cpifp Los Enlaces

18/05/2024

GYM TRAINER

2°DAW 2023/2024
OSCAR ARIÑO
ALEJANDRO MARÍN
CPIFP LOS ENLACES

1. Descripción del Proyecto

1.1. Contexto del Proyecto

1.1.1. Ámbito y entorno

Consiste en la realización de un servicio deportivo ofrecido directamente por la empresa GymTrainer S.L. que ofrece gimnasios para todo tipo de personas por unos precios razonables. Nos vamos a acercar a todos los usuarios potenciales, mediante la inserción de gimnasios y actividades accesibles.

Tras varios meses de preparación, la empresa se ha dado a conocer y desde el equipo vamos a intentar dar un paso adelante en la mejora de la salud de todas las personas, sin importar edades o ritmos de vida, adaptándonos a cada grupo que será atendido por todos nuestros profesionales. La aplicación web les permitirá consultar avisos del gimnasio, así como ofertas de nuevas actividades o precios. El apartado de marketing va enfocado principalmente a que los usuarios conozcan sus compras y vea las fechas en las que están entrenando, para darles un empuje a sus vidas.

1.1.2. Análisis de la realidad

Nosotros somos una empresa nueva, que busca la innovación y la adaptación a otros mercados competitivos como puedan ser otros gimnasios u otras áreas deportivas. La competencia cuenta con más usuarios, pero nosotros cuidamos y mantenemos a nuestros clientes sin ofrecerles malas experiencias con los anuncios u otros elementos que puedan afectar a la calidad de la empresa.

1.1.3. Solución y justificación de la solución propuesta

Durante el proceso de desarrollo se han mantenido reuniones para la toma de decisiones en cuanto al marketing, así como los requisitos y calidad mínimos que se deben cumplir en la empresa. Además, hemos consultado a profesionales en el sector para alcanzar un servicio novedoso y progresivo.

La aplicación consiste en un servicio web con diversos contenidos enfocados en los temas de actualidad que están creciendo, además de ofrecer un servicio de atención al cliente de calidad y profesionalidad, porque así mantendremos a clientes y podremos crecer si tenemos buenas relaciones con ellos.

1.1.4. Destinatarios

Los destinatarios, como hemos nombrado en numerosas ocasiones, son todos los clientes potenciales, comenzando desde edades muy jóvenes como pueden ser menores de edad de entre 6 y 10 años, llegando hasta edades avanzadas, con el rango de entre 70 y 100 años.

1.2. Objetivo del Proyecto

1.2.1. Objetivos (Versión en Castellano)

Nuestro objetivo fundamental es garantizar la mejora de la calidad de vida de todas las personas, la atención al cliente de calidad y el progreso de sus vidas gracias a nuestra ayuda.

Dentro de la empresa creemos firmemente en la mejora de nuestros servicios y la expansión a otras áreas deportivas, para poder competir con otros proveedores.

Nuestra aplicación cuenta con una vista para los desarrolladores, donde pueden modificar las estructuras de las páginas y los contenidos en base al feedback de nuestros clientes.

Por otra parte, contamos con una vista para trabajadores, en la que se visualizarán el número de clientes que guardamos en nuestra base de datos. Asimismo, se cuenta con los contactos que realicen los clientes directamente a los trabajadores para contar con sus opiniones y tratarlas adecuadamente.

La vista de cliente contará con los avisos que se preparan desde la Administración, así como la posibilidad de nuevamente, contacto con los trabajadores. Incluye además ofertas y actividades que podrán visualizar y considerar.

1.2.2. Objectives (English version)

Our main goal is to ensure the enhancement of the quality of life of everyone, the customer service and the development of their lives through our help.

Within the Enterprise, we firmly believe in the enhancement of our services and the lengthening to other sports areas, to keep up with the competition.

Our application provides a view to developers, where they can modify the pages structures as well as contents based on customers feedback.

Additionally, we have another view for workers, in which the number of clients and the data of the clients that we store in our database will be displayed. In addition, the contacts made by the clients directly with the workers are relied upon to provide their opinions and deal with them accordingly.

The clients view includes announcements that are prepared in the Administration, as well as previously stated, contact with the workers. It will also include offers and activities to be viewed and considered.

2. Documento acuerdo del proyecto

2.1. Historias de Usuario

ID	Historia de Usuario	Criterios de aceptación
HU1	Como Trabajador, quiero poder acceder al historial de accesos del cliente para verificar la entrada.	Como Trabajador, para poder acceder al historial a través de la aplicación tengo que usar mis credenciales.
HU2	Como trabajador, quiero poder recibir mi sueldo anticipado	No debo solicitar dos anticipos como trabajador.
HU3	Como usuario, quiero poder darme de alta.	Debo tener mi tarjeta para poder validar mi entrada, como usuario.
HU4	Como usuario, quiero poder navegar por la página para ver ofertas.	Como usuario, acepto las condiciones y uso de la página al acceder a la página.
HU5	Como desarrollador, quiero modificar la página para ofrecer una mejor experiencia de usuario.	Para poder realizar modificaciones en la página, debo de registrarme como un desarrollador y acceder con mi cuenta de desarrollador.
HU6	Acceder al gimnasio como un cliente.	Como usuario, acepto inmediatamente las normas de uso del gimnasio cuando accedo al mismo.

2.2. Definición de Tareas

T-1 Entrega de propuesta de proyecto

T-2 Documento acuerdo del proyecto

T-3 Documento de desarrollo del primer sprint

T-4 Desarrollo del segundo sprint

T-5 Primer sprint + tareas funcionales

T-6 Documento análisis y diseño del segundo sprint

T-7 Documentación Completa

2.3. Metodología

2.3.1. Ágil

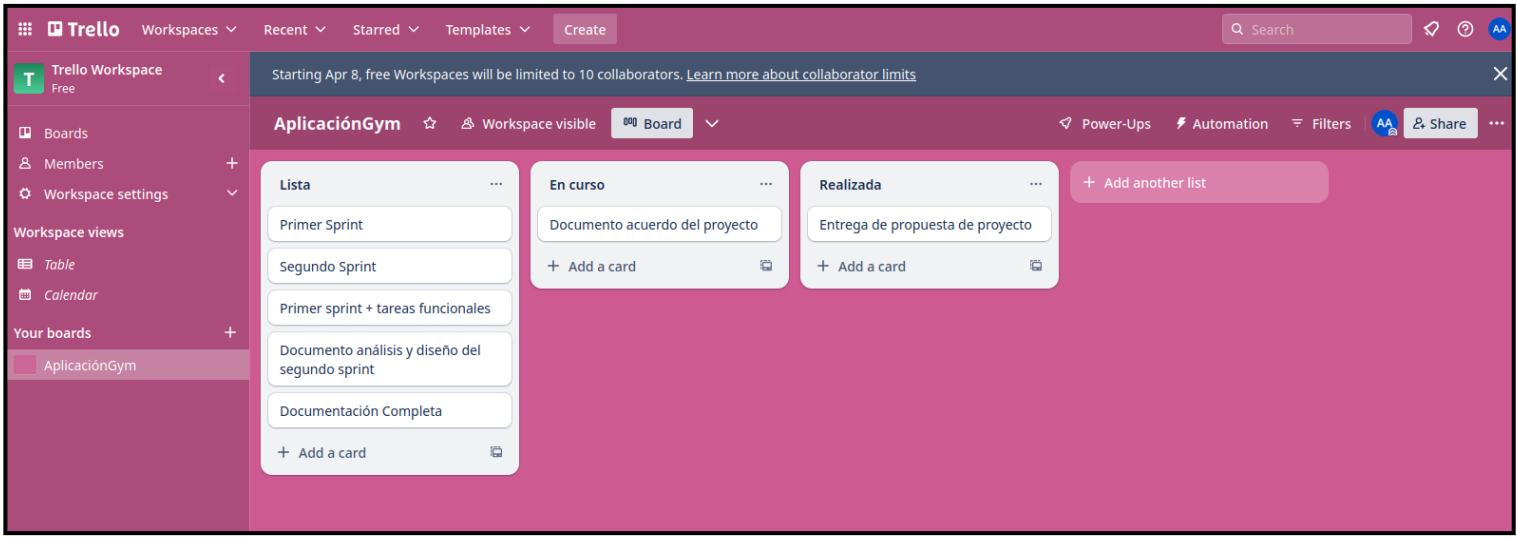

Imagen 1 Aplicación de las metodologías ágiles con Trello

2.3.2. Explicación Método

Metodología SCRUM. Esta metodología se basa en el desarrollo de proyectos complejos, iterativos e incrementales. Esto implica los sprints, en los que he establecido una cantidad de trabajo en un tiempo determinado. Se han establecido las planificaciones posteriormente, con la información de fechas y trabajos asociados a las mismas.

He elegido este método porque fundamentalmente, al trabajar con distintos periodos de entrega es un método muy bueno para trabajar además en conjunto con la herramienta de github para que se vean los commits y los distintos flujos del trabajo en conjunto.

2.4. Planificación Temporal de las Tareas

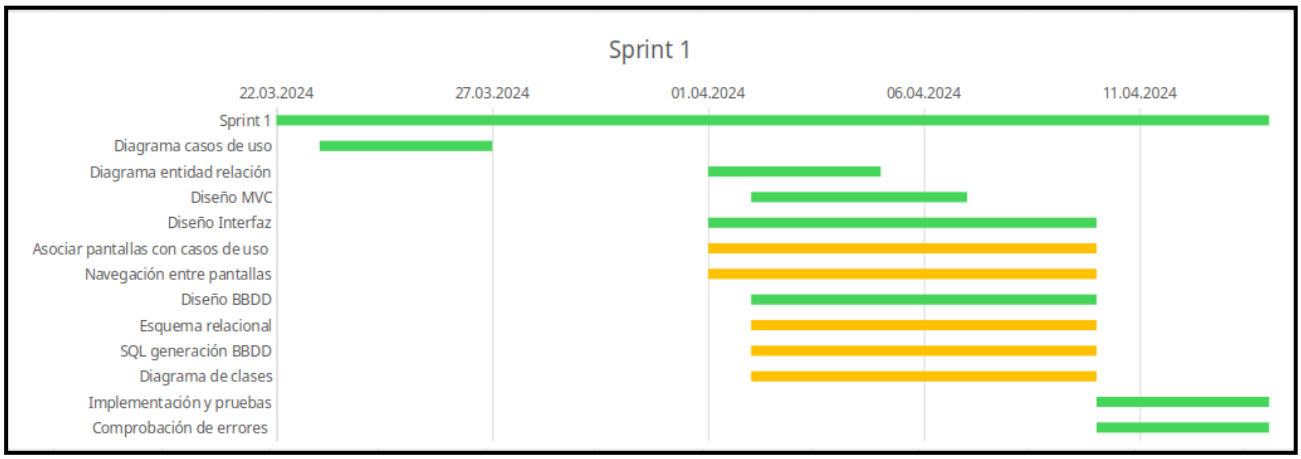

Imagen 2 Diagrama de Gantt del primer sprint

Durante este primer sprint, los primeros días se han trabajado los diagramas, en los que se detalla cómo el Trabajador tiene una serie de herramientas para contactar con el cliente.

Además, el cliente puede realizar numerosas acciones, posteriormente explicadas, en las que además se le permite contactar con el Trabajador.

Posteriormente se han desarrollado los prototipos de pantallas para todos los casos de uso de la aplicación.

Además, se ha especificado la navegación entre pantallas para ofrecer un diseño completo.

Por último, se han creado las pruebas y la implementación de la aplicación para una primera versión de esta.

