



AUTOESCUELA ALMANSA.ES

ANTONIO ALMANSA BECERRA

Curso 2024/2025

Índice Proyecto – Autoescuela Almansa.es

Introducción y presentación del proyecto.....	2
1.1 Objetivo del Proyecto.....	3
1.2 ¿Cómo funciona la autoescuela digital?.....	3
1.3 Beneficios del proyecto.....	3
1.4 Importancia del proyecto.....	4
Análisis de requisitos.....	4
2 Requisitos funcionales.....	4
2.1 Requisitos técnicos.....	5
Diseño del modelo de datos.....	6
3 Modelo Entidad-Relación de la Base de Datos de una Autoescuela.....	6
3.1 Usuarios (clientes).....	6
3.2 Profesores.....	7
3.3 Compras.....	7
3.4 Contactos.....	8
3.5 Preguntas_Test.....	8
3.6 Respuestas_Usuario.....	9
3.7 Errores_Usuario.....	9
3.8 Ayudas.....	10
3.9 Ayudas_Test_Aleatorios.....	10
3.10 Preguntas_Test_Temáticos.....	10
3.11 ¿Cómo comprobé que funcionaban la conexión entre tablas?.....	11
3.12 Como resumen del punto de la Representación de la Base de Datos.....	12
Diseño del Sitio Web.....	12
4 Introducción general al diseño del sitio web.....	12
Manual de Usuario y Técnico.....	14
5 ¿Qué es el Manual de Usuario?.....	14
5.1 ¿Qué es el Manual Técnico?.....	15
5.1 ¿Cómo voy a hacer mi Manual de Usuario y Técnico?.....	15
5.1.1 Parte Usuario.....	16
5.1.2 Parte Profesores.....	52

Introducción y presentación del proyecto

Esta documentación va a describir el desarrollo de una autoescuela digital: una plataforma web moderna diseñada para facilitar y mejorar el proceso de aprendizaje teórico para futuros conductores. A diferencia del modelo tradicional, donde es necesario asistir físicamente a clases en una autoescuela, esta nueva propuesta aprovecha los avances tecnológicos para ofrecer una alternativa más accesible, flexible y eficaz.

La idea surge de la necesidad de adaptar la formación vial a los tiempos actuales, permitiendo que cualquier persona pueda prepararse para obtener el permiso de conducir sin limitaciones

de horario, ubicación o recursos. En este documento se explican los objetivos del proyecto, cómo funciona la plataforma, sus beneficios principales y su importancia dentro del contexto de la educación vial.

1.1 Objetivo del Proyecto

El objetivo principal de esta autoescuela digital es proporcionar una formación teórica completa, cómoda y de calidad para todas aquellas personas que quieran obtener su carnet de conducir.

En lugar de depender exclusivamente de clases presenciales, la plataforma permitirá a los usuarios estudiar desde cualquier lugar, en el momento que les resulte más conveniente, y desde cualquier dispositivo con acceso a internet.

Además, se busca mejorar la preparación de los estudiantes mediante simulaciones de examen y un sistema de seguimiento personalizado. Gracias a estas herramientas, los alumnos podrán identificar fácilmente sus puntos fuertes y débiles, lo que aumentará sus posibilidades de aprobar y, más importante aún, de convertirse en conductores responsables y conscientes.

1.2 ¿Cómo funciona la autoescuela digital?

Esta plataforma está organizada de forma clara e intuitiva para que cualquier usuario, sin importar su nivel de experiencia con la tecnología, pueda utilizarla sin complicaciones. Su funcionamiento se basa en los siguientes elementos clave:

- **Simuladores de examen:** Permiten a los alumnos practicar con pruebas similares a las oficiales, para familiarizarse con el formato del examen real.
- **Seguimiento del progreso:** Cada usuario podrá ver sus resultados, repasar sus errores y recibir recomendaciones personalizadas para mejorar.
- **Acceso desde cualquier dispositivo:** La plataforma es totalmente compatible con ordenadores, tablets y móviles, sin necesidad de descargar ni instalar programas.

1.3 Beneficios del proyecto

La implementación de esta autoescuela digital ofrece numerosos beneficios tanto para los estudiantes como para los profesores. Entre los más destacados se encuentran:

- **Flexibilidad de aprendizaje:** Los usuarios pueden estudiar en el horario que prefieran y desde el lugar que les resulte más cómodo.
- **Mayor tasa de aprobación:** Al contar con una gran cantidad de test desde temáticos, aleatorios y de examen, los estudiantes llegan mejor preparados al examen oficial.
- **Reducción de costos:** No se requiere material impreso ni desplazamientos, lo que hace que la formación sea más económica.
- **Acceso sin fronteras:** Cualquier persona con acceso a internet puede utilizar la plataforma, sin importar en qué parte del país (o incluso del mundo) se encuentre.
- **Mejora de la seguridad vial:** Una mejor formación teórica contribuye a reducir los accidentes y promueve una conducción más consciente y respetuosa.

1.4 Importancia del proyecto

La educación vial es una pieza clave en la seguridad de conductores, peatones y ciclistas. A través de esta autoescuela digital, se busca ofrecer una solución moderna que permita llegar a más personas, con una enseñanza más clara, accesible y personalizada.

Este proyecto no solo pretende facilitar la obtención del carnet de conducir, sino también contribuir activamente a la formación de conductores más preparados, responsables y seguros. Con una buena implementación y difusión, la autoescuela digital podría convertirse en una herramienta clave para transformar y mejorar la educación vial en el futuro.

Análisis de requisitos.

Antes de comenzar con el desarrollo de la aplicación, es imprescindible establecer tanto los requisitos funcionales como los técnicos. Esto asegurará que la plataforma no solo cumpla con sus objetivos educativos, sino que también funcione correctamente desde el punto de vista técnico y proporcione una buena experiencia a los usuarios.

2 Requisitos funcionales

Los requisitos funcionales definen las acciones que podrá realizar el usuario dentro de la autoescuela digital, así como las principales herramientas que la plataforma debe ofrecer:

- Registro y autenticación:** Los estudiantes deben poder crear su cuenta, iniciar sesión y administrar su información personal.
- Exámenes tipo test con resultados:** El sistema debe generar pruebas aleatorias similares a los exámenes oficiales y mostrar el resultado final junto a las respuestas correctas.

