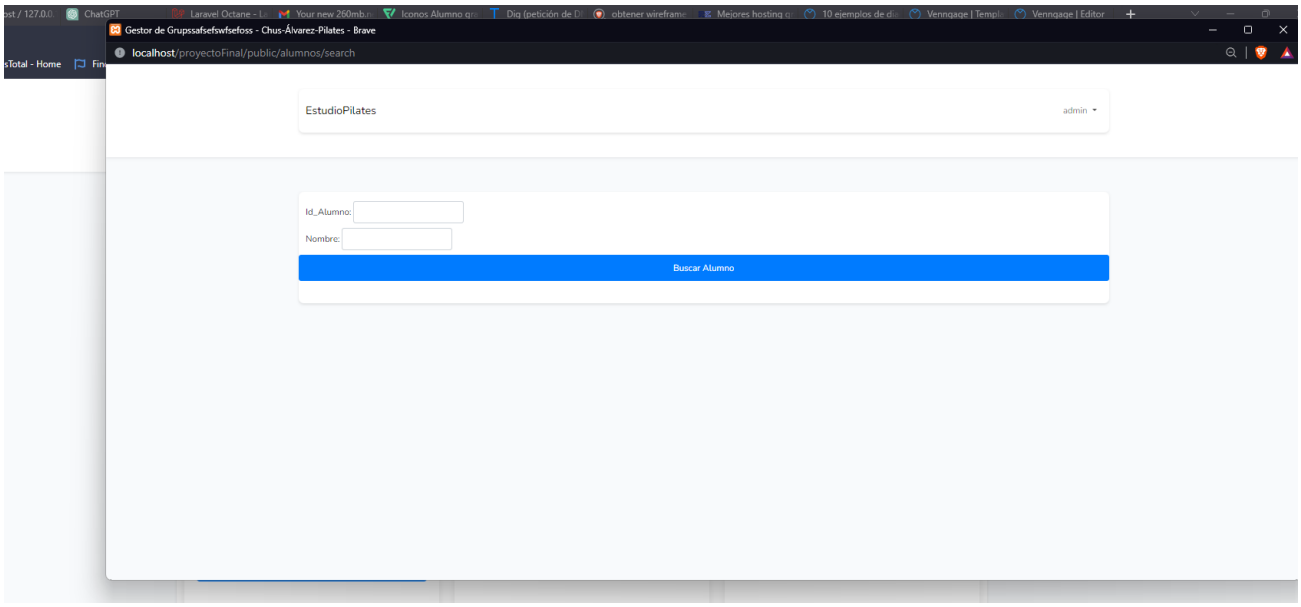
Fecha Nacimeinto:

dd/mm/aaaa

Direccion:

Grupo:

0 unassigned


Gestor de Grupssafsefswfsefoss - Chus-Álvarez-Pilates - Brave
localhost/proyectoFinal/public/alumnos/search
EstudioPilates
admin
Id_Alumno:
Nombre:
Buscar Alumno

7.5. EMPLEADOS GESTION

EstudioPilates admin

Crear nuevo empleado

Foto	ID		Apellidos		Telefono		fechaNacimiento		codigoGrupo	Opciones
Imagen del usuario	1	Clyde Bashirian	Gutkowski	07953298A	(716) 413-1208	kolby75@example.org	2008-02-16	76927 Schulist Parks New Pablo, PA 77797-1521	1	
Imagen del usuario	2	Rosina Bartell	McClure	09241767A	1-551-618-5338	jasper32@example.com	1970-03-30	2351 Emelia Corners Lake Elsefurt, MI 54432-0236	1	
Imagen del usuario	3	Mallie Wintheiser IV	Reichel	55202405A	351-489-7709	carroll.schiller@example.org	2002-03-01	214 Douglas Pass Suite 836 South Elsa, DC 94592-1772	1	
Imagen del usuario	4	Colby Hoeger	Waters	08433696A	+1-607-884-8821	cicero65@example.net	2023-05-10	1097 Santiago Coves Apt. 765 South Fletaborough, NV 81974	1	
Imagen del usuario	5	Prof. Henry Spencer PhD	Zieme	94930906A	1-484-475-0570	raphael.friesen@example.org	1985-07-15	87921 Aufderhar Creek Rosalynborough, ME 03157-9924	1	

‹ 1 2 3 ›

Crear empleado

Nombre:

Apellidos:

Seleccionar archivo N...