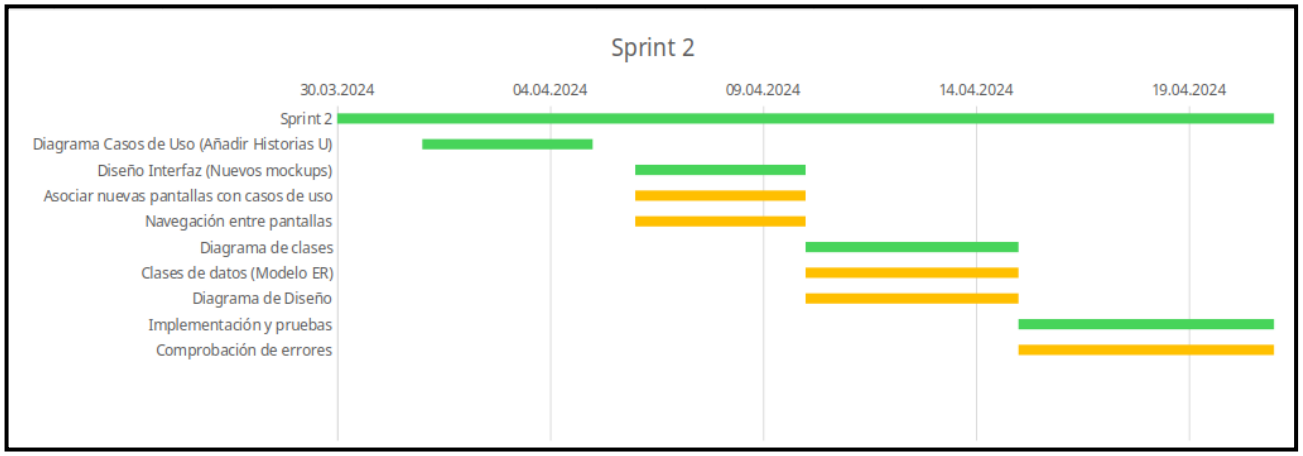

Imagen 3: Desarrollo del Segundo Sprint

En este segundo sprint, se han añadido Historias de Usuario, en las que se detallan los comportamientos de estos.

Se han modificado las pantallas y se han añadido nuevas, además de nuevas rutas por las que los diferentes usuarios pueden navegar a través de la aplicación.

Al haber creado Historias de Usuario y nuevas pantallas, se ha procedido a crear la implementación y las pruebas de la segunda versión de la aplicación, en la que además se han comprobado los posibles errores que surgen en el desarrollo.

2.5. Presupuesto y Análisis de Riesgos

2.5.1. Presupuesto

Se ha obtenido un presupuesto de 5.000€. Debido a las buenas relaciones con el cliente, hemos llegado a un acuerdo del precio por hora, además del interés por expandir nuestra empresa a cuantos más clientes mejor.

El importe ha sido reducido también por continuas modificaciones solicitadas por el cliente, en las que consideramos no aumentar significativamente el mismo.

Elementos	Precio/Hora	Importe
Análisis	8,00€	300€
Diseño	10,00€	300€
Implementación	12,00€	1010€
Pruebas	8,00€	400€
TOTAL		2010€

El total de los gastos incluye a los costes fijos que serán abonados por el cliente en su totalidad (internet, luz…), y los costes variables (herramientas, software…) que se corresponden con el patrimonio de la empresa, y se podrán emplear en futuras ocasiones, en base a un porcentaje.

Costes Variables			Costes Fijos	
Concepto	Cantidad	Precio	Concepto	Precio
Herramientas	1	100€	Internet	30€
Software	-	120€	Luz	20€
TOTAL		220€	TOTAL	50€

Beneficios

La cantidad representa a el número de proyectos (1)

Las siglas CF corresponden a los Costes Fijos.

Las siglas CVu significan el Coste Variable Unitario.

La sigla Q se refiere a la cantidad

Pt = CF/Q+Cvu

Total = 270€

En este caso se debe a que sólo es un proyecto, por lo que coincide con el coste total.

Beneficio

El presupuesto es de 5000€

P es la sigla del precio.

B = IT-CT

IT = Q*P

CT = CF+(CVu*Q)

Por tanto, realizando la operación del beneficio, nos da 4730€ de beneficio.

2.5.2. Análisis de Riesgos

Los riesgos laborales referentes al desarrollo del proyecto son los siguientes:

- Peligro por cortocircuito de elementos electrónicos.
- Lesiones musculares o de articulaciones por movimientos repetitivos.
- Diseño de las instalaciones (condiciones ambientales).

Las medidas preventivas ante estos casos son:

1. Comprobación de equipos electrónicos ante fallos, tanto de software como hardware.
2. Descansos de 5 minutos cada hora para descansar las articulaciones del cuerpo.
3. Adecuadas condiciones ambientales.
4. Elección de mobiliario adecuado para el trabajo.
5. Entorno de trabajo ergonómico.
6. Los trabajadores deben estar informados acerca de los riesgos laborales y su prevención, además de cuando se produzcan cambios.
7. Los medios necesarios son aquellas herramientas informáticas que provean al trabajador de ergonomía en el lugar de trabajo y comodidad.
8. Los equipos necesarios son los que soporten las tareas que deban llevar a cabo los trabajadores.

3. Análisis y Diseño

3.1. Modelado de datos. Análisis y diseño de la base de datos

3.1.1. Diagrama E/R

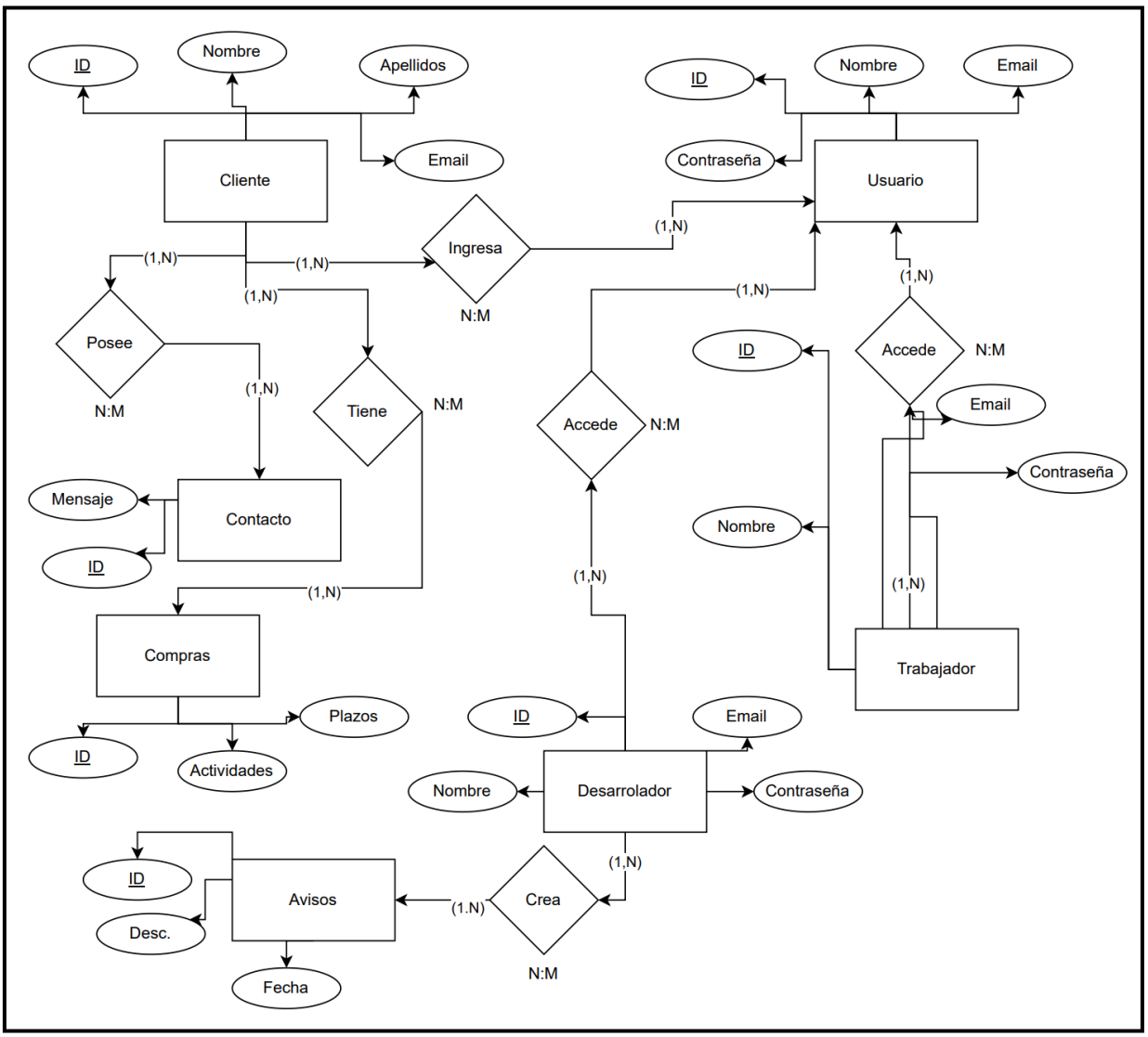

Imagen 4 En este diagrama se han representado todas las clases con las que trabajo: Contacto, Compras, Cliente, Usuario, Trabajador, Desarrollador y Avisos. El cliente puede realizar compras y contactos. El desarrollador puede crear avisos. El trabajador, por otro lado, puede ver, editar borrar y crear nuevos clientes. Los desarrolladores y trabajadores se identifican además al usar una contraseña específica. Todos los usuarios se deben de registrar antes de poder realizar compras o contactos, pero si pueden ver la página con las vistas para usuarios sin registro.

3.1.2. Diagrama Relacional

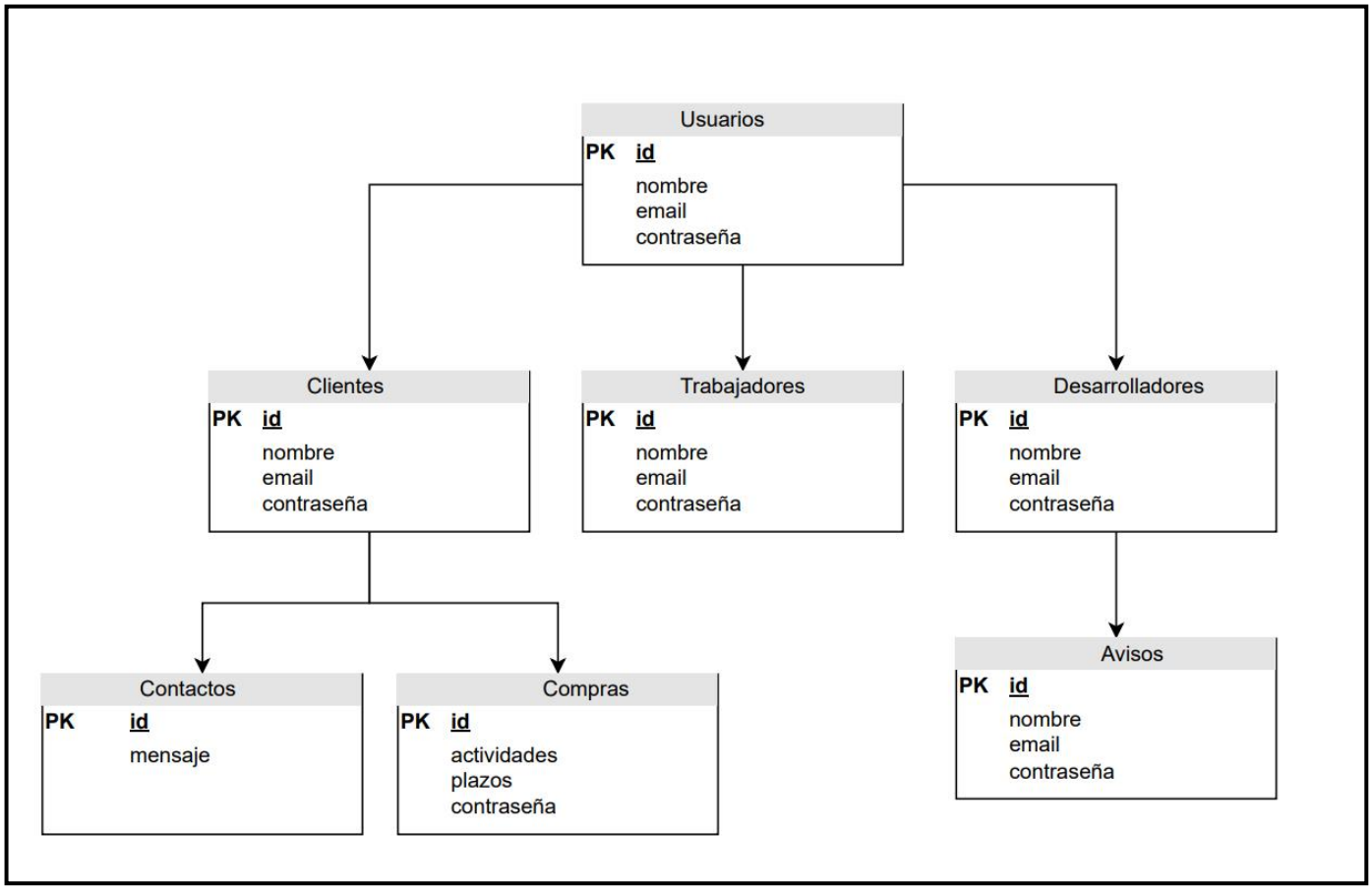

Imagen 5 En esta imagen se ha representado la organización de los datos. Los usuarios son la parte más importante, porque los otros datos (Clientes, Trabajadores y Desarrolladores) se guardan en ella. Los datos de contactos y compras se guardan en los clientes y los datos de avisos en los desarrolladores.