101	Test Avanzado	30	10	20	11.11%	Suspension
1	Señales de tráfico	0	0	0	0%	Suspension
201	Test de Diaman	0	0	0	0%	Suspension

Detalle del Intento 86				
Número de Pregunta	Pregunta	Tu respuesta	Estado	
1	¿Qué indica una señal de intersección con borde rojo y un coche detenido?	A	Correcta	
2	¿Qué significa una señal circular con fondo azul y una flecha blanca hacia la derecha?	A	Incorrecta	
3	Una señal de stop/pausa se debe:	C	Correcta	
4	¿Qué indica una señal circular con borde rojo y un coche detenido?	B	Incorrecta	
5	Una señal triangular con un tren detenido adverte de:	C	Incorrecta	
6	¿Qué indica una señal con una V blanca sobre fondo azul?	C	Correcta	
7	Una señal circular con una linea roja diagonal sobre una bicicleta significa:	C	Incorrecta	
8	¿Qué significa una señal de fondo azul con una figura blanca de persona caminando?	C	Incorrecta	
9	Una señal de tránsito invertida con borde rojo indica:	C	Incorrecta	
10	¿Qué indica una señal con fondo azul y dos ruedas, uno rojo y otro negro?	A	Incorrecta	
11	¿Qué indica una señal circular con borde rojo y numero dentro?	A	Correcta	
12	Una señal de paso de peatones azul:	C	Incorrecta	
13	¿Qué significa una señal de dirección prohibida?	B	Correcta	
14	¿Qué indica una señal circular blanca con una linea negra en diagonal?	B	Incorrecta	
15	Una señal de forma octogonal y color rojo siempre indica:	C	Correcta	
16	¿Qué significa una señal circular con una flecha curva y textada?	A	Correcta	
17	¿Qué indica una señal cuadrada azul con un coche y linea diagonal?	A	Incorrecta	
18	Una señal triangular con rallas curvadas indica:	B	Incorrecta	
19	Una señal con un círculo roscado significa:	C	Incorrecta	
20	¿Qué indica una señal con un semáforo en su interior?	C	Incorrecta	
21	Una señal azul con una bicicleta blanca significa:	C	Incorrecta	

- Control del progreso:** Cada usuario debe poder consultar su historial de pruebas y ver cómo avanza en los distintos módulos.
- Corrección automática:** Las evaluaciones deben corregirse al instante, mostrando una nota y un resumen de los errores cometidos.
- Accesibilidad desde distintos dispositivos:** Debe funcionar de forma fluida tanto en ordenadores como en tablets y teléfonos móviles.
- Diseño fácil de usar:** La interfaz debe ser clara e intuitiva, pensada para personas con distintos niveles de experiencia digital.



2.1 Requisitos técnicos

Este apartado recoge las tecnologías necesarias y las características mínimas que debe tener el entorno donde se ejecutará la aplicación:

- Tecnologías utilizadas:** Se usará PHP con XAMPP para la lógica del servidor y conectar la base de datos de phpMyAdmin. JavaScript para funcionalidades interactivas, y HTML/CSS para estructurar y diseñar las páginas.
- Gestión de datos:** Se empleará una base de datos como MySQL con XAMPP para almacenar la información de usuarios, compras, temario de test, etc.

A screenshot of the phpMyAdmin interface. The left sidebar shows a tree view of databases: 'autoescuela' (selected), 'information_schema', 'mysql', 'performance_schema', 'phpmyadmin', and 'test'. Under 'autoescuela', there are tables like 'ayudas', 'ayudas_test_aleatorios', 'backup_errores_usuario', 'backup_resuestas_usuario', 'compras', 'contactos', 'errores_usuario', 'preguntas_test', 'preguntas_test_aleatorios', 'profesores', 'respuestas_usuario', and 'temas'. The main area displays a table titled 'eyudas' with columns: 'Número', 'Acción', 'Filas', 'Tipo', 'Cotejamiento', 'Tamaño', and 'Residuo a depurar'. The table contains 30 rows of data. At the bottom, it says '14 tablas' and 'Número de filas 877'. There are also buttons for 'Examinar', 'Estructura', 'Buscar', 'Insertar', 'Vaciar', and 'Eliminar'.

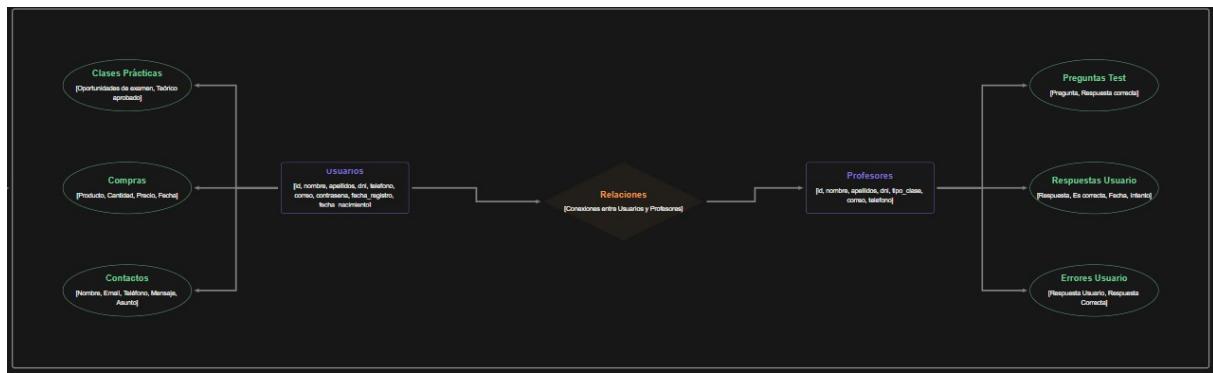


- **Servidor web:** Se utilizará un servidor como Apache para ejecutar los archivos PHP y comunicarse con bases de datos MySQL .

Diseño del modelo de datos.

3 Modelo Entidad-Relación de la Base de Datos de una Autoescuela

El modelo entidad-relación representa la estructura lógica de una base de datos pensada para gestionar el funcionamiento de una autoescuela. Este modelo está compuesto por diez entidades (tablas), cada una con atributos específicos y relaciones entre ellas. A continuación se explican las funciones de cada entidad y sus vínculos principales.