DNI:

Telefono:

Crear datos empleado

Regresar

Email:

Contraseña:

Confirmar contraseña:

Fecha Nacimeinto:

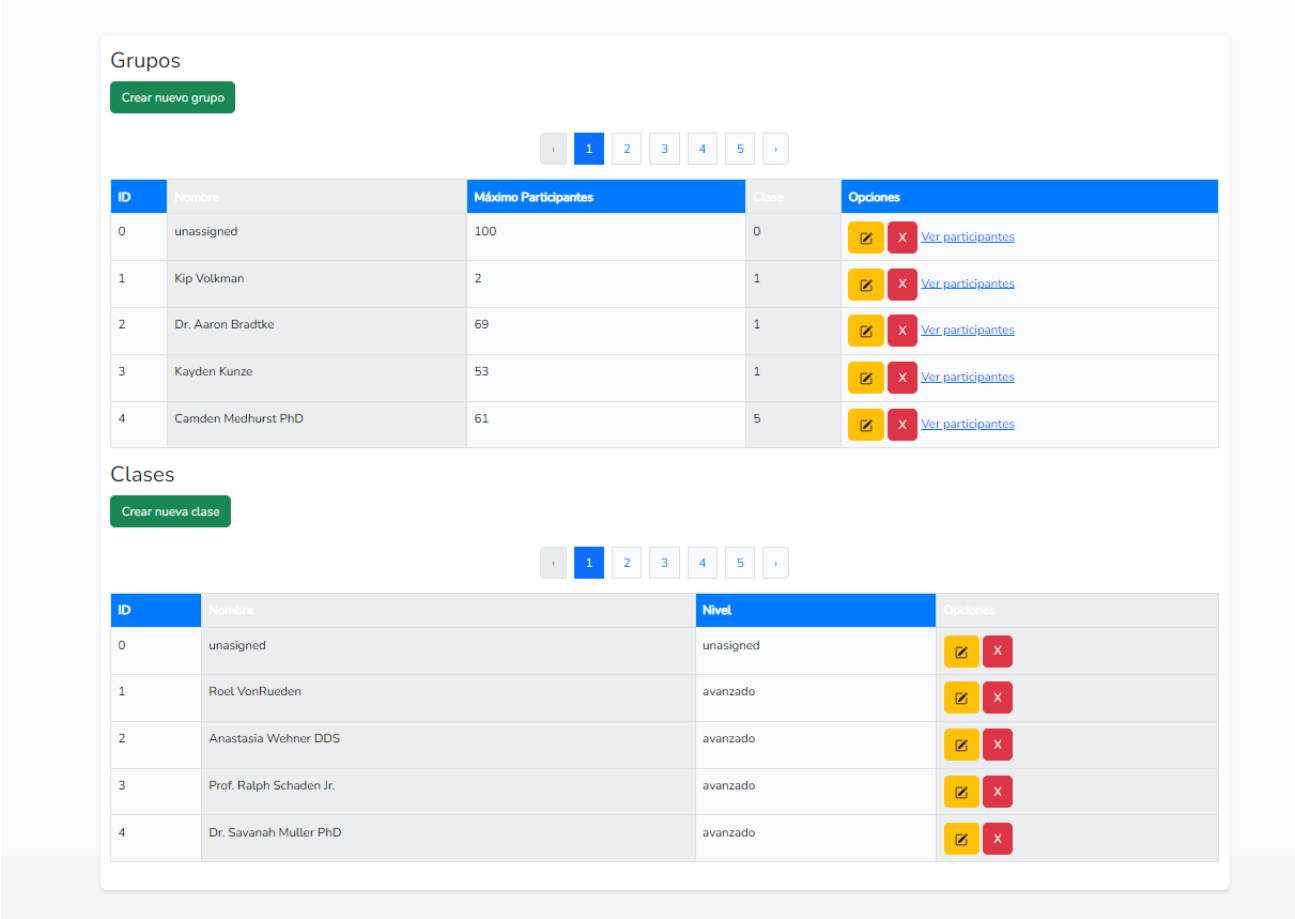
dd/mm/aaaa

Direccion:

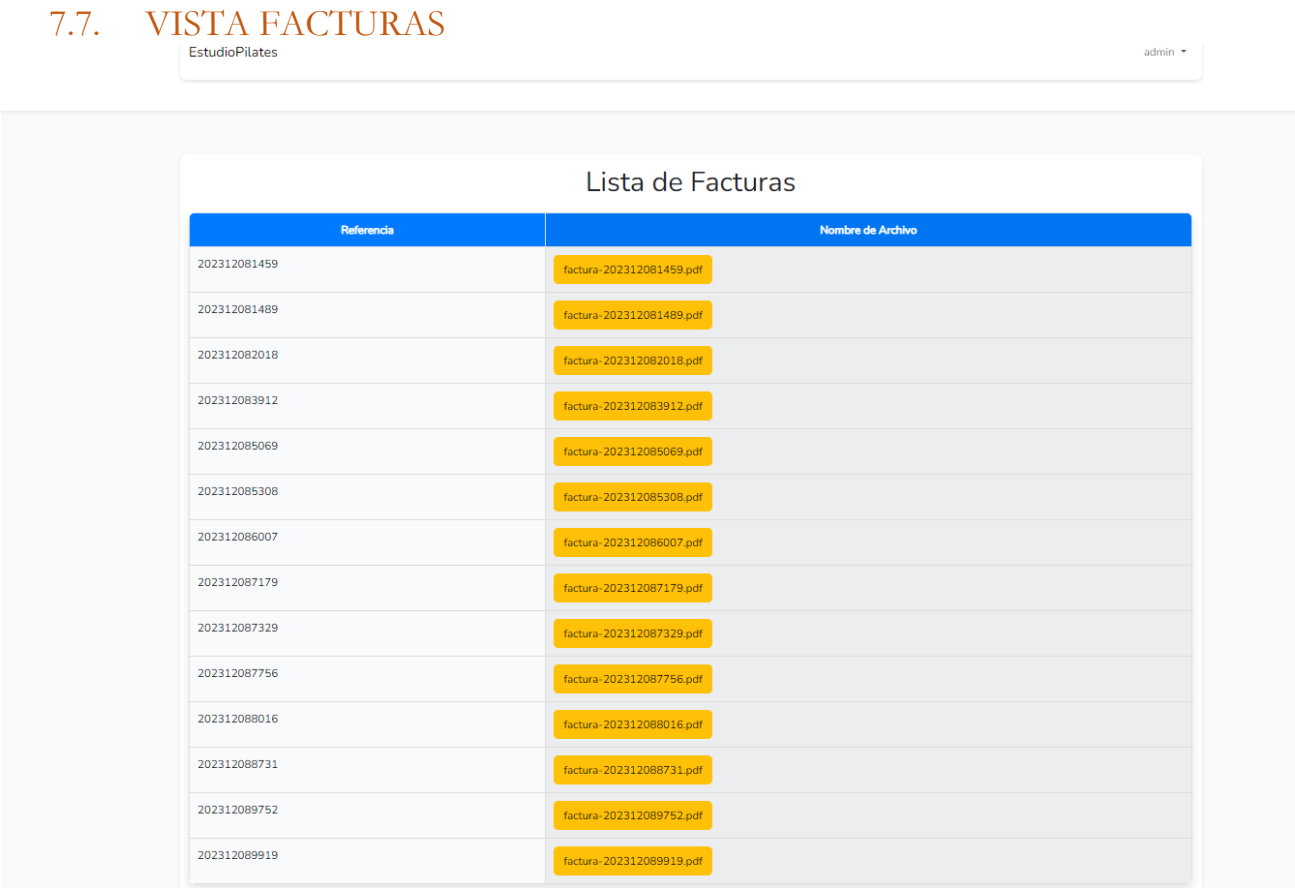
Clase:

0 unasigned

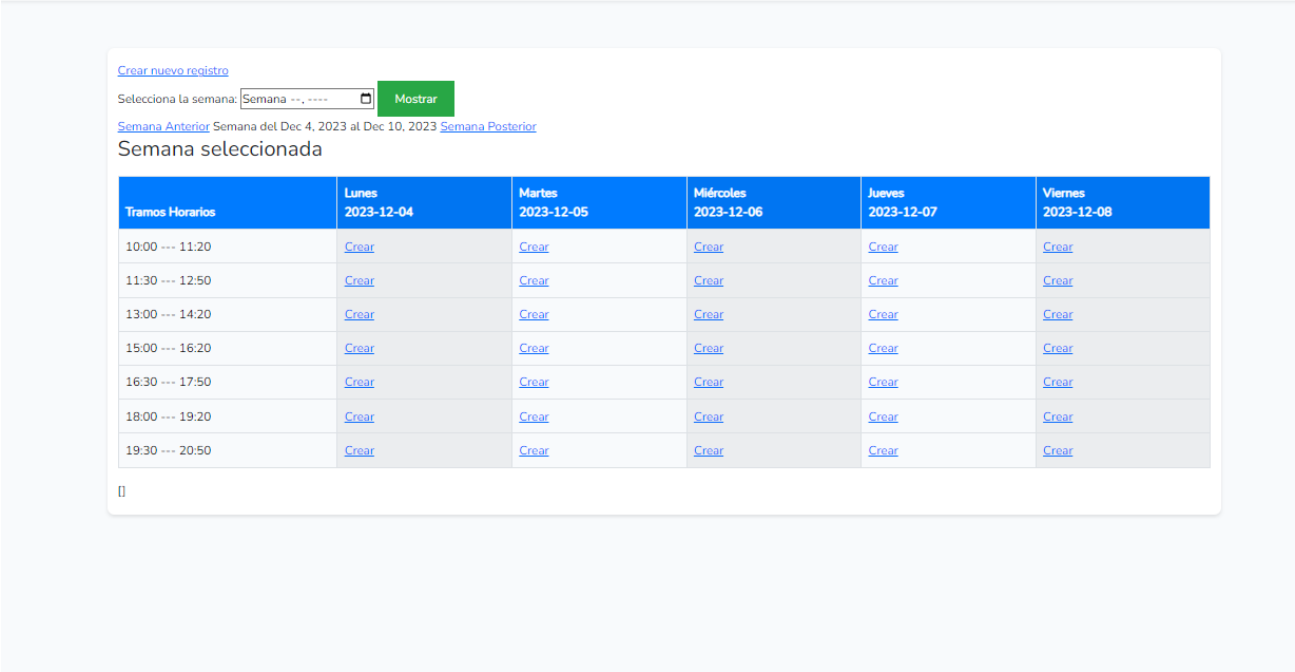
7.6. GESTION GRUPOS Y CLASES



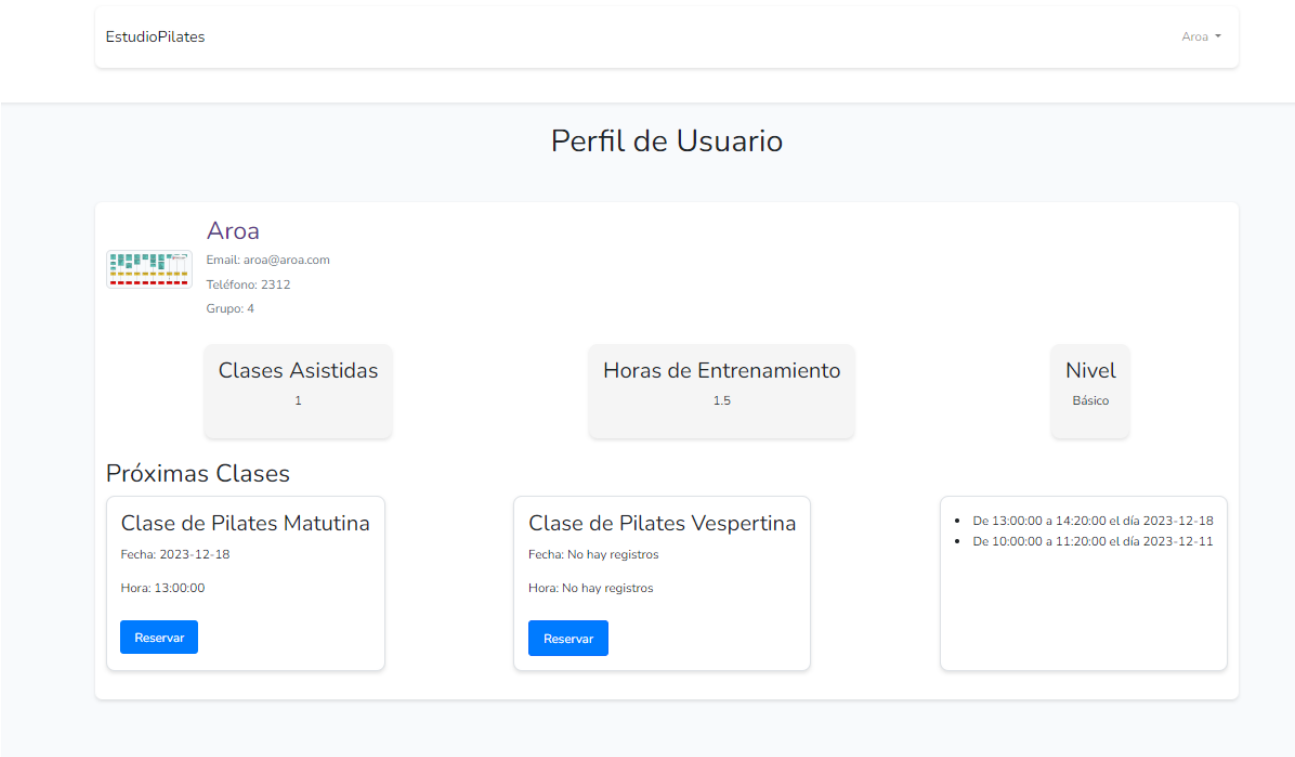
7.7. VISTA FACTURAS



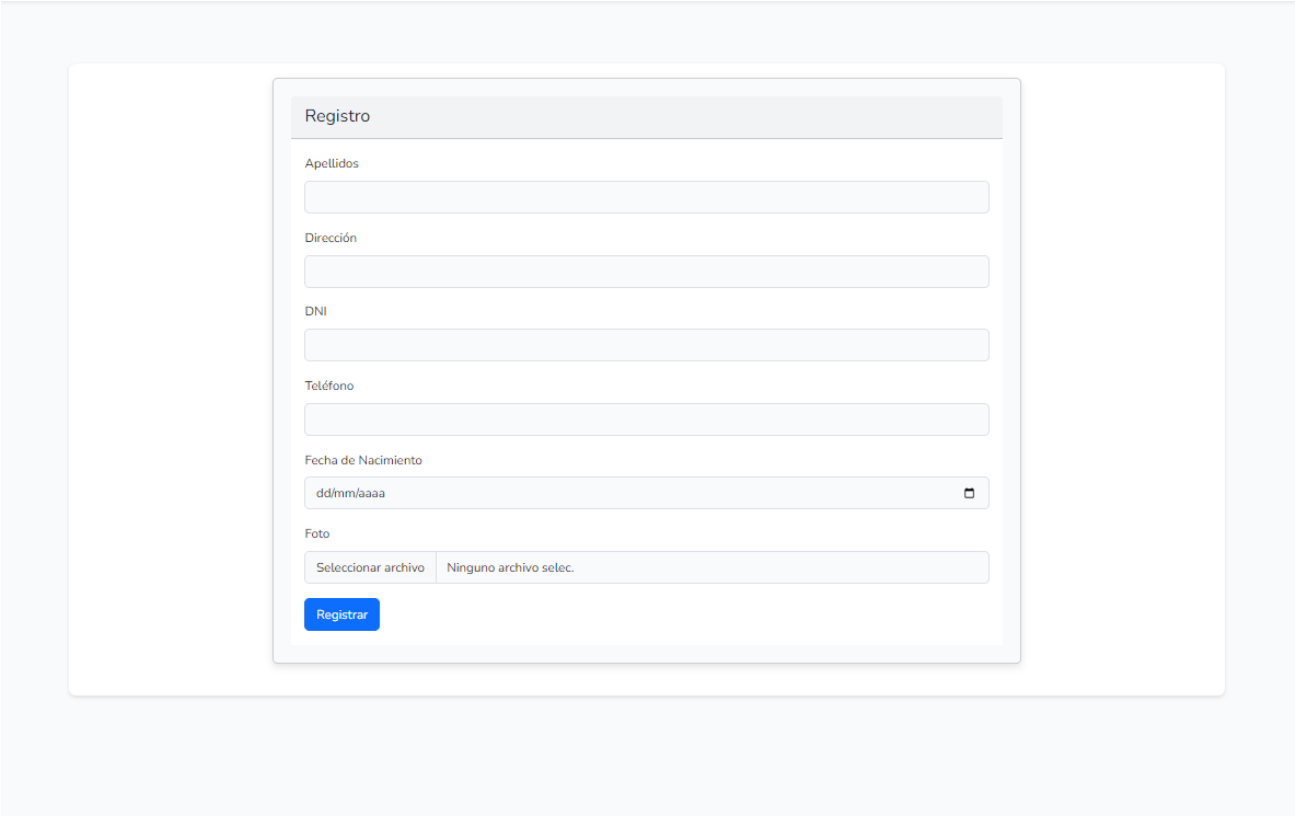
7.8. HORARIOS



7.9. INICIO SESION DE ALUMNO



7.10. REGISTRO DE EMPLEADO O ALMUNO



7.11. PERFIL DE EMPLEADO

Detalles del Empleado