3.2. Análisis y diseño del sistema funcional

3.2.1. Diagrama de Clases

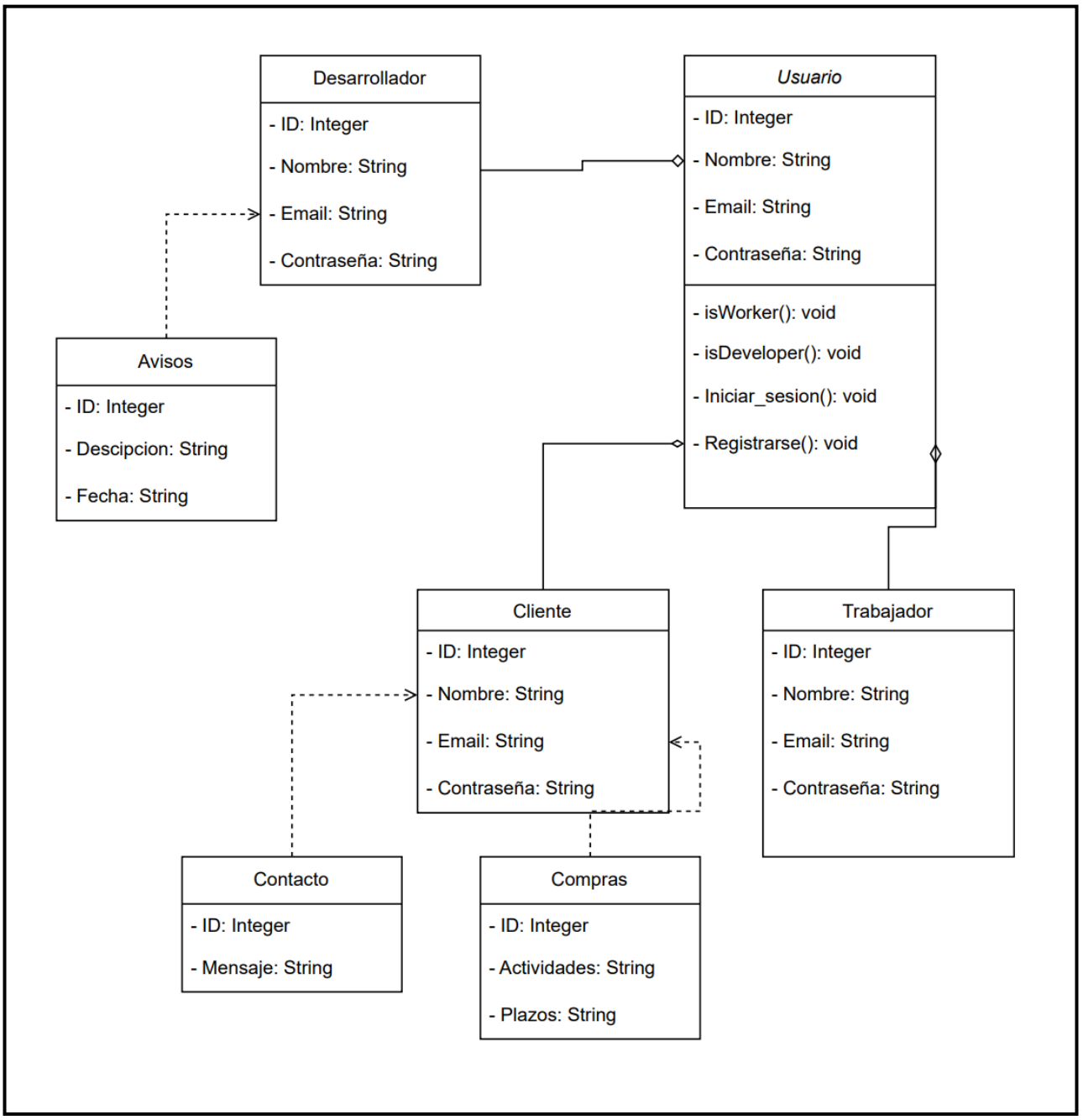

Imagen 6 Diagrama de clases. Estructura de clientes: contacto y compras dependen del cliente. Por otra parte, los avisos son dependientes del desarrollador. La clase usuario actúa de contenedor de las clases de Trabajador, Cliente y Desarrollador.

3.2.2. Diagrama de Casos de Uso

3.2.2.1. Casos de Uso de Trabajador

DCU-1 Funciones Trabajador	
Descripción:	Acciones llevadas a cabo por el trabajador en la empresa.
Actores:	Usuario Trabajador
Flujo:	Iniciar Sesión Cerrar Sesión Registro Acceso Historial Cliente Editar Cliente Borrar Cliente Añadir Cliente Mostrar Cliente

En este caso, el trabajador puede registrar una cuenta en la base de datos. Con su cuenta, puede ver entradas de clientes a través del historial de registro. Además, puede editar clientes por si, en el caso concreto, se dieran de baja o cambiaran algún dato personal.

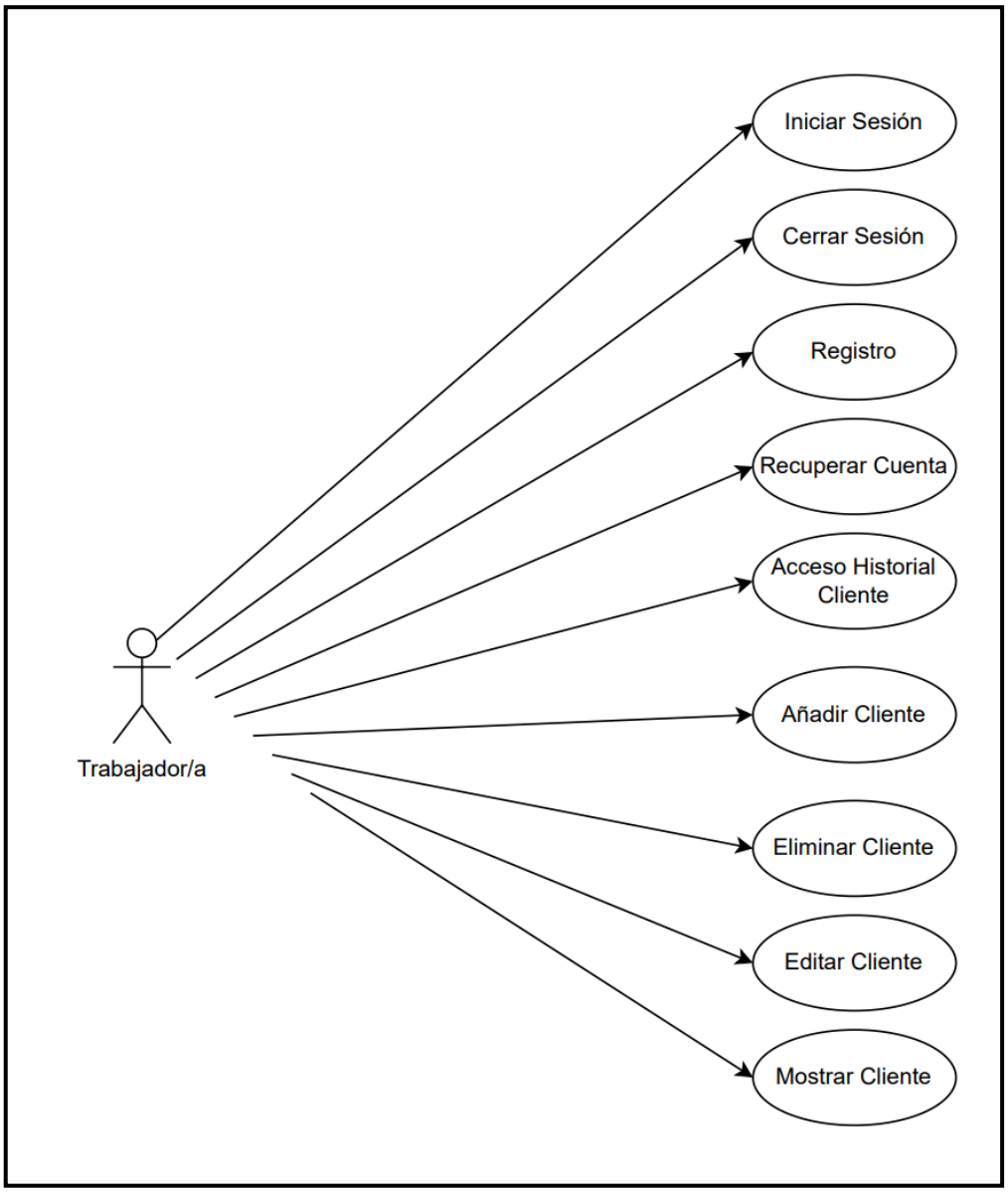

Imagen 7 Diagrama de Casos de Uso de Trabajador/a.

3.2.2.2. Diagrama de Casos de Uso de Cliente

DCU-2 Funciones Cliente	
Descripción:	Acciones llevadas a cabo por el cliente en el espacio proporcionado por la empresa.
Actores:	Usuario Cliente
Flujo:	Acceso Gimnasio Salida Gimnasio Pago cuota Cancelación Cuota Aceptación Normas de Uso

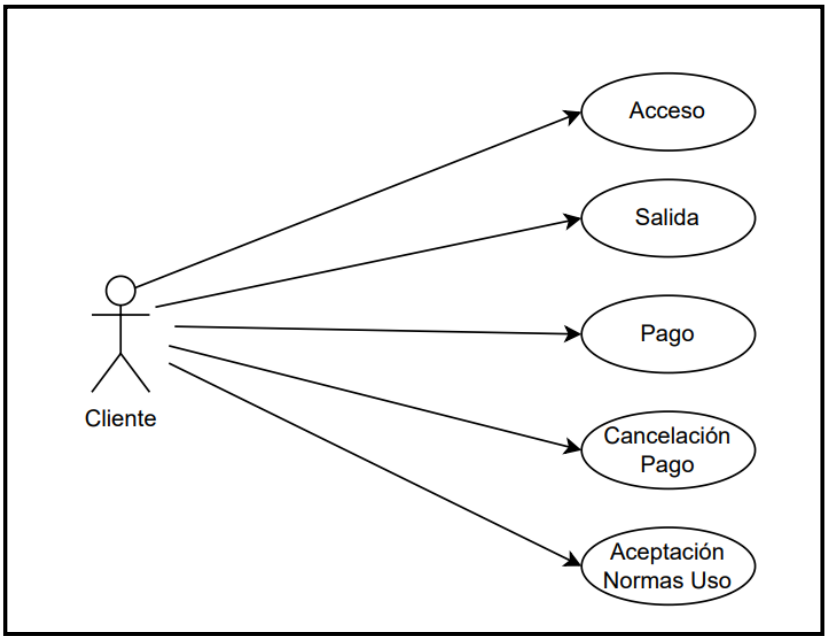

Imagen 8 Diagrama de Caso de uso de Cliente.

Aquí el cliente puede acceder al gimnasio con una cuenta, pagar una actividad y cerrar su cuenta. Al acceder, acepta todas las normas de uso y privacidad asociadas a la empresa.

