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

2. INTRODUCCIÓN

2.1. 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…

2.2. 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.

2.3. 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.

2.4. 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

3. ESPECIFICACION DE REQUISITOS

3.1. Introducción

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

3.2. 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:

3.2.1. Requisitos funcionales

3.2.1.1. 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.

3.2.1.2. *Usuario 'admin o empleado'*

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.

3.2.1.3. *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

3.2.1.4. *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.

3.2.2. Requerimientos de interfaces Externas

3.2.2.1. *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.

3.2.2.2. *Interfaces hardware*

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

3.2.2.3. Interfaces Software

La aplicación funcionará en cualquier equipo que tenga navegador web y

conexión a Internet.

3.2.2.4. 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.

3.2.3. 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.

3.2.4. Obligaciones del Diseño.

3.2.4.1. 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.

3.2.4.2. 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.

3.2.5. Atributos de la Aplicación Usando Laravel

3.2.5.1. 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.

3.2.5.2. 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.

3.2.5.3. 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.

3.2.5.4. Otros Requerimientos

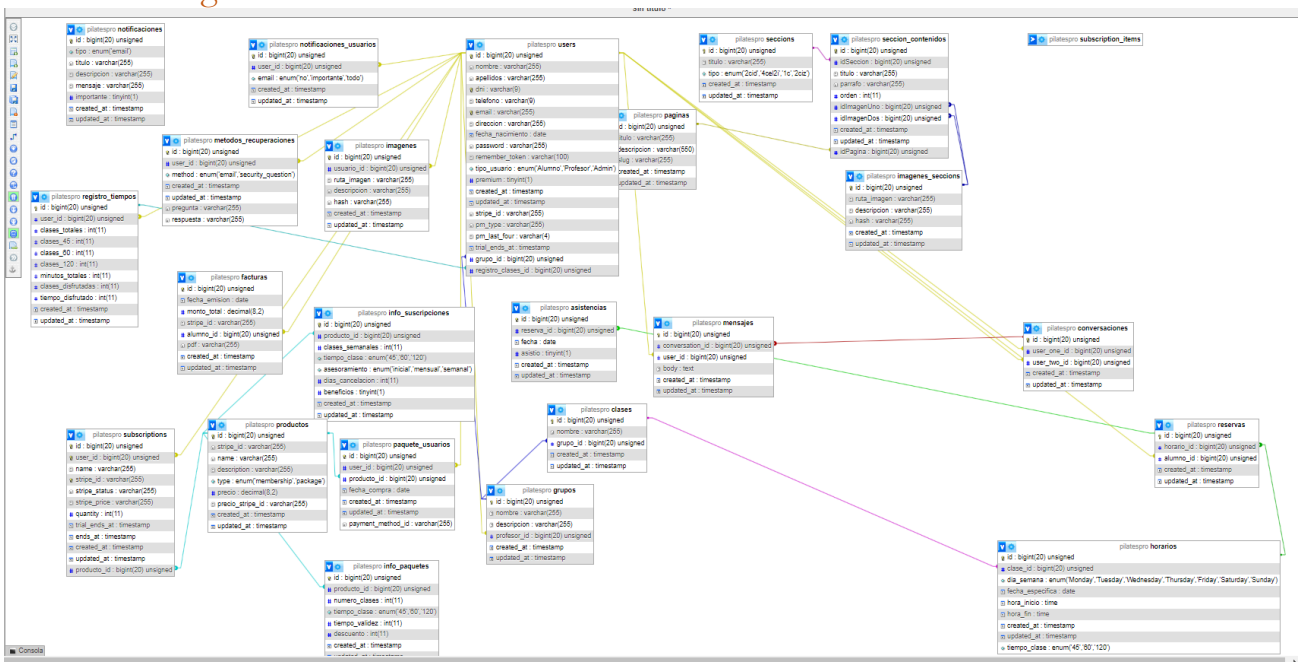
La aplicación utiliza una base de datos MySQL (phpMyAdmin) para almacenar toda la información relevante. Laravel facilita la realización de consultas a la base de datos de manera eficiente y segura a través de PHP, aprovechando su ORM Eloquent para una mejor gestión de los datos.