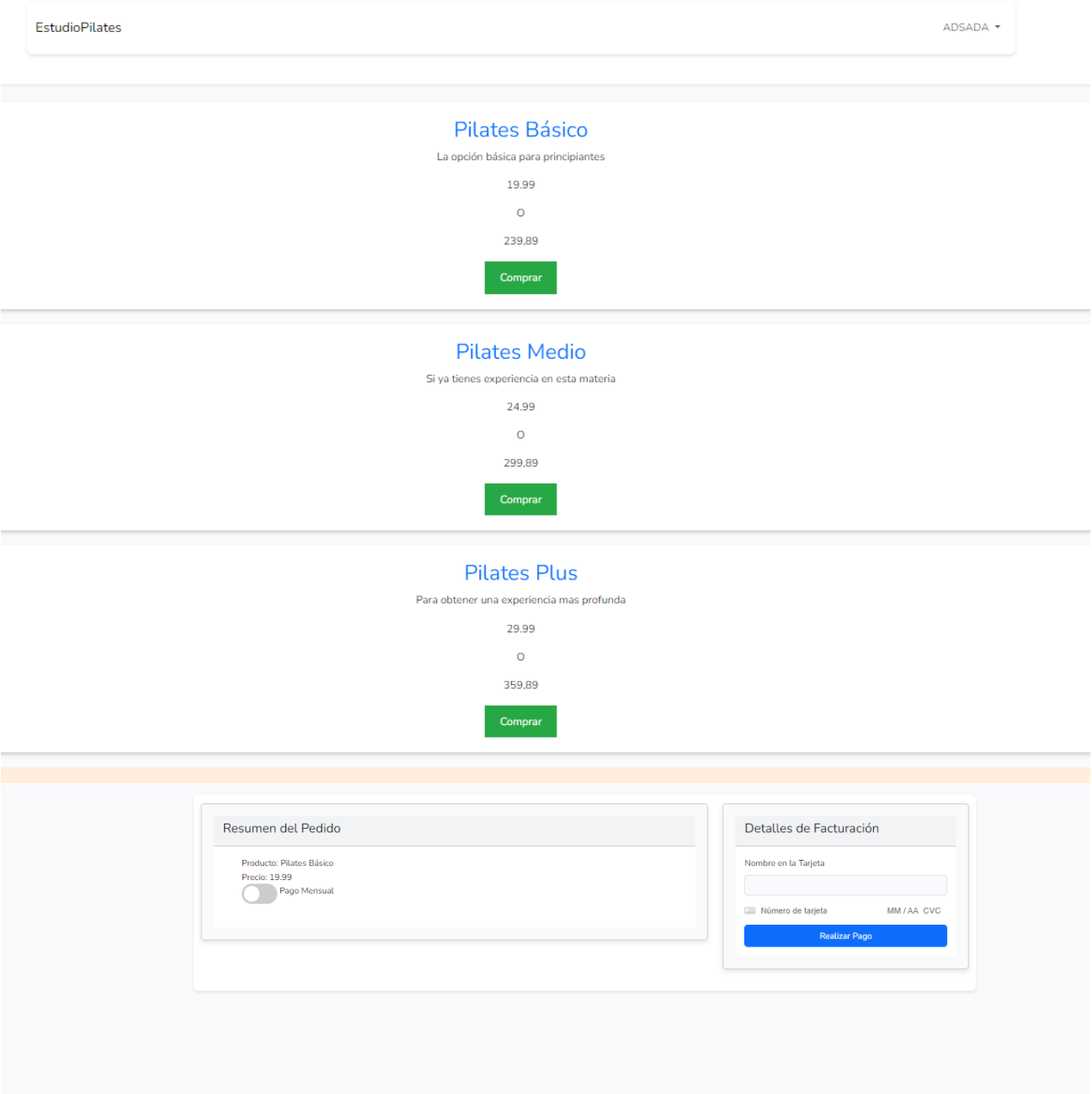
RUBEN AGS

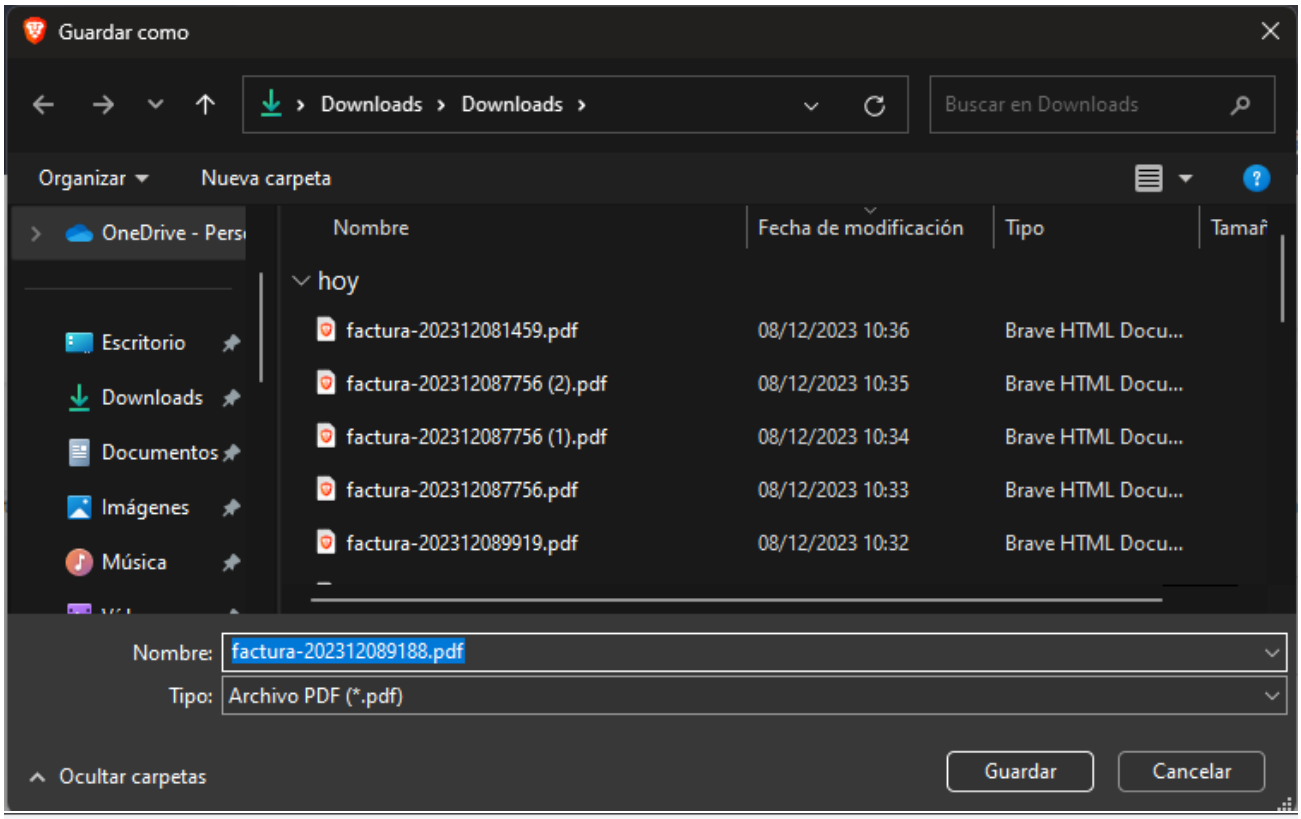
DNI: 72574475M
Teléfono: 675466363
Email: RUB@RUB.COM
Fecha de Nacimiento: 1997-11-27
Dirección: GASF
Código de Clase: 0

Lista de Alumnos

Nombre	Edad	Email
ruben	Torres	infotrabajo97@gmail.com
ruben	torres	pabloterror99@gmail.com
fvfd	ar32243r	bfd@dg.efd
ddd	ewrerw	ejemplo@ejemplo.com
admin	qrqe	easdf@sdf.com
pru1	pru	pru1@pru.com
Aroa	Diaz	aroa@aroaa.com
ruben	a	ruben@ruben.com
admin	torres	rffdgsd@hgddg.gc

7.12. FACTURACION ALUMNO





EstudioPilates
pru1
Logout

Nombre de la Empresa: Estudio Pilates	Dirección: Sin direccion
Teléfono: 643 51 09 09	Email: infotrabajo97@gmail.com

Producto	Precio
Producto: Pilates Básico	Total: 19.99

8. EVALUACION

8.1. Introducción

En esta sección detallo brevemente las validaciones por las que ha pasado nuestra página web.

8.2. Validación de enlaces

Esta aplicacion web sólo utiliza enlaces propios de este proyecto, así que todos los enlaces son seguros y funcionan correctamente.