3.3. Análisis y diseño de la interfaz de usuario. Mockups

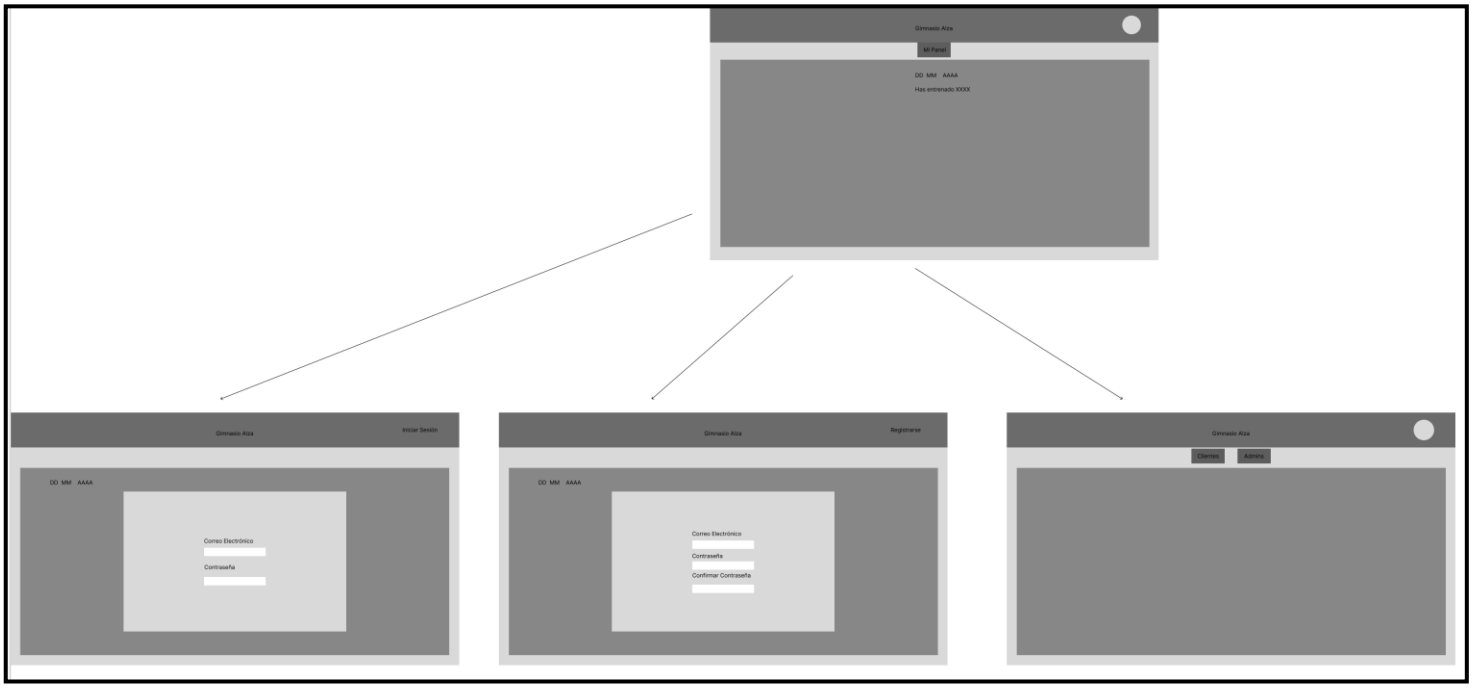

Imagen 9 Primer diseño de interfaz de Usuario, con una sección de Inicio, otra de Inicio de Sesión, Registro y otra ventana con los botones de clientes y administradores.

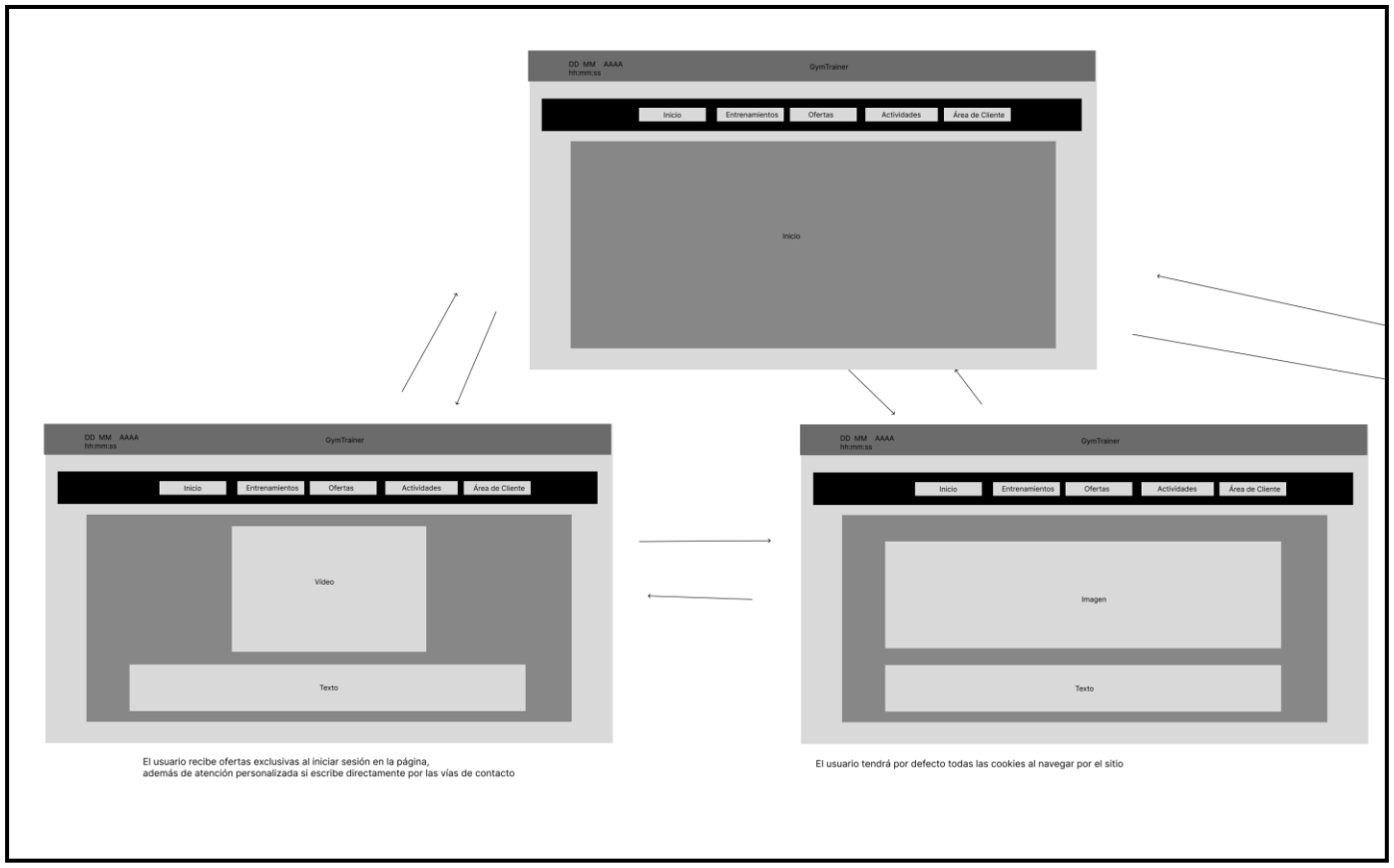

Imagen 10 Diseño derivado del primero, en el que se encuentran 3 pantallas del diseño Usuario. Cada botón de navegación (desde la izquierda a la derecha) son Inicio, Entrenamientos, Ofertas, Actividades, Área de Cliente.

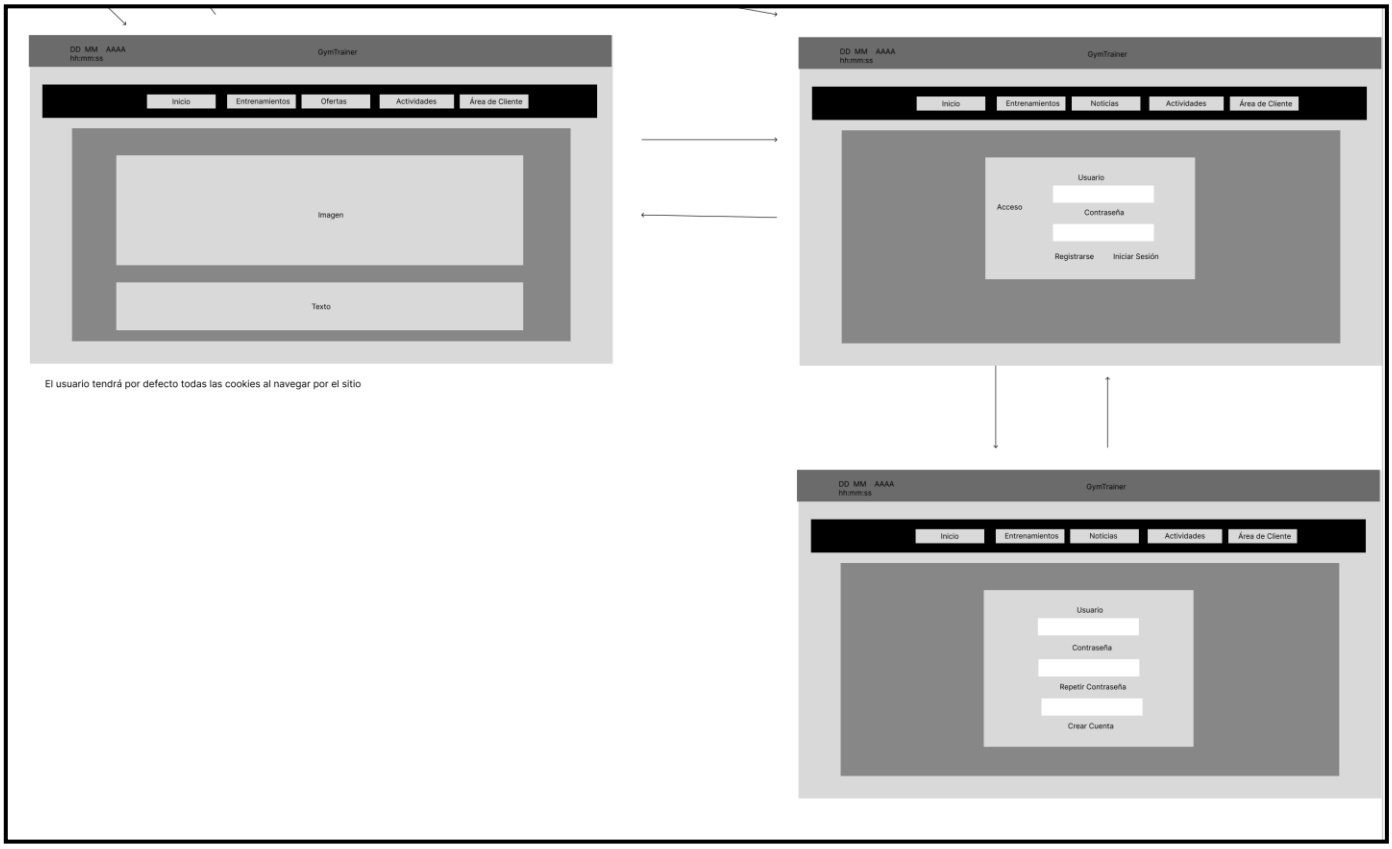

Imagen 11 Segunda parte de la vista de Usuario. En este caso, aparecen las pantallas de registro, inicio de sesión y actividades.

Imagen 12 Apartado de Historial de clientes, el trabajador podrá ver el id con el que se registra el usuario, la fecha y hora completas del momento de ingreso.