3.1 Usuarios (clientes).

Esta entidad almacena los datos de las personas que se registran como alumnos en la autoescuela. Contiene información personal como nombre, apellidos, correo electrónico, fecha de nacimiento, número de clases prácticas realizadas, número de oportunidades de examen, si ha aprobado la parte teórica y el profesor asignado.

```
--  
-- Estructura de tabla para la tabla `clientes`  
--  
  
CREATE TABLE `clientes` (  
  `id` int(11) NOT NULL,  
  `nombre` varchar(50) NOT NULL,  

```

Relaciones:

- Cada usuario puede tener un profesor asignado.
- Un usuario puede haber realizado varias compras, responder muchas preguntas de test y cometer errores en ellas.

3.2 Profesores

Aquí se guarda la información del personal docente de la autoescuela. Se incluye el nombre, apellidos, tipo de clase que imparte (teórica o práctica), contacto y datos de registro.

```
--  
-- Estructura de tabla para la tabla `profesores`  
--  
  
CREATE TABLE `profesores` (  
  `id` int(11) NOT NULL,  
  `nombre` varchar(100) NOT NULL,  

```

Relaciones:

- Un profesor puede estar asignado a varios alumnos (usuarios), pero cada alumno tiene un único profesor asignado.

3.3 Compras

Esta entidad registra todas las compras que realiza un alumno. Puede tratarse de cursos teóricos, prácticos o paquetes combinados. Se guarda el nombre del producto, su precio, la cantidad y la fecha de compra.

```
--  
-- Estructura de tabla para la tabla `compras`  
--  
  
CREATE TABLE `compras` (  
    `id` int(11) NOT NULL,  
    `usuario_id` int(11) NOT NULL,  

```

Relaciones:

- Cada compra está asociada a un usuario.
- Un usuario puede tener muchas compras registradas.

3.4 Contactos

Contiene los datos que se recogen a través del formulario de contacto de la autoescuela. Se guarda el nombre de la persona que escribe, su correo, el teléfono, el asunto de la consulta y el mensaje que envía.

```
--  
-- Estructura de tabla para la tabla `contactos`  
--  
  
CREATE TABLE `contactos` (  
    `id` int(11) NOT NULL,  
    `nombre` varchar(100) NOT NULL,  

```

Relaciones:

- Esta tabla no tiene relaciones directas con otras entidades, aunque está relacionada conceptualmente con los usuarios o personas interesadas.

3.5 Preguntas_Test

Es una de las entidades centrales del sistema. Contiene todas las preguntas utilizadas en los exámenes o prácticas teóricas, junto con su respuesta correcta, el tema al que pertenece y su categoría (temático o aleatorio).

```
--  
-- Estructura de tabla para la tabla `preguntas_test`  
  
CREATE TABLE `preguntas_test` (  
  `id` int(11) NOT NULL,  
  `pregunta` text NOT NULL,  
  `respuesta_correcta` varchar(255) NOT NULL,  
  `tema_id` int(11) DEFAULT NULL,  
  `categoria` varchar(50) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

Relaciones:

- Una pregunta puede tener muchas respuestas de usuarios.
- También puede tener errores asociados cuando los usuarios fallan.
- Además, cada pregunta puede tener una ayuda vinculada.

3.6 Respuestas_Usuario

Esta tabla almacena cada respuesta que un alumno da a una pregunta. Registra si la respuesta fue correcta, la fecha, el intento en el que se respondió y el tema correspondiente.

```
--  
-- Estructura de tabla para la tabla `respuestas_usuario`  
  
CREATE TABLE `respuestas_usuario` (  
  `respuesta_id` int(11) NOT NULL,  
  `usuario_id` int(11) NOT NULL,  
  `pregunta_id` int(11) DEFAULT NULL,  
  `respuesta` varchar(255) DEFAULT NULL,  
  `es_correcta` tinyint(1) NOT NULL,  
  `fecha_respuesta` timestamp NOT NULL DEFAULT current_timestamp(),  
  `intento` int(11) DEFAULT 1,  
  `tema_id` int(11) DEFAULT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

Relaciones:

- Cada respuesta pertenece a un usuario y a una pregunta concreta.

3.7 Errores_Usuario

Registra los errores cometidos por los alumnos en los tests. Para cada error se almacena el usuario, la pregunta fallada, la respuesta que dio el alumno, la correcta y el intento.

```
--  
-- Estructura de tabla para la tabla `errores_usuario`  
  
CREATE TABLE `errores_usuario` (  
  `error_id` int(11) NOT NULL,  
  `usuario_id` int(11) NOT NULL,  
  `pregunta_id` int(11) NOT NULL,  
  `respuesta_usuario` varchar(1) DEFAULT NULL,  
  `respuesta_correcta` varchar(1) DEFAULT NULL,  
  `intento` int(11) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

Relaciones:

- Cada error está asociado a un usuario y a una pregunta del test.

3.8 Ayudas

Esta entidad ofrece explicaciones o aclaraciones asociadas a las preguntas. Por ejemplo, si un alumno falla una pregunta, se le puede mostrar un texto de ayuda que le explique la señal o el concepto relacionado.

```
--  
-- Estructura de tabla para la tabla `ayudas`  
  
CREATE TABLE `ayudas` (  
  `id` int(11) NOT NULL,  
  `pregunta_id` int(11) NOT NULL,  
  `texto_ayuda` text NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

Relaciones:

- Cada ayuda está vinculada a una pregunta específica.

3.9 Ayudas_Test_Aleatorios

Funciona de forma similar a la tabla de ayudas, pero está pensada para tests que se generan aleatoriamente. Esto permite mostrar ayudas incluso en pruebas dinámicas o aleatorias.

```
--  
-- Estructura de tabla para la tabla `ayudas_test_aleatorios`  
  
CREATE TABLE `ayudas_test_aleatorios` (  
  `id` int(11) NOT NULL,  
  `pregunta_id` int(11) NOT NULL,  
  `texto_ayuda` text NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

Relaciones:

- Cada entrada se asocia a una pregunta.

3.10 Preguntas_Test_Temáticos

Esta entidad guarda preguntas estructuradas por tema y numeradas dentro de cada test. Incluye las opciones de respuesta, la correcta, y opcionalmente una imagen asociada a la pregunta.