8.3. Validación de la resolución

EstudioPilates pru1

Perfil de Usuario

pru1

Email: pru3@pru.com

Teléfono: 2343432

Grupo: 0

Clases Asistidas	Horas de Entrenamiento	Nivel
0	0	Básico

Próximas Clases

Clase de Pilates Matutina

Fecha: No hay registros

Hora: No hay registros

Reservar

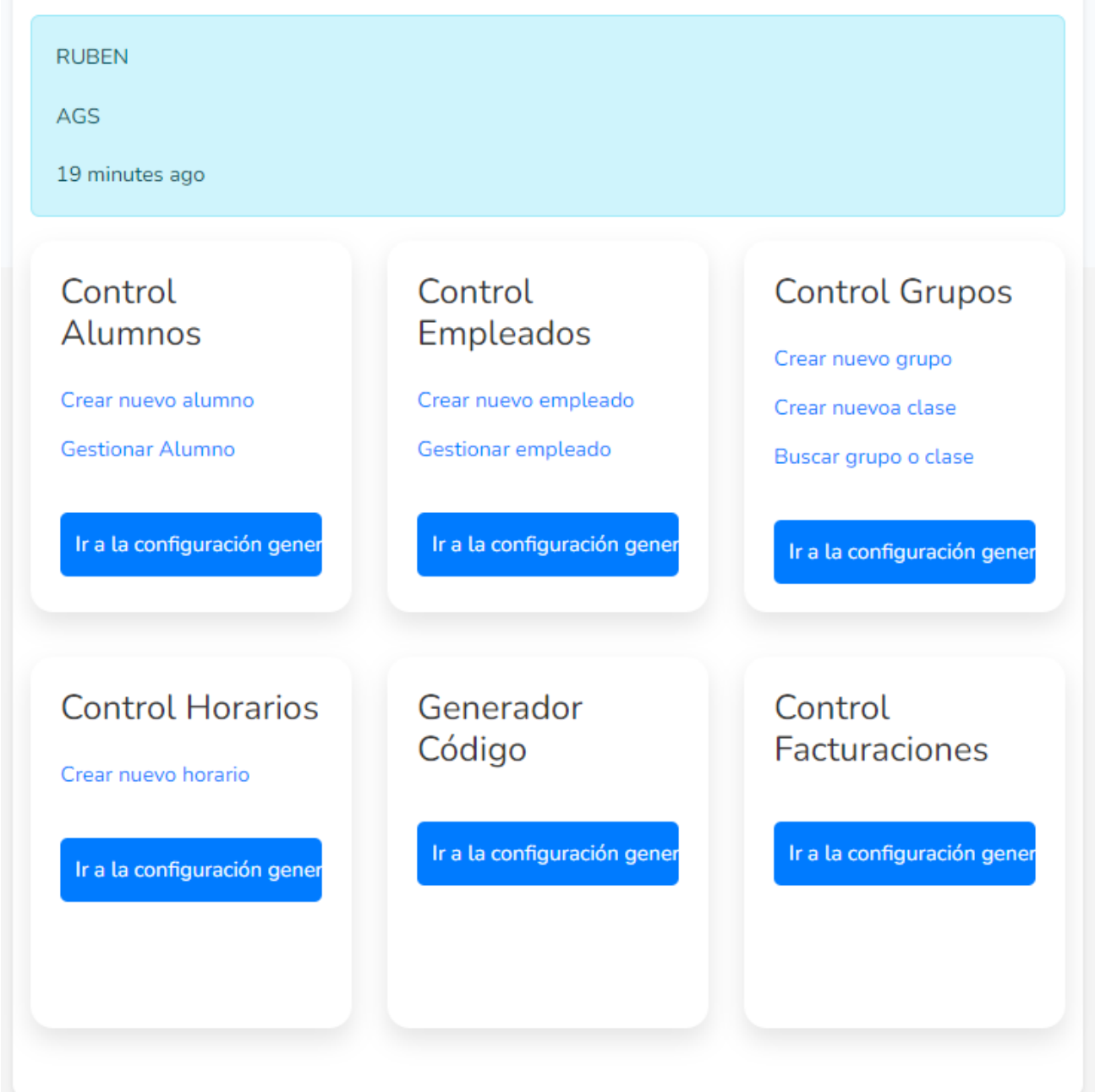
Clase de Pilates Vespertina

Fecha: No hay registros

Hora: No hay registros

Reservar

© 2023 EstudioPilates



9. COONCLUSION

9.1. Valoración personal

Para mi el desarrollar este proyecto es un objetivo real el cual me ha costado mucho tiempo y el cual he invertido muchas horas de investigación tanto en el framework Laravel como en las mismas implementaciones del código con PHP.

El tema del proyecto es bastante complejo, pero eso me gusta ya que aumenta mis capacidades de aprendizaje al enfrentarme a mayores retos.

Igualmente, me siento orgulloso de haber conseguido construir este proyecto.

9.2. Posibles ampliaciones

Hay que ver que se puede hacer con pequeñas cosas como por ejemplo que el alumno tenga alguna función mas como comunicarse con el profesor

Mejorar el control de facturas reservas y pagos para el correcto desarrollo en un futuro con la arquitectura del proyecto

9.3. BIBLIOGRAFIA

Documentación oficial de laravel: https://laravel.com/docs/10.x/readme

Laravel 10 Cheat Sheet / Summary https://itf-laravel-10.netlify.app/config/cheatsheet

Stripe docs https://stripe.com/docs

9.4. IMPLEMENTACION

Accedemos al enlace de git y descargamos el zip, descomprimir en xampp/htdocs con el nombre **proyectoFinal**

Abrimos visual estudio con el proyecto y abrimos la terminal

Escribimos el siguiente comando npm install

A continuación, escribimos composer install y ejecutamos

Lo siguiente que tenemos que hacer es las migraciones

Ejecutamos php artisan migrate

Luego introduciremos datos en la bd

Ejecutamos php artisan db:seed

Si queda alguna sin ejecutar se puede prescindir de ella, pero mejor hacer con todas

Php artisan db:seed –class=nombreclase

Con esto debería de funcionar todo