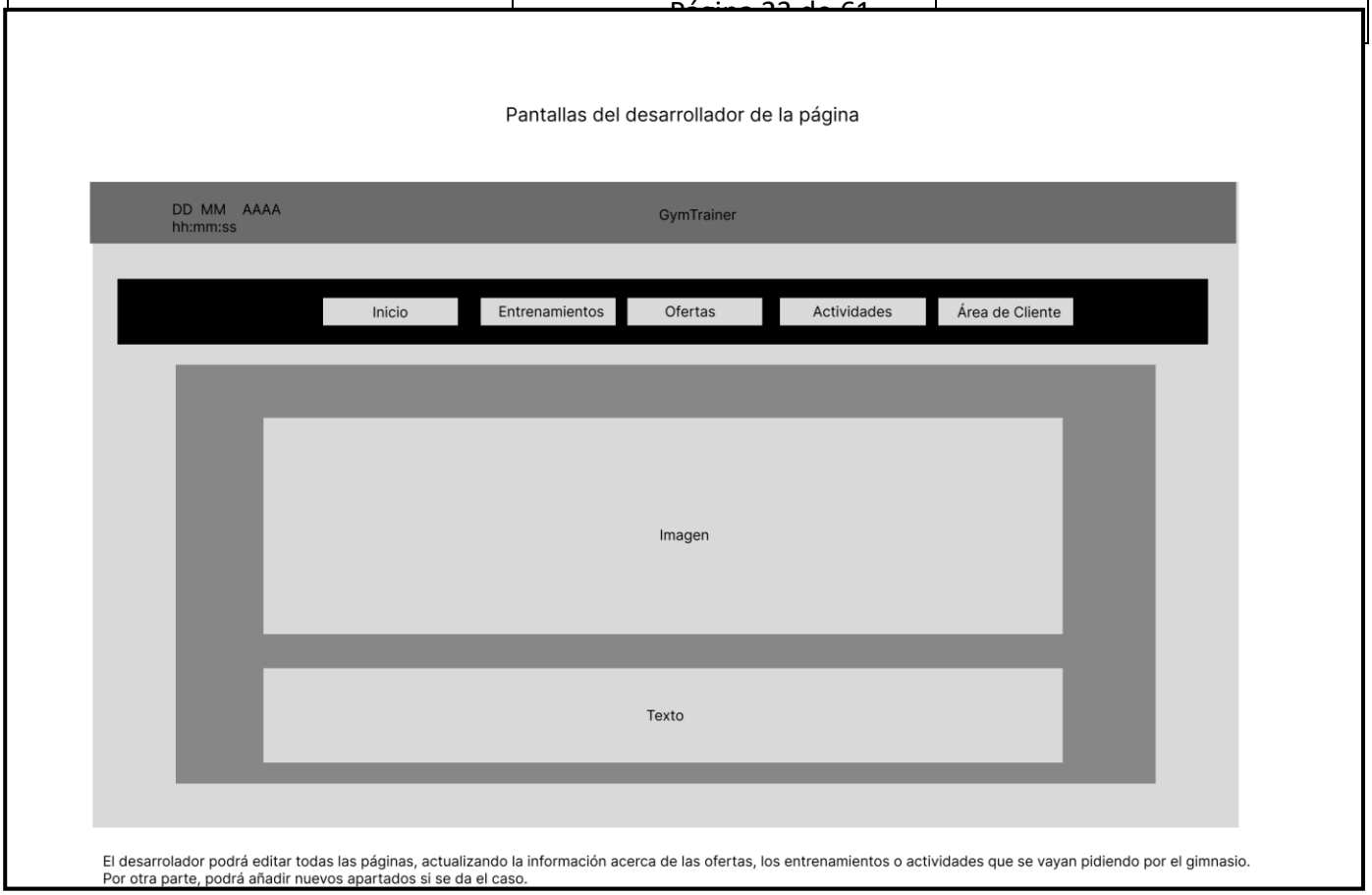

Imagen 13 Pantalla del desarrollador de la página. El desarrollador podrá ver todas las vistas del cliente y además las suya propia que es Avisos.

3.4. Diseño de la arquitectura de la aplicación

3.4.1. Tecnologías/Herramientas usadas y descripción de las mismas

He elegido Tailwind para el CSS, porque me ayuda bastante en la creación de estilos para la página, es personalizable y es también la herramienta más usada que tengo para el css. Con esta herramienta de diseño, se pueden aplicar clases directamente en el HTML específicas de esta herramienta, y con ello se crean las clases de CSS automáticamente para poder aplicarlo al HTML.

Laravel para el desarrollo siguiendo el modelo MVC, que me aporta todas las herramientas necesarias. Para las vistas he usado Blade y además he añadido algunas funcionalidades que Laravel ofrece en su framework.

GitHub, para almacenar el desarrollo del proyecto, en el que subo el mismo. GitHub es una herramienta en línea para almacenar archivos y compartirlos con otros usuarios. Esta herramienta permite gestionar y seguir los cambios a través de los commits que se van realizando. El usuario actúa como una copia de seguridad de lo que suba a GitHub, que permanece en línea.

XAMPP para ejecutar el apache y la BBDD de MySQL para el proyecto. Esta herramienta contiene Apache, MySQL y PHP entre otros. Todas estas tecnologías ofrecen un desarrollo esencial en las aplicaciones web y permite probar a los desarrolladores la aplicación en un entorno local.

3.4.2. Arquitectura de componentes de la aplicación

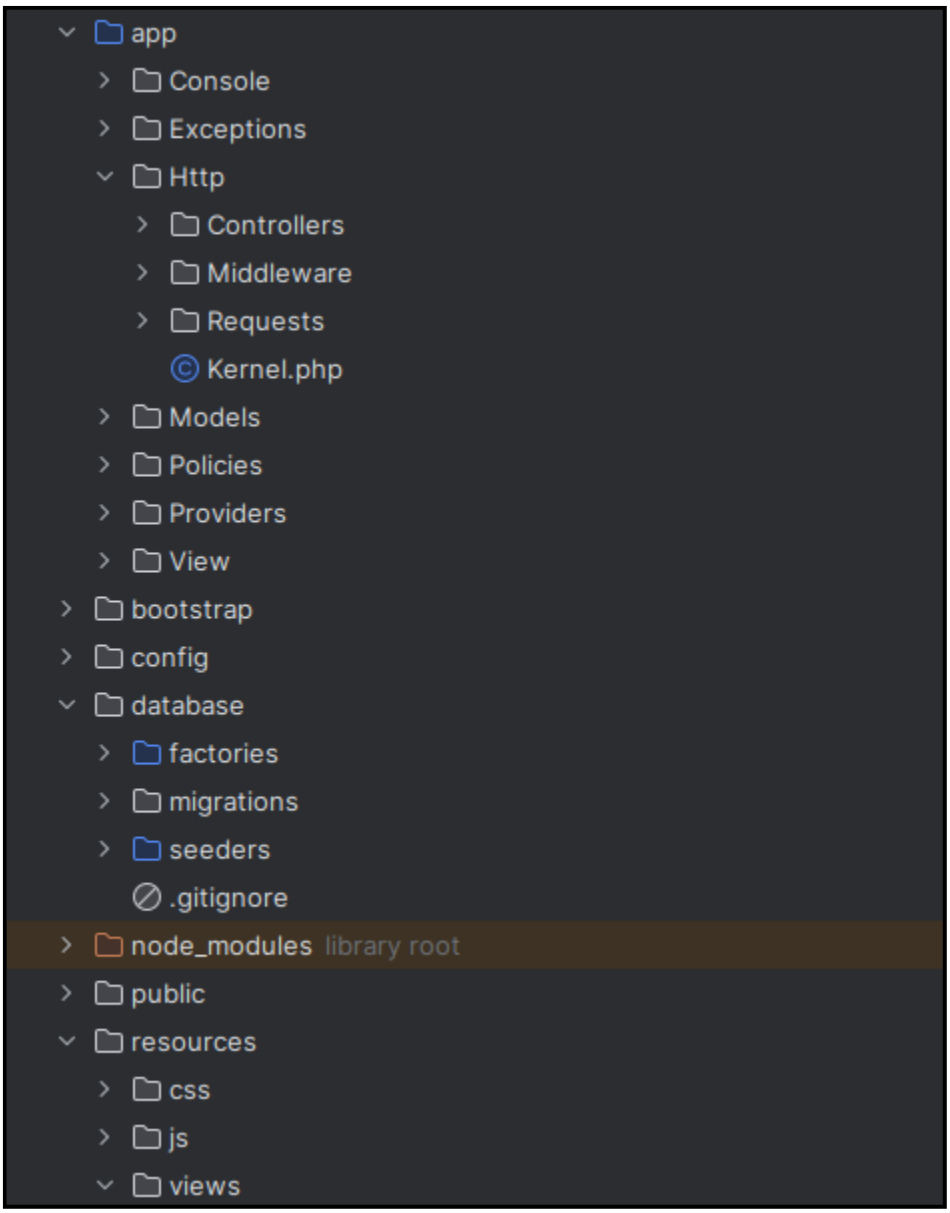

Imagen 14 Imagen de la arquitectura MVC por el framework Laravel.

En esta imagen se puede ver como las carpetas se organizan dentro de app/. Por una parte, está la de Http/ en la que se gestionan los controladores, los middleware (software de gestión de intercambio de información) y las peticiones. Por otro lado, tenemos la sección Models/ donde aparecen los modelos de la aplicación.

Después está la carpeta /database dentro de /app. En esta carpeta se guardan las capetas /factories, /migrations y /seeders. En la primera se guardan los valores para el registro en la BBDD, en la de migrations se crean tablas con los campos definidos y en la de seeders se insertan los datos con los datos del factory dentro de la tabla creada.

Por último, tenemos las Vistas, que se guardan en /app/resources/views. Las vistas son las páginas HTML que el usuario, el trabajador o el desarrollador ven, cada uno.

Imagen 15 Archivo de configuración.

El archivo .env es para pasar las variables de entorno a la aplicación.

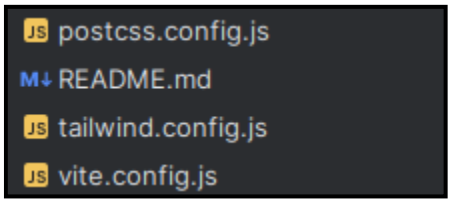

Imagen 16 Archivos generados tras instalar Tailwind por primera vez

Estos archivos (el poscss y el tailwind.config) se generan al instalar el Tailwind por primera vez en el proyecto.

Después están el README que es para poner un mensaje a otras personas para que lo lean.

Y el vite que carga los estilos definidos en el tailwind en la aplicación web.

4. Documento de Implementación, pruebas e implantación del sistema

4.1. Implementación

Los modelos nuevos que se han implementado son los de Consultas y Avisos. Cada modelo va a tener una identificación, y una fecha, para saber el momento de la creación de cada uno.

Las funciones empleadas van a ser mayormente de restricción de creación de los modelos, en los que se van a verificar casos independientes. Además, van a ver casos en los que según el usuario sea Desarrollador o Trabajador, va a poder realizar unas acciones u otras.

Los clientes podrán crear consultas a los trabajadores, ya sean consultas o avisos, pero no acceder a los datos de las tablas. Los usuarios sin registro solo podrán visualizar las pantallas, pero no acceder a ninguna pantalla de creación de datos.

Las vistas de visualización de clientes, trabajadores u otras siempre tendrán información asociada a cada acción relevante, como por ejemplo a la edición de un usuario por cambio de correo electrónico...etc.

Los Clientes pueden elegir distintos entrenamientos para el gimnasio, y estos entrenamientos sólo los pueden ver los clientes.

Los Desarrolladores y Trabajadores pueden añadir tanto nuevos entrenamientos como nuevas actividades, dependiendo del número de personas apuntadas para las mismas.

Los Trabajadores pueden ver las consultas que se les han realizado, pueden “responder” a través de la aplicación, y ese mismo mensaje los pueden leer los clientes a los que vayan dirigidos, a través de su dirección de correo electrónico.

Las vistas de las aplicaciones, como he mencionado en otras ocasiones, serán visibles según el rol que se tenga en la aplicación.

Las vistas, en cuanto a información o características pueden ser modificadas en cualquier momento por los Desarrolladores de la aplicación.

Las unidades de programación utilizadas van a ser asociadas a cada una de las clases que se han mencionado en el primer punto.