```
-- Estructura de tabla para la tabla `preguntas_test_tematicos`  
--  
  
CREATE TABLE `preguntas_test_tematicos` (  
    `id` int(11) NOT NULL,  
    `test_id` int(11) DEFAULT NULL,  
    `numero_pregunta` int(11) DEFAULT NULL,  
    `pregunta` text DEFAULT NULL,  
    `opcion_a` text DEFAULT NULL,  
    `opcion_b` text DEFAULT NULL,  
    `opcion_c` text DEFAULT NULL,  
    `respuesta_correcta` char(1) DEFAULT NULL,  
    `tema_id` int(11) DEFAULT NULL,  
    `imagen` varchar(255) DEFAULT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

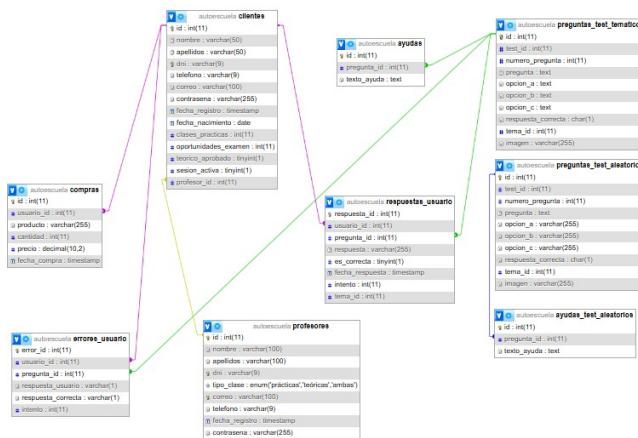
Relaciones:

- Aunque no tiene una clave foránea explícita, está relacionada lógicamente con la entidad de preguntas generales, ya que muchas preguntas temáticas son versiones organizadas de las mismas preguntas base.

3.11 ¿Cómo comprobé que funcionaban la conexión entre tablas?

Después de crear todas las tablas, hice lo siguiente:

- Inserté datos de prueba (por ejemplo, un alumno, un profesor, una compra, una pregunta, etc.).
- Hice algunas consultas para ver si las relaciones funcionaban bien.
- Probé respuestas correctas e incorrectas, y verifiqué si se guardaban en sus respectivas tablas.
- Revisé en el apartado “Diseñador” de phpMyAdmin cómo estaban relacionadas las tablas visualmente.



Todo funcionó como esperaba, lo cual me permitió confirmar que la base de datos está bien construida y lista para usarse. La única tabla que no está insertada es la tabla de contactos, que no está enlazada a ninguna otra tabla. Sólo quise reflejar las conexiones.

3.12 Como resumen del punto de la Representación de la Base de Datos.

Puedo decir que mi base de datos autoescuela ha sido implementada correctamente en MySQL usando phpMyAdmin.

El diseño que hice en papel o digital (modelo E/R) ha cobrado vida y ahora funciona como un sistema real. Todas las tablas están conectadas entre sí, y permiten almacenar y consultar información de manera ordenada, clara y eficiente.

Esta experiencia me ha servido para entender mejor cómo se pasa de la teoría a la práctica, y cómo una base de datos puede convertirse en el motor de un sistema funcional, en este caso, una autoescuela que gestiona alumnos, profesores, tests, compras y errores de forma organizada.

The screenshot shows the phpMyAdmin interface with the following details:

- Servers: 127.0.0.1:3319 - Base de datos: autoescuela
- Structure, SQL, Buscar, Generar una consulta, Exportar, Importar, Operaciones, Privilegios, Rutinas, Eventos, Disparadores
- Filtros: Que contengan la palabra: [empty]
- Tables list:
 - ayudas
 - ayudas_test_aleatorios
 - backup_errores_usuario
 - backup_resuestas_usuario
 - clientes
 - compras
 - contactos
 - errores_usuario
 - errores_test
 - preguntas_test_aleatorios
 - preguntas_test_tematicos
 - profesores
 - respuestas_usuario
 - temas
- 14 tablas
- Número de filas: 0
- Para los elementos que están marcados: [empty]

Aquí se muestran tablas como Backup_errores_usuario y respuestas_usuario, que son copias de seguridad que hice en su momento por si algo iba mal, poder recuperar los archivos con facilidad.

Diseño del Sitio Web

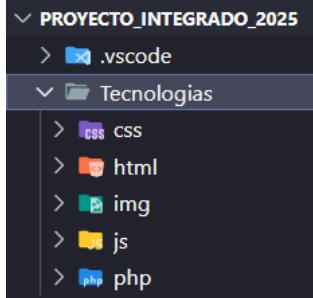
4 Introducción general al diseño del sitio web

A lo largo del desarrollo de este proyecto, he llevado a cabo el diseño y construcción de un sitio web completo para gestionar una autoescuela. Este sitio ha sido creado utilizando tecnologías web estándar: **HTML, CSS, JavaScript y PHP**, todo organizado y gestionado con **Visual Studio Code** y ejecutado en entorno local mediante **XAMPP**, conectando con la base de datos a través de **phpMyAdmin** y **MySQL**.

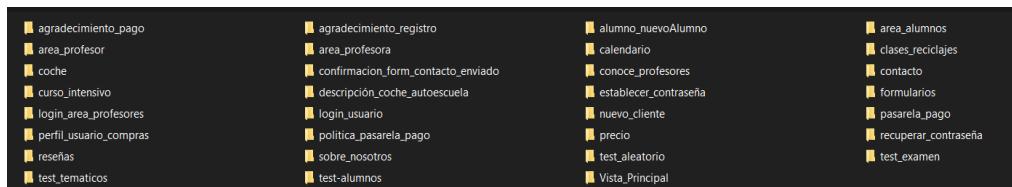
El diseño del sitio web no solo se enfoca en la estética, sino también en la **estructura de navegación, funcionalidad, lógica de usuarios y flujo de datos**. Para ello, he organizado todo mi trabajo en carpetas claras dentro de un proyecto llamado **Tecnologías**, que contiene la estructura completa de desarrollo.

(la imagen de mi TREE o Árbol de Rutas, no puedo insertarlo porque ocuparía 2 páginas, pero lo iré explicando conforme vaya avanzando).