4. ANALISIS

4.1. 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.

4.2. 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

4.3. Diagramas de uso

5. DISEÑO

5.1. 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.

5.2. 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.

5.3. 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.

5.4. 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.

6. IMPLEMENTACION

6.1. 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

6.2. Establecimiento de la base de datos

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

```
APP_NAME=PilatesPro
APP_ENV=local
APP_KEY=base64:fs0B4CDIohgpA11fjMkFAF2n4ycpE0mAhjZDABu4KXA=
APP_DEBUG=true
APP_URL=http://localhost/PilatesPro/public

LOG_CHANNEL=stack
LOG_DEPRECATIONS_CHANNEL=null
LOG_LEVEL=debug

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=PilatesPro
DB_USERNAME=root
DB_PASSWORD=

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

MAIL_MAILER=smtp
MAIL_HOST=smtp.gmail.com
MAIL_PORT=587
MAIL_USERNAME=pilates24pro@gmail.com
MAIL_PASSWORD=mmzaiadrqujwlnjd
MAIL_ENCRYPTION=tls
MAIL_FROM_ADDRESS="pilates24pro@gmail.com"
MAIL_FROM_NAME="PilatesPro"

AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=
AWS_DEFAULT_REGION=us-east-1
AWS_BUCKET=
AWS_USE_PATH_STYLE_ENDPOINT=false

PUSHER_APP_ID=
PUSHER_APP_KEY=
PUSHER_APP_SECRET=
PUSHER_HOST=
PUSHER_PORT=443
PUSHER_SCHEME=https
PUSHER_APP_CLUSTER=mt1

VITE_APP_NAME="${APP_NAME}"
VITE_PUSHER_APP_KEY="${PUSHER_APP_KEY}"
VITE_PUSHER_HOST="${PUSHER_HOST}"
VITE_PUSHER_PORT="${PUSHER_PORT}"
VITE_PUSHER_SCHEME="${PUSHER_SCHEME}"
VITE_PUSHER_APP_CLUSTER="${PUSHER_APP_CLUSTER}"


STRIPE_KEY=pk_test_51PLsOh03z1nQyNdHhv1SQOKrRROQLphbNkI6rZZvohCH6aZNSrLeb97q5qZAVDTr9JIxE5YO2doT0ilehn313Xn300JcrkNHxM
STRIPE_SECRET=sk_test_51PLsOh03z1nQyNdHYcNA5A3d8twolAdpIGq7dlLIxsQp7v02fMHyj16Dv5qYOXd6EDE5dzbej5Fj0Zv4oY5dH06f00Srua1Dcj


TWILIO_SID=ACdf5371f518ad502608e0f15925ae585b
TWILIO_AUTH_TOKEN=751f25766f6c71e92e3494d653e19383
TWILIO_PHONE_NUMBER=+17064938419
```

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

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.