4.2. Pruebas

RF1 El usuario sin registro accede a las vistas	
Precondición	Un usuario sin registro accede a las vistas y ve los diferentes apartados
Datos de entrada	Se accede a las vistas asociadas a los usuarios sin registro
Datos de salida	El usuario observa las vistas. OK Si el usuario intenta acceder a otras vistas, aparece un mensaje de aviso. OK

RF2 El usuario registrado accede a las vistas	
Precondición	Un usuario registrado accede a las vistas y ve los diferentes apartados
Datos de entrada	Se accede a las vistas asociadas a los usuarios registrados
Datos de salida	a) El usuario observa las vistas. OK b) Si el usuario intenta crear una consulta y no rellena datos, aparece un mensaje de aviso. OK c) Si en el momento de la creación de datos hay algún error, se le notifica. OK d) En caso de que la creación sea correcta, se le notifica. OK

RF3 El trabajador accede a la vista de Consultas	
Precondición	Un trabajador accede a las Consultas de un usuario
Datos de entrada	Se accede a las vistas asociadas a los usuarios registrados
Datos de salida	a) El trabajador observa las vistas. OK b) Si el trabajador intenta eliminar algún dato de las consultas, se le notifica. OK c) Si en el momento de la visualización hay algún error, se le notifica. OK

RF4 El desarrollador accede a la vista de Avisos	
Precondición	Un desarrollador accede a los avisos de la empresa
Datos de entrada	Se accede a la vista asociadas a los desarrolladores
Datos de salida	a) El desarrollador observa las vistas. OK b) Si el desarrollador intenta eliminar algún dato de las vistas, se le notifica. OK c) Si en el momento de la visualización hay algún error, se le notifica. OK

RF5 El desarrollador o el trabajador acceden a la vistas de clientes	
Precondición	Un desarrollador o un trabajador accede a las vistas de clientes
Datos de entrada	Se accede a la vista asociadas a los clientes, se modifica la información requerida, y se actualiza.
Datos de salida	a) El Desarrollador o Trabajador observa las vistas. OK b) El Trabajador o Desarrollador modifica las vistas. OK c) Si el desarrollador intenta eliminar algún dato de las vistas, se le notifica. OK d) Si en el momento de la visualización hay algún error, se le notifica. OK

RF6 El Trabajador accede a los contactos de clientes	
Precondición	Un Trabajador accede al contacto de un cliente por correo
Datos de entrada	Se accede al correo que va a ser enviado al cliente en específico, o que ha sido mandado por el cliente mismo y va a ser contestado, se modifica la información requerida, y se actualiza.
Datos de salida	a) El Trabajador observa los correos. OK b) El Trabajador contesta al correo. OK c) Si el desarrollador intenta eliminar algún dato del correo del cliente, se le deniega. OK d) Si en el momento de la visualización hay algún error, se le notifica. OK

RF7 El desarrollador modifica la vista de Avisos	
Precondición	Un desarrollador accede a los avisos de la empresa
Datos de entrada	Se modifica la vista asociada a los desarrolladores
Datos de salida	a) El desarrollador observa las vistas. OK b) Si el desarrollador intenta modificar algún dato de la vista, se le notifica. OK c) Si en el momento de la modificación hay algún error, se le notifica. OK

Requisito RF8: El sistema permite ingresar a un cliente en la base de datos	
Precondición	Se permite ingresar correctamente un cliente a la base de datos, con el rol de trabajador.
Datos de entrada	Se añaden los campos de nombre, apellidos y correo electrónico.
Datos de salida	a) La pantalla avisa de si ha ingresado nombre. OK b) La pantalla avisa de si ha ingresado apellidos. OK c) La pantalla avisa de si ha ingresado el correo electrónico. OK d) Si lo ha ingresado todo correctamente, se procede al ingreso a la base de datos del cliente. OK

Requisito RF9: El sistema permite ver una compra en la base de datos	
Precondición	Se permite la visualización de los datos correspondientes de la base de datos, con el rol de desarrollador.
Datos de entrada	Se visualizan los campos requeridos de la compra.
Datos de salida	a) La pantalla permite ver todos los valores de la compra del cliente. OK b) La pantalla permite el filtrado de las compras para su visualización. OK

Requisito RF10: El cliente puede ver sus compras	
Precondición	Se permite la visualización de los datos correspondientes de la base de datos, con el rol de cliente.
Datos de entrada	Se visualizan los campos requeridos de la compra.
Datos de salida	a) La pantalla permite ver todos los valores de la compra del cliente. OK b) La pantalla permite el filtrado de las compras para su visualización. OK c) Si ocurriese algún error de visualización, sería notificado en la vista. OK

Requisito RF11: El cliente puede ver los correos de los trabajadores	
Precondición	Se permite la visualización de los datos correspondientes de la base de datos, con el rol de cliente.
Datos de entrada	Se visualizan los campos requeridos del correo asociado al trabajador correspondiente.
Datos de salida	a) La pantalla permite ver todos los valores del correo del Trabajador. OK b) La pantalla permite el filtrado de los correos para su visualización. OK c) Si ocurriese algún error de visualización, sería notificado en la vista. OK

4.3. Instalación/Despliegue y configuración

Primeramente, será necesario tener los prerrequisitos disponibles en el sistema operativo.

4.3.1. Instalación en Windows:

4.3.1.1. Instala Composer

- Descarga e instala Composer.
- Asegúrate de agregar la ubicación de Composer al PATH del sistema.
- En el CMD, ejecuta: composer global require laravel/installer
- Crea una carpeta vacía para almacenar el proyecto y ejecuta laravel new nombre-aplicación

4.3.1.2. Instala PHP

4.3.1.3. Instala XAMPP

4.3.1.4. Instala Tailwind

- npm install -D tailwindcss
- npx tailwindcss init
- Ve a tailwind.config.css y pon

```
/** @type {import('tailwindcss').Config} */
module.exports = {
  content: ["./src/**/*.{html,js}"],
  theme: {
    extend: {},
  },
  plugins: [],
}
```

En el resource/css/app.css

@tailwind base;

@tailwind components;

@tailwind utilities;

4.3.2. Instalación en Linux (Ubuntu)

4.3.2.1. Instala Composer

- sudo apt update
- sudo apt install composer
- ejecuta composer global require laravel/installer
- Crea una carpeta vacía para almacenar el proyecto y ejecuta laravel new nombre-aplicación

4.3.2.2. Instala XAMPP

4.3.2.3. Instala PHP

4.3.2.4. Instala Tailwind

- npm install -D tailwindcss
- npx tailwindcss init
- Ve a tailwind.config.css y pon

```
/** @type {import('tailwindcss').Config} */
module.exports = {
  content: ["./src/**/*.{html,js}"],
  theme: {
    extend: {},
  },
  plugins: [],
}
```

En el resource/css/app.css

@tailwind base;

@tailwind components;

@tailwind utilities;

4.3.3. Despliegue de aplicación y configuración

4.3.3.1. Despliegue de aplicación

Para realizar el despliegue hay que ejecutar XAMPP y levantar la BBDD de MySQL y el servidor apache.

Por otro lado, tenemos que ejecutar, en el código, el comando *php artisan serve &* para poder ejecutar el código en el navegador, y después para cargar los estilos será necesario ejecutar *npm run dev*.

4.3.3.2. Configuración de aplicación

La configuración para XAMPP

- MySQL en el puerto 3306
- Apache en el puerto 80

4.3.4. Manual de Usuario

El manual de usuario está definido en el código, y se puede encontrar en el README.md

5. Documento de Cierre

5.1. Resultados obtenidos y conclusiones

Durante el desarrollo del proyecto he estado aprendiendo acerca del framework Laravel y de cómo se maneja el modelo MVC en el mismo.

Por otra parte, he estado aprendiendo cómo interactúa Tailwind CSS con el framework de Laravel, y además he visto cómo se va rellenando el css de la página, con las clases colocadas en los archivos Blade.

He analizado además , la manera en la que el componente de MySQL se comunica con Apache a través de Docker.

Mis previsiones iniciales eran una página web sencilla, en la que se visualizaban algunas vistas para el usuario y tampoco se interactuaba demasiado, pero con el paso del tiempo he comprobado que he podido alcanzar un resultado óptimo, puesto que he podido alcanzar la funcionalidad que esperaba en la aplicación, al tener varias vistas para varios usuarios, y que éstos tengan funcionalidades diversas.

Mi planificación inicial en cuanto a la realización de horas en el proyecto era lineal, es decir, cuando terminaba un apartado, continuaba con el otro. Me he desviado de mi planificación inicial, puesto que he tenido apartados en los que he dedicado más horas que en otros, pero estoy cumpliendo con los objetivos.

Tengo posibilidades de mejora, puesto que las vistas se pueden tratar de diversas maneras y los elementos empleados en el proyecto se pueden optimizar.

6. Cuaderno de Bitácora

Como cuaderno de bitácora he optado por presentar los commits que he realizado en github.

Commits · HuBaMarin/projDespliegue (github.com)

7. Bibliografía

Installation - Laravel 10.x - The PHP Framework For Web Artisans

Mensajes de validación personalizados para Laravel

php - laravel allowing only authenticated user to access a specific route - Stack Overflow

Laravel 10 full course for beginners

Building Your First Laravel 10 CRUD Application in Under 30 Minutes | by Udara Liyanage | Medium

Laravel 10 Crash Course For Beginners. - Webdev Trainee

Getting Started with Version 10 of PHPUnit

Installation - Tailwind CSS

XAMPP Installers and Downloads for Apache Friends

8. Anexos

8.1. Código

8.1.1. Avisos

```blade
@if(auth()->user()->isDeveloper)
    <a href="{{ route('avisos.create') }}" class="btn btn-primary mb-3 mx-auto w-full text-center">Nuevo aviso</a>
    <div class="overflow-x-auto bg-gray-800">
        <table class="table w-full">
            <thead>
            <tr class="text-white border-none">
                <th>Fecha</th>
                <th>Descripción</th>
                <th>Borrar</th>
                <th>Editar</th>
            </tr>
            </thead>
            <tbody>

            @foreach($avisos as $aviso)
                <tr class="text-white border-none">
                    <td>{{ $aviso->created_at }}</td>
                    <td>{{ $aviso->descripcion }}</td>
                    <td>
                        <form action="{{ route('avisos.destroy', $aviso->id) }}" method="POST">
                            @csrf
                            @method('DELETE')
                            <button type="submit"
                                    onclick="return confirm('¿Deseas borrar el aviso {{ $aviso->mensaje }}?')"
                                    class="btn btn-error">Borrar
                            </button>
                        </form>
                    </td>
```

Imagen 17 En esta imagen muestro como se listan los avisos, además, se muestra cómo se envía un mensaje al desarrollador, en la vista, para que borre un aviso.

Imagen 18 Este método es para guardar un aviso en la base de datos. Con Laravel, se puede enviar un aviso en conjunto con la ruta, para informar al desarrollador.

8.1.2. Compras

```
{{--El cliente puede ver las compras--}}
@if(auth()->user())
    @if(!auth()->user()->isWorker && !auth()->user()->isDeveloper && auth()->user())
        <a href="{{ route('compra.create') }}" class="btn btn-primary mb-3 mx-auto w-full text-center">Nueva compra</a>
        <div class="overflow-x-auto">
            <table class="table w-full ">
                <thead>
                <tr class="text-white border-none ">
                    @isset($compra_usuario_sesion)
                        @switch($compra_usuario_sesion)
                            @case($compra_usuario_sesion>1)
                                <td>Tienes {{$compra_usuario_sesion}} compras</td>
                                @break
                            @case($compra_usuario_sesion===0)
                                <td>No tienes ninguna compra</td>
                                @break
                            @case($compra_usuario_sesion===1)
                                <td>Tienes una compra</td>
                                @break

                        @endswitch
                    @endisset
                </tr>
```

Imagen 19 En este código, se muestra cómo se cuentan las compras que ha realizado el usuario, y se comprueba si existen. El usuario ha de estar registrado para comprar también.

```
<tr class="text-white border-none">
    @foreach($compras as $compra)
        @if($compra->id_usuario === auth()->user()->id)

            <td>{{ $compra->created_at }}</td>
            <td>{{ $compra->actividades }}</td>
            <td>{{ $compra->plazos }}</td>

            <td class="border-none">
                <form action="{{ route('compra.destroy', $compra) }}" method="POST">
                    @csrf
                    @method('DELETE')
                    <button type="submit"
                            onclick="return confirm('¿Deseas borrar la compra {{ $compra->actividades }}?')"
                            class="btn btn-error ">Borrar
                    </button>
                </form>
            </td>
        @endif
    @endforeach
</tr>
```

Imagen 20 En esta imagen se muestran las compras al trabajador.

```
private function comprobarId()
{
    do {

        $max_id = count(Compra::all())+1;

        $id = rand(1, $max_id);

    } while (Compra::where('id', '=', $id)->exists());

    return $id;
}
```