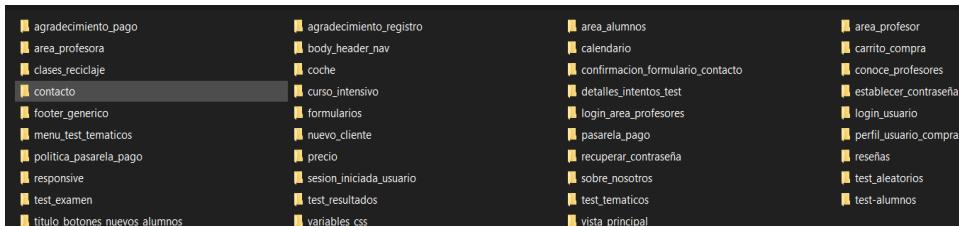
Este árbol incluye subcarpetas específicas para cada tipo de archivo:



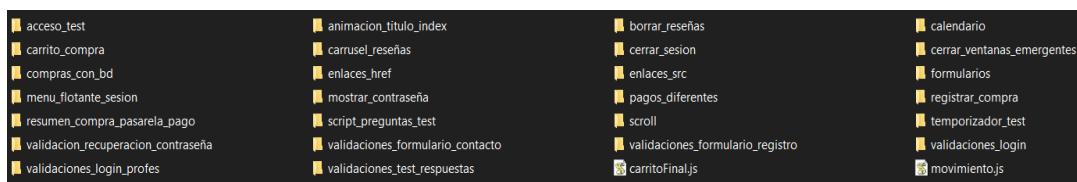
- HTML: contiene todas las vistas HTML del proyecto (una por cada pantalla del usuario). Además de algunos archivos.php que he ido actualizando dentro del html por necesidad del funcionamiento de la página.



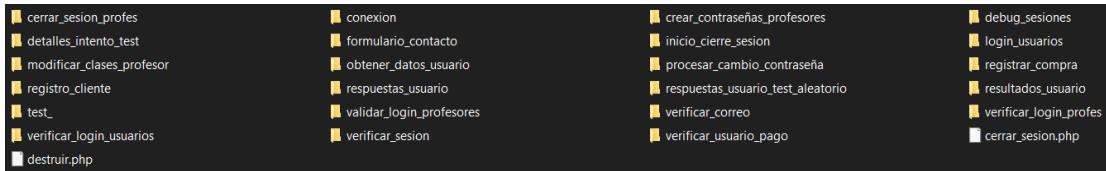
- CSS: alberga todos los estilos, distribuidos por páginas y componentes reutilizables. Sin olvidar el responsive de todo el proyecto que hace que sea capaz de verse todo el contenido en cualquier pantalla.



- JS: incluye scripts que manejan la lógica del lado del cliente: validaciones, animaciones, temporizadores, y acciones con formularios.



- PHP: contiene los archivos que procesan datos, validan accesos, conectan con la base de datos, gestionan sesiones, y más.



- IMG: agrupa las imágenes del sitio (logos, gif e imágenes en general).



Gracias a esta organización modular, me ha sido más fácil mantener el control del desarrollo, reutilizar código, separar responsabilidades (**Front-end y Back-end**) y aplicar buenas prácticas de programación web. También una cosa cabe destacar, que todo al estar almacenado por lenguaje es más fácil para llevar el control de las carpetas que tienen dentro cada uno.

Manual de Usuario y Técnico.

5 ¿Qué es el Manual de Usuario?

El **manual del usuario** es un documento que explica de forma clara y sencilla **cómo se utiliza una aplicación o un sitio web**, paso a paso. Su objetivo es **ayudar a las personas que van a usar el sistema** a entender cómo funciona, qué pueden hacer en cada pantalla y qué deben hacer si algo no funciona como esperaban.

No hace falta tener conocimientos de programación para entender este manual, porque está escrito pensando en cualquier persona: **desde un alumno que entra por primera vez, hasta un profesor que quiere ver sus clases o alumnos asignados**.

Este tipo de manual es muy útil porque:

- Ahorra tiempo al usuario.
- Reduce errores.
- Mejora la experiencia con la plataforma.
- Sirve de apoyo en caso de dudas.

Normalmente, se acompaña de capturas de pantalla donde se muestra el “camino” que debe hacer un alumno o cualquier persona que visite la autoescuela digital.

5.1 ¿Qué es el Manual Técnico?

El **manual técnico** en cambio, está pensado para explicar cómo está construida la aplicación por dentro. Este documento va dirigido principalmente a programadores, desarrolladores o personas que vayan a mantener, modificar o instalar el sistema en el futuro.

Incluye información como:

- Qué tecnologías y lenguajes se han utilizado.
- Cómo están organizados los archivos y carpetas.
- Cómo se conecta con la base de datos.
- Qué hace cada parte del código.
- Cómo acceder a la base de datos (phpMyAdmin).

El objetivo del manual técnico es ofrecer una **guía clara y detallada del funcionamiento interno del proyecto**, para que pueda ser entendido, instalado, gestionado y ampliado en el futuro por cualquier persona que trabaje con él.

5.1 ¿Cómo voy a hacer mi Manual de Usuario y Técnico?

Después de explicar qué es un manual de usuario y qué es un manual técnico, quiero contar cómo he planteado y organizado **mi propio manual**, que combina ambos en un solo bloque estructurado.

He dividido mi sitio web en distintas secciones, teniendo en cuenta los dos perfiles principales de usuarios que existen en la aplicación: **alumnos y profesores**. Cada uno tiene un tipo de acceso, funcionalidades distintas, y una experiencia diferente dentro de la plataforma, por lo que era importante diferenciarlos a la hora de documentar su uso.

A lo largo de este manual voy a ir explicando, paso a paso, qué puede hacer cada tipo de usuario en cada página de la aplicación. Para ello, incluiré **capturas de pantalla de la web**, explicaciones claras de cada sección, y los distintos caminos que puede seguir el usuario: qué pasa si introduce bien sus datos, qué ocurre si comete un error, cómo se muestra la información, y cómo interactúa con el sistema.

Pero además de mostrar cómo se usa la aplicación, también voy a explicar cómo la he desarrollado. Por eso, **he unido este manual con el manual técnico**, para que cada bloque contenga tanto la parte visual y funcional como la parte de programación y estructura interna.

En la parte técnica incluyo:

- Fragmentos reales del código utilizado (**HTML, CSS, JavaScript y PHP**).
- Explicaciones de qué hace cada parte del código.
- Cómo se conecta cada funcionalidad con la base de datos (mediante **PHP** y **MySQL**).
- Validaciones que se han aplicado para evitar errores.

Gracias a este enfoque combinado, cualquier persona podrá entender **cómo usar la aplicación y también cómo está hecha por dentro**.