7. EVALUACION

7.1. Introducción

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

7.2. Validación de enlaces

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

7.3. Validación de la resolución

EstudioPilates pru1

Perfil de Usuario

pru1

Email: pru3@pru.com

Teléfono: 2343432

Grupo: 0

Clases Asistidas

0

Horas de Entrenamiento

0

Nivel

Básico

Próximas Clases

Clase de Pilates Matutina

Fecha: No hay registros

Hora: No hay registros

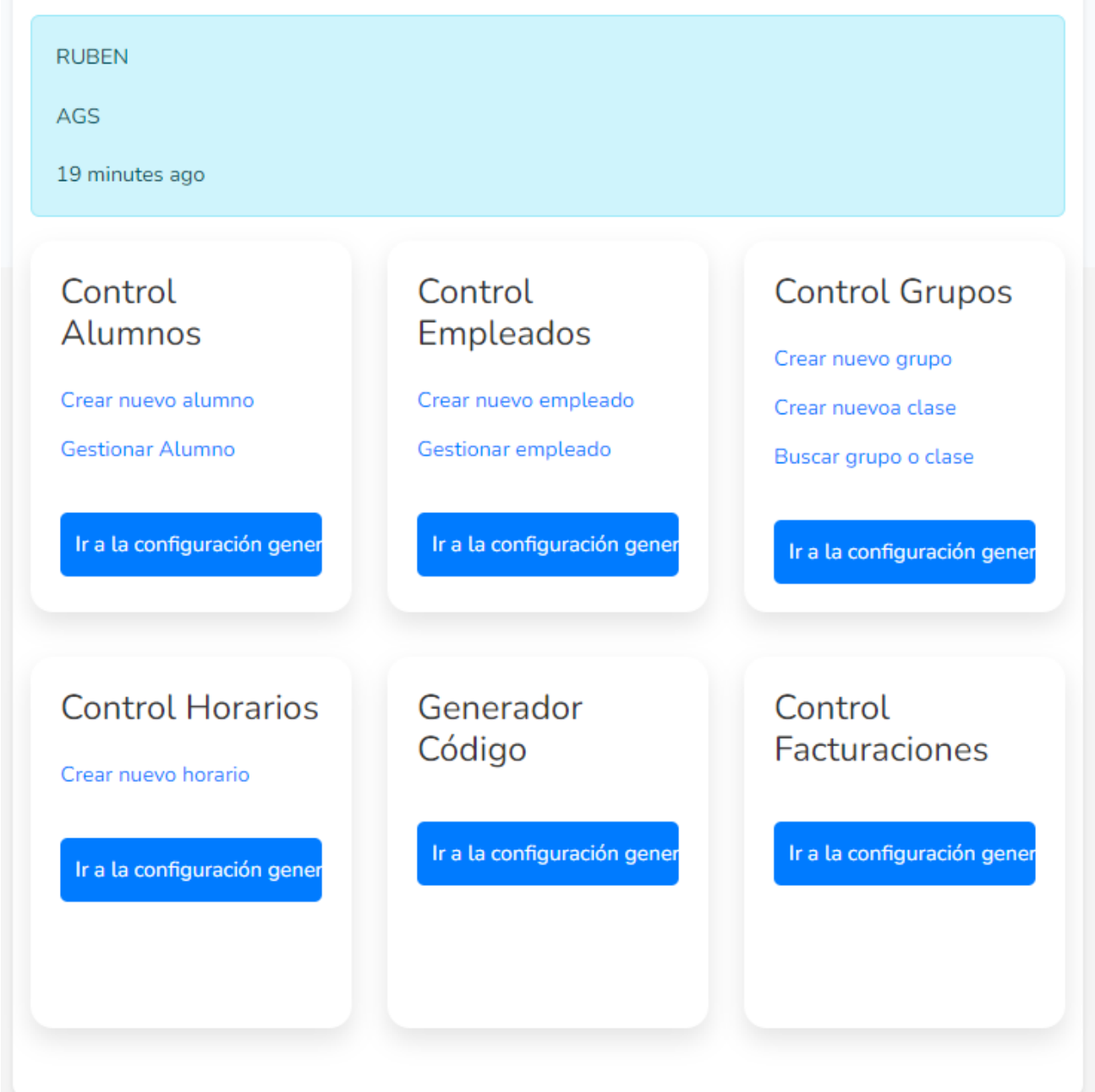
Reservar

Clase de Pilates Vespertina

Fecha: No hay registros

Hora: No hay registros

Reservar



8. COONCLUSION

8.1. 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.

8.2. 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

8.3. 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

8.4. IMPLEMENTACION

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

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