Imagen 21 Aquí se muestra como se le da una id a un usuario sin registro, para que pueda comprar sin necesidad de estar registrado

```php
public function index()
{
    //

    $compras = Compra::all();
    $numero = count($compras);

    if (!isset(Auth()->user()->id))
        $id = $this->comprobarId();
    else
        $id = Auth()->user()->id;



    /*
    * Devolver las compras totales, las del usuario de la sesión y los datos
    */
    $compra_usuario_sesion = null;
    if ($compras->where('id_usuario', '=', $id)->count() > 0 ) {
        $compra_usuario_sesion = $compras->where('id_usuario', '=', $id)->count();
    }


    return view( view: 'compra.listado', compact( var_name: 'compras', ...var_names: 'numero', 'compra_usuario_sesion'));
}
```

Imagen 22 En esta imagen se ve como se guardan todas las compras y el número de ellas. Además, se asigna una id dependiendo de si el usuario se ha registrado o no. Se cuentan las compras del usuario y se pasa todo a la vista listado de compras

```php
//obtenemos el usuario
$usuario = Auth()->user();

//creamos la compra
$compra = new Compra();
$compra->actividades = implode( separator: ',', $request->actividades);
$compra->plazos = $request->plazos;
$compra->privacidad = $request->privacidad;

//asignamos el id del usuario

if (isset($usuario->id))
    $compra->id_usuario = $usuario->id;
$compra->save();

//creamos el cliente
$cliente = new Cliente();
$cliente->nombre = Auth()->user()->name ?? "";
$cliente->email = Auth()->user()->email ?? "";

if (isset($usuario->id))
    $cliente->id = Auth()->user()->id;

if ($cliente->id === $usuario->id) {
    $cliente->id = $this->comprobarId();
}
```

Imagen 23 En este apartado, está el store. Aquí se obtiene un usuario, se crea la compra, se asigna la id de usuario dependiendo de si es registrado o no lo es

```php
//si el cliente ya estaba en clientes, no lo guardamos
if ( $cliente->id!==$this->comprobarId()|| $cliente->nombre!=$usuario->name) {
    $cliente->save();
}


return redirect()->route( route: 'compra.index')->with('status', 'Compra realizada');
```

Imagen 24 Se envía toda la información a la compra, con un mensaje de compra realizada

8.1.3. Bases de Datos

```php
class DatabaseSeeder extends Seeder
{
    ✨ Codeium: Refactor  Explain
    /**
     * Seed the application's database.
     */
    public function run(): void
    {
        // \App\Models\User::factory(10)->create();

        // \App\Models\User::factory()->create([
        //     'name' => 'Test User',
        //     'email' => 'test@example.com',
        // ]);

        $this->call( class: ClienteSeeder::class);
        $this->call( class: TrabajadoresSeeder::class);
        $this->call( class: ContactoSeeder::class);
        $this->call( class: CompraSeeder::class);
        $this->call( class: AvisosSeeder::class);
    }
}
```

Imagen 25 Este apartado cubre todas las llamadas a los seeder para enviar los datos a la base de datos

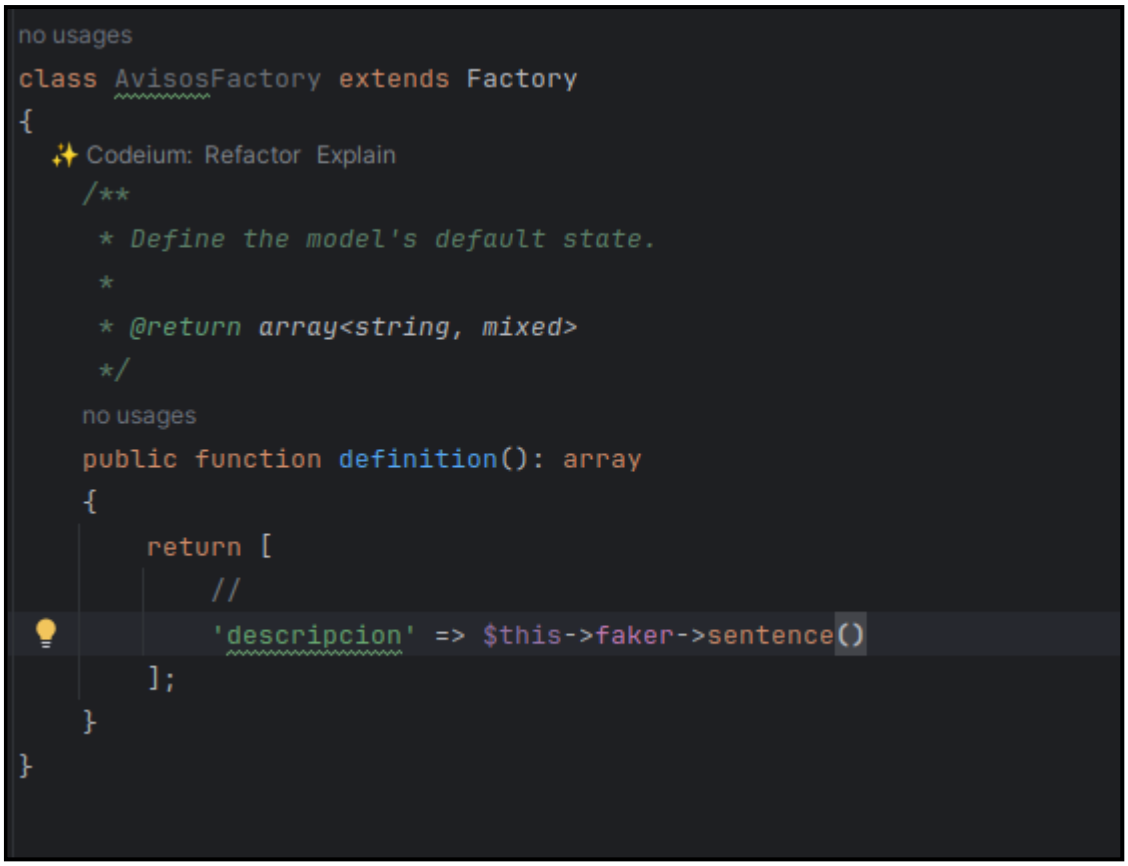

Imagen 26 Esto es el modelo base para la creación aleatoria de avisos con datos iniciales en la base de datos

```php
/**
 * Run the migrations.
 */
public function up(): void
{
    Schema::create( table: 'avisos', function (Blueprint $table) {
        $table->id();
        $table->string( column: 'descripcion');
        $table->timestamps();
    });
}

✨ Codeium: Refactor  Explain
/**
 * Reverse the migrations.
 */
public function down(): void
{
    Schema::dropIfExists( table: 'avisos');
}
```

Imagen 27 Estructura de la tabla de avisos.

```php
class Avisos extends Model
{
    use HasFactory;

    no usages
    protected $fillable = [
        'descripcion',
        'fecha'
    ];
}
```

Imagen 28 Modelo de la clase Avisos

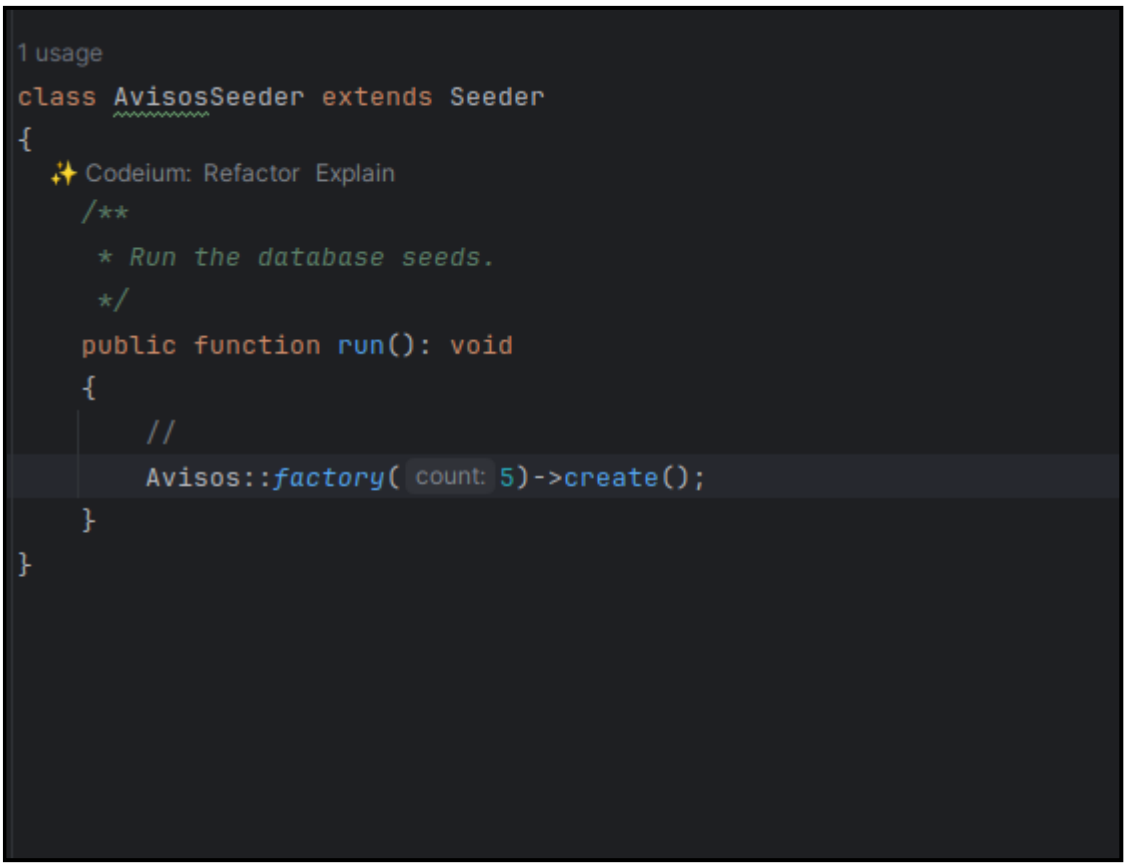

Imagen 29 Función para la ejecución del factory de avisos.

```php
 */
public function authorize(): bool
{
    return true;
}

✨ Codeium: Refactor  Explain
/**
 * Get the validation rules that apply to the request.
 *
 * @return array<string, \Illuminate\Contracts\Validation\ValidationRule|array<mixed>|string>
 */
public function rules(): array
{
    return [
        //
        'descripcion' => ['required', 'string']
    ];
}

✨ Codeium: Refactor  Explain  Docstring
public function messages() {
    return [
        'descripcion.required' => 'Es necesario rellenar la descripción',
    ];
}
```

Imagen 30 Este apartado es tanto para el StoreAvisosRequest como para el UpdateAvisosRequest. Con esto se controla tango el guardado como la edición de datos de la base de datos. Además, se pueden poner mensajes personalizados.