A lo largo del manual, cada sección seguirá siempre la misma estructura:

- 1. Parte de usuario:** Qué puede hacer, cómo se ve, qué ocurre al interactuar.
- 2. Parte técnica:** Cómo se ha programado, fragmentos de código y explicación de la lógica.

Este método me permite ahorrar tiempo, evitar repetir contenidos y mantener la documentación ordenada, clara y útil desde ambos puntos de vista: el de uso y el de desarrollo.

5.1.1 Parte Usuario.

The screenshot shows the homepage of the Autoescuela Almansa.es website. At the top, there is a dark blue header bar with the logo 'autoescuelaalmansa@hotmail.com' and social media icons for Facebook and Instagram. Below the header, a large yellow banner features the text 'Autoescuela Almansa.es' and 'Consigue tu permiso de conducir de la mejor forma.' In the center, a dark blue call-to-action button says '¿ESTÁS INTERESADO EN SACARTE EL CARNET DE CONDUCIR?' with a yellow arrow pointing to it. Below this, there is a section titled '¿EN QUÉ NOS DIFERENCIAMOS?' containing text and a small image of a car. Further down, there is a section titled 'QUÉ DICEN SOBRE NUESTRA AUTOESCUELA*' with a small image of a person. At the bottom, there is a contact form with fields for name, email, phone, and message, along with a 'Enviar' button. The footer contains a map showing the location of the driving school in Almansa, Spain, and various links including 'Reservar cita', 'Calendario', 'Reservas', 'Sobre nosotros', and 'Contacto'.

Cuando un usuario accede por primera vez al sitio web de la autoescuela, se encuentra con una interfaz clara, moderna y organizada, diseñada para facilitar la navegación tanto a nuevos visitantes como a usuarios registrados.

En la parte superior destaca el nombre de la autoescuela y una barra de navegación con accesos directos a las secciones principales: sobre nosotros, contacto, precios, tests, profesores, etc. Esto permite moverse fácilmente por el sitio.

Los botones de “**Iniciar sesión**” y “**Registrarse**” están ubicados en dos puntos:

- Un avatar abajo a la derecha como acceso rápido.
- Botones visibles al hacer scroll, fácilmente reconocibles.

También hay enlaces a las redes sociales en la parte superior, lo que ofrece acceso a novedades, publicaciones y mejora la confianza del usuario.

Más abajo se incluye un texto informativo donde se destacan los valores, experiencia y servicios diferenciales de la autoescuela.

Luego aparece un **carrusel de reseñas** con opiniones reales de antiguos alumnos. El usuario puede:

- Navegar entre comentarios con botones.
- O esperar ~8 segundos para que el carrusel avance automáticamente.

Cerca del final, se encuentra un **formulario de contacto** para que cualquier visitante pueda enviar dudas. Algunas consultas frecuentes incluyen:

- Información sobre el carnet tipo B.
- Cursos intensivos.
- Clases prácticas sueltas o de reciclaje.

En el **footer**, se muestra la información de contacto: horarios, correo, dirección y enlace a Google Maps. También se repiten los accesos a redes sociales para que estén siempre accesibles.

Hay que destacar que todas los HTML o PHP que tienen contenido visible, contienen el mismo header, body, nav, footer, carrito ... Tanto el HTML como el CSS. De ese modo para no repetir muchas veces lo mismo, voy a ponerlo solamente al principio.

Ahora vamos a centrarnos en primer lugar en los lenguajes que he usado:

- **HTML:** Se ha utilizado para crear la estructura básica de la página, es decir, el esqueleto donde se colocan todos los elementos visibles como títulos, botones, formularios y enlaces.

- **CSS:** Se ha encargado de dar estilo a toda la página. Esto incluye colores, disposición de los elementos, botones animados y adaptabilidad a distintos tamaños de pantalla (responsive design).
- **JavaScript (JS):** Se ha usado para la parte interactiva. Por ejemplo, para validar ciertos formularios, mostrar mensajes, controlar el comportamiento del carrusel de reseñas.
- **PHP:** Para enviar el formulario de contacto, la redirección a otras páginas según sesión iniciada, o la carga de información personalizada. También será clave en secciones posteriores como login, tests o gestión de usuarios.
- **SQL:** Las consultas SQL se utilizan desde PHP para guardar los datos que el usuario introduce (por ejemplo, si completa el formulario de contacto), o para mostrar datos previamente almacenados, como reseñas de otros usuarios.

En primer lugar con el HTML, tenemos que dejar claro como se ha enlazado el CSS y el JS:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Autoescuela Almansa.es</title>

    <!-- Links de css -->
    <link rel="stylesheet" href="../../css/vista_principal/index.css">
    <link rel="stylesheet" href="../../css/body_header_nav/body_header_nav.css">
    <link rel="stylesheet" href="../../css/carrito_compra/carrito_compra.css">
    <link rel="stylesheet" href="../../css/formularios/formulario_contacto/formulario_contacto.css">
    <link rel="stylesheet" href="../../css/titulo_botones_nuevos_alumnos/titulo_botones_nuevos_alumnos.css">
    <link rel="stylesheet" href="../../css/footer_generico/footer.css">
    <link rel="stylesheet" href="../../css/sesion_iniciada_usuario/sesion_iniciada_usu.css">

    <!-- Link de las fuentes de google font -->
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link href="https://fonts.googleapis.com/css2?family=Boldonse&family=Matemasie&family=Open+Sans:ital,wght@0,300..800;1,300..800" type="text/css" rel="stylesheet">
    <link rel="shortcut icon" href="../../img/logo/logo-autoescuela.png" type="image/x-icon">

    <!-- JS -->
    <script src="../../js/enlaces_href/universal.js"></script>
    <script src="../../js/enlaces_src/imagenes.js"></script>
    <script src="../../js/carritoFinal.js"></script>
    <script src="../../js/carrusel_reseñas/carrusel_reseñas.js"></script>
    <script src="../../js/validaciones_formulario_contacto/validaciones_formulario_contacto.js"></script>
    <script src="../../js/menu_flotante_sesion/menu_flotante_sesion.js"></script>
    <script src="../../js/cerrar_sesion/cerrar_sesion.js"></script>
```

Dentro del **<head>** del documento, lo primero que se encuentran son las **metaetiquetas** y los enlaces a archivos externos, como las hojas de estilo **CSS**, la fuente “**Roboto**” desde Google Fonts, el logo de la autoescuela con **shortcut icon** y por último, los enlaces de **JavaScript**.

Algo que destaca es que las rutas para enlazar estos archivos empiezan con “**../../**”. Esto se hace así para usar **rutas relativas**, que son más cortas y prácticas que las rutas absolutas,

sobre todo cuando el proyecto se trabaja en local. Además, así se mantiene todo más ordenado dentro de las carpetas.

También he separado los CSS de los elementos comunes como el **body**, **header** y **nav**, **footer** o **el carrito** en archivos individuales. Luego, los voy incluyendo en las páginas HTML o PHP que lo necesitan.

Esto lo hago para que el código esté **más limpio y organizado**, y también para que sea más fácil encontrar lo que necesito modificar.

Gracias a esto, si algún día tengo que hacer un cambio (por ejemplo, en el menú), **solo edito un archivo y se actualiza en todo el sitio**, sin tener que cambiarlo página por página.

En cuanto al diseño, todo está hecho con **CSS puro**, sin usar ningún framework externo. Uso etiquetas y selectores básicos para darle estilo a los elementos.

En las capturas que incluyo, se puede ver cómo organizo los estilos usando **comentarios como títulos** para separar cada sección del código, como en este ejemplo de index.css.

```
/* ----- TÍTULO PRINCIPAL ----- */
.contenedor-titulos {
    background-color: #midnightblue;
    color: #fff7fa;
    padding: 3rem 2rem;
    border-radius: 12px;
    box-shadow: 0 10px 30px rgba(0, 0, 0, 0.15);
    max-width: 960px;
    margin: 60px auto 30px;
    text-align: center;
    border-left: 10px solid #eeee22;
    animation: fadeInDown 0.8s ease forwards;
}

/* ----- ANIMACIÓN PRINCIPAL ----- */
@keyframes fadeInDown {
    from {
        opacity: 0;
        transform: translateY(-50px);
    }
    75% {
        transform: translateY(10px);
    }
    to {
        opacity: 1;
        transform: translateY(0);
    }
}

/* ----- RESPONSIVA ----- */
@media (max-width: 768px) {
    .contenedor-titulos h1 {
        font-size: 2.4rem;
    }

    .contenedor-titulos h2 {
        font-size: 1.8rem;
    }

    #subtitulo-etiquetas {
        font-size: 1.5rem;
    }

    .etiqueta {
        width: 100px;
    }

    #container-etiquetas {
        flex-direction: column;
        gap: 20px;
    }
}

@media (max-width: 480px) {
    .contenedor-titulos {
        padding: 2rem;
        border-radius: 8px;
    }

    .contenedor-titulos h1 {
        font-size: 2.2rem;
    }

    .contenedor-titulos h2 {
        font-size: 1.6rem;
    }

    #subtitulo-etiquetas {
        font-size: 1.4rem;
    }

    .etiqueta {
        width: 100px;
        margin: 0 auto;
        padding: 10px;
    }

    #container-etiquetas {
        gap: 15px;
    }
}

#auto-titulo{
    color: #eeee22;
}

#motivacion{
    color: #eeee22;
}

/* ----- SUBTÍTULO DIFERENCIAS ----- */
/* Estilo para el subtítulo */
#subtitulo-diferencias {
    font-size: 2rem;
    color: #midnightblue;
    text-align: center;
    margin: 60px auto 20px;
    font-weight: bold;
    letter-spacing: 0.1em;
    position: relative;
}

#subtitulo-diferencias::after {
    content: " ";
    display: block;
    width: 80px;
    height: 4px;
    background-color: #fff7fa;
    margin: 10px auto 0;
    border-radius: 2px;
    box-shadow: 0px 0px 6px rgba(244, 249, 250, 0.6);
}

/* Contenedor del texto */
#container-texto-diferencias {
    display: flex;
    flex-direction: column;
    gap: 20px;
    background: linear-gradient(to bottom, #midnightblue, #1e3a77);
    border-radius: 15px; /* Bordes más suaves */
    padding: 40px;
    box-shadow: 0px 10px 20px rgba(0, 0, 0, 0.15);
    max-width: 960px;
    margin: 0 auto;
    transition: box-shadow 0.3s ease, background 0.3s ease;
}

#container-texto-diferencias:hover {
    box-shadow: 0px 15px 25px rgba(0, 0, 0, 0.25);
    background: linear-gradient(to bottom, #midnightblue, #283f80);
}

/* Estilo para los parrafos */
#container-texto-diferencias p {
    color: #fff7fa;
    text-align: justify;
    transition: transform 0.3s ease, color 0.3s ease, scale 0.3s ease;
    font-size: 1.2rem;
    font-weight: 400;
    margin: 10px 0;
}

#textos-diferencias-3 {
    text-align: center; /* Sobrescribe para centrar */
}

#textos-diferencias-3:hover,
#container-texto-diferencias #textos-diferencias-3 {
    color: #fff7fa;
    text-align: justify;
    transform: scale(1.05);
    color: #dcdedf;
}
```

También como dato importante del CSS, uso **Media Queries** para hacer el responsive de mi página web.

The image displays three screenshots of the Autoescuela Almansa.es website. The first screenshot shows the header with a dark blue navigation bar containing links for Carnets, Alumnos, Profesores, Calendario, Reseñas, Sobre nosotros, and Contacto. It also features social media icons for Gmail, Facebook, and Instagram. The second screenshot is a landing page with a green header asking if you're a student or want to join. It has two buttons: '¡Quiero unirme ahora!' and 'Soy alumno de esta autoescuela, ¡Accede a tu área personal ahora!'. Below this is a section titled '¿EN QUÉ NOS DIFERENCIAMOS?' with text about their teaching methods and facilities. The third screenshot shows a contact form with fields for Nombre, Apellidos, Email, Teléfono, Asunto, and Mensaje, along with 'Enviar info!', 'Resetear info!', and 'Carrito' buttons.

Seguidamente voy a mostrar la parte superior del header y el carrito y como lo he hecho. He usado etiquetas básicas de HTML en las que se muestran:

```
<!-- PARTE SUPERIOR DEL HEADER -->


<div class="email">
        
        autoescuelaalmansa@hotmail.com
    </div>

    <a href="https://www.facebook.com/antonio.almansa.397/locatives_ES" class="facebook" target="_blank">
        