8.1.4. Rutas

```php
/**
 * Rutas asociadas a usuarios registrados
 */
Route::middleware(['auth'])->group(function () {

    Route::resource( name: 'clientes', controller: ClienteController::class);
    Route::resource( name: 'trabajadores', controller: TrabajadoresController::class);
    Route::resource( name: 'avisos', controller: AvisosController::class);
    Route::resource( name: 'contacto', controller: ContactoController::class);
});
```

Imagen 31 Estas rutas son para usuarios registrados, se definen en el archivo web.php

```php
/**
 * Ruta de inicio por defecto
 */
Route::get( uri: '/', function () {
    return view( view: 'inicio');
});

/**
 * Páginas o vistas
 */
Route::get( uri: '/inicio', function () {
    return view( view: 'inicio');
})->name( name: 'inicio');

Route::get( uri: '/ofertas', function () {
    return view( view: 'ofertas');
})->name( name: 'ofertas');

Route::get( uri: '/actividades', function () {
    return view( view: 'actividades');
})->name( name: 'actividades');
```

Imagen 32 Rutas de las vistas para usuarios no registrados y usuarios registrados

8.1.5. Tests

```php
/**
 * Ver si se puede borrar un cliente
 * @test
 */

$cliente = Cliente::factory()->create();

$cliente->delete();

$this->assertDatabaseMissing( table: 'clientes', ['nombre' => $cliente->nombre]);
```

Imagen 33 Se crea un cliente con el factory, se borra y se comprueba que en la base de datos no exista

```php
/**
 * Crear un cliente
 */
$cliente = Cliente::factory()->create();

$this->post( uri: '/clientes'.$cliente->id, [
    'nombre' => 'testtest',
    'email' => 'test@test.test',
]);

// Comprobar que el cliente se ha creado correctamente
$this->assertDatabaseHas( table: 'clientes', ['nombre' => $cliente->nombre]);
```

Imagen 34 Creación de un cliente. Se crea el cliente con el factory, se envía un post a /clientes (ruta definida anteriormente) y se comprueba que está en la base de datos

```php
/**
 * Editar un cliente
 * @return void
 */
public function testEditarCliente()
{
    $cliente = Cliente::factory()->create();

    // Editar el cliente
    $this->put( uri: '/clientes'.$cliente->id, [
        'nombre' => 'nuevoNombre',
        'email' => 'nuevo@test.test',
    ]);

    $this->assertDatabaseHas( table: 'clientes', ['nombre' => $cliente->nombre]);

}
```

Imagen 35 Edición de un cliente. Se crea el cliente, se envía una petición put y luego se comprueba que existe en la base de datos

```php
/**
 * Mostrar un cliente
 * @return void
 */
public function testMostrarCliente()
{
    $cliente = Cliente::factory()->create();
    $this->get( uri: "/clientes".$cliente->id);
    $this->assertDatabaseHas( table: 'clientes', ['id' => $cliente->id]);

}
```

Imagen 36 Prueba de muestra de un cliente en la BBDD.

8.1.6. Barra de navegación y Pie de página

```
<footer class="mx-auto bg-gray-800 text-white dark:bg-gray-800 border-t border-gray-100 dark:border-gray-700 text-center">
    @if(auth()->check())
        @if(auth()->user()->isWorker)
            <a href="/contacto" class="btn btn-ghost hover:bg-gray-500 sm:btn-sm md:btn-md lg:btn-lg xl:btn-xl  text-white dark:te

        @elseif(auth()->user()->isDeveloper)
          <a href="/contacto" class="btn btn-ghost hover:bg-gray-500 sm:btn-sm md:btn-md lg:btn-lg xl:btn-xl  text-white dark:text-

        @else
            <a href="{{route('aviso-legal')}}" class="btn btn-ghost hover:bg-gray-500  sm:btn-sm md:btn-md lg:btn-lg xl:btn-xl  te
            <a href="{{route('privacidad')}}" class="btn btn-ghost hover:bg-gray-500  sm:btn-sm md:btn-md lg:btn-lg xl:btn-xl  text
            <a href="/contacto" class="btn btn-ghost hover:bg-gray-500  sm:btn-sm md:btn-md lg:btn-lg xl:btn-xl  text-white dark:te
        @endif
    @else
        <a href="{{route('aviso-legal')}}" class="btn btn-ghost hover:bg-gray-500  sm:btn-sm md:btn-md lg:btn-lg xl:btn-xl  text-wh
        <a href="{{route('privacidad')}}" class="btn btn-ghost hover:bg-gray-500  sm:btn-sm md:btn-md lg:btn-lg xl:btn-xl  text-whi

    @endif
```

Imagen 37 Navegación del pie de página

```
    @guest
        {{--Usuario sin registro--}}
            <a href="{{route('inicio')}}"
               class="btn btn-ghost bg-dark hover:bg-gray-500 hover:text-white sm:btn-sm md:btn-md lg:btn-
            <a href="{{route('ofertas')}}"
               class="btn btn-ghost bg-dark hover:bg-gray-500  sm:btn-sm md:btn-md lg:btn-lg xl:btn-xl  te
            <a href="{{route('actividades')}}"
               class="btn btn-ghost bg-dark hover:bg-gray-500  sm:btn-sm md:btn-md lg:btn-lg xl:btn-xl  te
            <a href="{{route('apuntarse')}}"
               class="btn btn-ghost hover:bg-gray-500 sm:btn-sm md:btn-md lg:btn-lg xl:btn-xl  text-white
        <a href="{{route("login")}}" class="btn btn-primary text-lg">Acceder</a>
        <a href="{{route("register")}}" class="btn btn-primary text-lg">Registrarme</a>
    @endguest

    @auth
        <h3 class="text-lg">{{auth()->user()->name}}</h3>
        <form action="{{route("logout")}}" method="post">
            @csrf
            <button class="btn glass text-white" type="submit">Cerrar Sesion</button>
        </form>
    @endauth
```

Imagen 35

Imagen 38 Navegación referente a la barra de la navegación

8.2. Contrato

REUNIDOS

D__Alejandro_Marín_Andrés___, mayor de edad, con DNI__25364874Z___ y domicilio en__Utebo_(Zaragoza)_____, actuando en nombre y representación de__GymTrainer_S.L._____ inscrita en el Registro Mercantil de__Zaragoza_____ con domicilio social en ___Zaragoza____, actuando en su calidad de __Programador___, en posesión de poderes suficientes para este acto. (El Desarrollador).

D__XXXXXXXXXX___, mayor de edad, con DNI__XXXXXXXXX___ y domicilio en__XXXXXXXXXXXXX_____, en posesión de poderes suficientes para este acto. (El cliente)

MANIFIESTAN

Que las partes estén interesadas en formalizar este contrato; que poseen suficientes poderes para firmarlo; que se reconocen capacidad legal necesaria para celebrarlo y declaran que actúan de forma libre, voluntaria y no viciada.

EXPONEN

El cliente está interesado en que el prestador lleve a cabo el diseño y desarrollo de un proyecto que incluye una aplicación de escritorio o aplicación web que se utilizará como acceso a los contenidos del gimnasio, y una aplicación web, para realizar el acceso automático al espacio proporcionado por la empresa.

Que el prestador tiene como objeto el diseño y desarrollo de las aplicaciones arriba mencionadas, y que cuenta con los medios necesarios para ello; que las características encargo son las indicadas por el cliente; que ambas partes aceptan cumplir con sus respectivas obligaciones.

Con relación a lo anteriormente expuesto, las partes otorgan el presente contrato que se regirá por las siguientes clausulas:

I. OBJETO DEL CONTRATO

El presente contrato regula la prestación de servicio de diseño y programación solicitado por el cliente. Las aplicaciones se acogerán en todo caso a las categorías, diseño y contenidos firmados por ambas partes.

II. PROPIEDAD INTELECTUAL

El prestador garantiza al cliente que todo el material utilizado para el desarrollo del proyecto, así como el resultado obtenido, es un producto original que no vulnera ninguna ley o derechos de terceros, en especial los referidos a propiedad industrial e intelectual.

En caso de ser el cliente el encargado de proporcionar los contenidos (textos, vídeos, categorías…), éste se hace responsable de cualquier tipo de reclamación de terceros en relación con la titularidad de dichos contenidos, eximiendo de toda responsabilidad al prestador.

III. OBLIGACIONES DEL PRESTADOR

El prestador se compromete a desarrollar este proyecto bajo las directrices del cliente, ajustándose a los términos y condiciones definidos en el desarrollo de este contrato y conforme a las mejores prácticas existentes en el mercado y con la máxima diligencia posible.

Una vez aceptadas las características del proyecto, pueden producirse variaciones en el diseño y/o contenido de este a petición del cliente. Salvo que conllevaran una variación sustancial del proyecto inicial no supondrá aumento del precio.

El prestador se compromete a finalizar el desarrollo en el plazo acordado si la otra parte colabora en el desarrollo de este (aceptando las condiciones y términos, entregando los contenidos correspondientes, etc.)

IV. OBLIGACIONES DEL CLIENTE

El cliente se obliga a realizar el pago del precio en los términos indicados en el Anexo VI.

El cliente se obliga a mantener un contacto constante con el prestador entregando en tiempo y forma los contenidos del proyecto (Textos, imágenes, videos, categorías…), la aceptación del diseño y cualquier otra necesidad que requiera el prestador para poder finalizar el proyecto. En cualquier caso, se atenderá a lo dispuesto en la cláusula II.

En caso de depender la entrega de contenidos de un tercero seleccionado por el cliente, éste deberá indicarlo en el apartado comunicaciones y comprometiéndose a responder ante cualquier responsabilidad.

V. COMUNICACIONES

Las partes se obligan a comunicarse toda la información que pudiera ser necesaria para el correcto desarrollo del proyecto. Toda comunicación entre las partes relativa al presente contrato se realizará por escrito o teléfono. A efectos de comunicaciones y/o notificaciones las partes designan:

Prestador

Domicilio en _____, con número de Fax _____, correo electrónico _____ y Teléfono _____

Cliente

Domicilio en _____, con número de Fax _____, correo electrónico _____ y Teléfono _____

Cualquier cambio de domicilio o dirección de contacto deberá ser comunicado a la parte correspondiente por escrito, teléfono o cualquier medio disponible con una antelación mínima de 2 días hábiles.

VI. DURACIÓN Y PRECIO

Este contrato entrará en vigor el 23 de marzo de 2024 y durará 2 meses, a petición del prestador, añadir 10 días más para entregar el proyecto, sin suponer repercusión económica alguna.

El precio para abonar por parte del cliente como pago por la prestación del servicio prestado equivale a 5.000 € (cinco mil euros).

Dicho precio será abonado de la siguiente forma:

Se irán abonando porcentajes de dinero disponible por el cliente, en el momento de la firma de este mismo contrato, cada semana.

El porcentaje final debe alcanzarse cuando termine el proyecto.

Si el prestador no cumple el plazo de entrega o el cliente no ha entregado el dinero correspondiente se dará por vencido el contrato presente, debiendo finalizar el proceso de contenidos, diseño o entrega de las aplicaciones encargadas por el cliente, y el pago de plazos por parte del cliente.

El pago será realizado mediante transferencia Bancaria en el número de cuenta XXXX XXXXXXXXXXX.

VII. CONFIDENCIALIDAD Y PROTECCIÓN DE DATOS

Este Acuerdo de Confidencialidad y Protección de Datos (el "Acuerdo") se celebra entre las partes contratantes (las "Partes").

1. Confidencialidad: Cada Parte se compromete a mantener en estricta confidencialidad toda la información que reciba de la otra Parte en el marco de este Acuerdo, a no divulgar dicha información a terceros sin el consentimiento previo por escrito de la otra Parte y a utilizar dicha información únicamente para los fines de este Acuerdo.

2. Protección de Datos: Cada Parte se compromete a cumplir con todas las leyes y regulaciones aplicables en materia de protección de datos. Cada Parte implementará medidas técnicas y organizativas adecuadas para proteger los datos personales contra la destrucción accidental o ilícita, la pérdida accidental, la alteración, la divulgación o el acceso no autorizados y contra cualquier otra forma de tratamiento ilícito.

3. Incumplimiento: En caso de incumplimiento de este Acuerdo, la Parte infractora será responsable de todos los daños y perjuicios que pueda sufrir la otra Parte.

Este Acuerdo se rige por las leyes del país de las Partes contratantes. Al firmar ambas partes se da por acordado y aceptado el cumplimiento de los elementos descritos en este documento de contrato de aplicación.

VIII. NO COMPETENCIA

El cliente se compromete a seguir la confidencialidad con respecto al punto 1 del apartado VIII, descrito anteriormente.

El incumplimiento del anterior compromiso llevará aparejada una penalización equivalente a 3.000€, sin perjuicio y de las indemnizaciones que correspondiesen por los daños y perjuicios causados al prestador.

IX. EXTINCIÓN

Además de por las causas generales del Derecho, este contrato se extinguirá:

- Por el transcurso de este.

Por declararse en suspensión de pagos, quiebra o concurso de acreedores de cualquiera de las partes.

Por incumplimiento de las obligaciones previstas en este contrato.

X JURISDICCIÓN Y LEGISLACIÓN APLICABLE

Todas las cuestiones litigiosas sobre el presente contrato mercantil quedarán regidas por lo descrito en el apartado VIII sobre legislación.

En cualquier caso, será obligatorio que en caso de conflicto las partes intenten previamente resolver la cuestión de mutuo acuerdo, sometiéndose en su caso a los Juzgados y Tribunales del país correspondiente.