    </a>

    <a href="https://www.instagram.com/autoalmansa.es/?igsh=K0CscXkIV2NkZDRn7g83D0K3D0" class="instagram" target="_blank">
        
    </a>



<!-- CARRITO DE LA COMpra -->
<div id="carrito">
    <span>Carrito</span>
    <span id="carrito-count" class="carrito-count">0</span>
</div>

<div id="carrito-menu" class="carrito-menu">
    <div id="carrito-header" class="carrito-header">
        <div>Tu carrito</div>
        <button class="close-carrito" onclick="toggleCarrito()>X</button>
    </div>
    <div id="carrito-vacio" class="carrito-vacio">
        <p>No tienes artículos en el carrito.</p>
    </div>
    <div id="carrito-contenido" class="carrito-contenido">
        <div id="carrito-item" class="carrito-item">
            <div id="carrito-titulo" class="carrito-titulo"></div>
            <div id="carrito-total" class="carrito-total"></div>
            <div>Total: <span id="total-carrito">0</span></div>
        </div>
    </div>
    <div>Botón que ahora es manejado por JavaScript -->
        <button class="btn-pago">Ir a pagar</button>
    </div>
</div>
```

```
<!-- HEADER -->
<header class="header">
    <div class="logo-contenedor">
        <div class="logo-principal" data-entlace="inicio">
            
        </div>
    </div>

    <div id="menu-navegacion">
        <div id="menu-alumnos">
            <a href="#" data-entlace="alumnos">Alumnos</a>
            <div id="submenú-alumnos">
                <a data-entlace="carnets">Carnets</a>
                <div id="submenú-coche" data-entlace="coche">Coche - B</div>
                <div id="submenú-precio" data-entlace="precio">Precio - B</div>
                <div id="submenú-intensivos" data-entlace="cursos_intensivos">Cursos Intensivos</div>
                <div id="submenú-reciclaje" data-entlace="clases_reciclaje">Clases de Reciclaje</div>
            </div>
        </div>
        <div id="menu-profesores">
            <a href="#" data-entlace="profesores">Profesores</a>
            <div id="submenú-profesores">
                <a data-entlace="profesores">Conoce a tus profesores</a>
                <a data-entlace="login_profesores">Área profesores</a>
            </div>
        </div>
        <div id="menu-calendario">
            <a href="#" data-entlace="calendario">Calendario</a>
        </div>
        <div id="menu-reseñas">
            <a href="#" data-entlace="reseñas">Reseñas</a>
        </div>
        <div id="menu-sobrenosotros">
            <a href="#" data-entlace="sobre_nosotros">Sobre nosotros</a>
        </div>
        <div id="menu-contacto">
            <a href="#" data-entlace="contacto">Contacto</a>
        </div>
    </div>
</header>
```

En el código HTML he utilizado **etiquetas básicas y comunes** para estructurar el contenido de la página web de forma organizada. Algunas de las etiquetas más destacadas que se utilizan son:

- **<div>**: Se usa como contenedor para agrupar secciones del contenido. Por ejemplo, se utilizan **<div>** para separar la parte del encabezado y el carrito de la compra.
- **<a>**: Esta etiqueta sirve para crear enlaces, como los que redirigen a las redes sociales (Facebook e Instagram) o enlaces de contacto por correo electrónico.
- ****: Permite insertar imágenes en la página, como los iconos del correo y redes sociales. Se le asignan atributos como width y height para definir su tamaño.
- Y más etiquetas que podrán ir viendo durante la documentación.

Además, he utilizado **atributos como class, id, onclick o target** para dar estilos (CSS) o interactividad (JavaScript) a los elementos.

En esta parte tengo que explicar que hace la función **toggleCarrito()**. Esta función está metida dentro un archivo JavaScript, llamado **carritoFinal.js**

Concretamente esta función busca un elemento con id "**carrito-menu**" en el **HTML** y muestra u oculta la clase **CSS "visible"** cada vez que se llama.

```
// ----- MOSTRAR / OCULTAR CARRITO -----
function toggleCarrito() {
  const menu = document.getElementById("carrito-menu");
  if (menu) menu.classList.toggle("visible");
}

/* Mostrar carrito cuando esté visible (añade esta clase desde JS) */
.carrito-menu.visible {
  opacity: 1;
  visibility: visible;
}
```

También podemos ver como hago el llamamiento de los diferentes HTML con data-enlace y data-src para las imágenes.

¿Por qué lo he hecho así? Para organizar mejor mi código, puse todas las rutas de mi sitio en un solo objeto en JavaScript. Así, los enlaces **HTML** solo tienen una palabra clave con data-enlace, y el script se encarga de redirigir a la página correcta. Esto es útil porque si una ruta cambia, no tengo que ir a todos los archivos a cambiarla, solo la actualizo una vez.

También me permite controlar qué pasa cuando se hace clic en un enlace, por ejemplo, mostrar una animación o evitar errores si la ruta no existe.

```
document.addEventListener('DOMContentLoaded', () => {
  // Obtengo todos los enlaces con el atributo data-enlace
  const enlaces = document.querySelectorAll('[data-enlace]');

  // Configura las rutas en un objeto (rutas relativas ajustadas)
  const rutas = {
    inicio: '../../../../../html/vista_Principal/index.php',
    sobre_nosotros: '../../../../../html/vista_Nosotros/sobre_nosotros.html',
    precios: '../../../../../html/precios/precios.php',
    politica_pasarela_pago: '../../../../../html/politica_pasarela_pago/politica_pasarela_pago.html',
    pasarela_pago: '../../../../../html/pasarela_pago/pasarela_pago.html',
    contacto: '../../../../../html/contacto/contacto.html',
    logo_reseñas_google: '../../../../../img/logo/reviews-google.png',
    cursos_intensivos: '../../../../../html/cursos_intensivos/courses_intensivo.html',
    clases_reciclaje: '../../../../../html/clases_reciclaje/clases_reciclaje.html',
    profesores: '../../../../../html/profesores/profesores.html',
    calendario: '../../../../../html/calendario/calendario.php',
    formulario_registro_nuevo_cliente: '../../../../../html/formularios/formulario_registro_cliente/formulario_registro_cliente.html',
    agradecimientos: '../../../../../html/agradecimientos/paginas_agradecimientos_pago.html',
    login_usuario: '../../../../../html/login_usuario/login_usuario.html',
    login_pago: '../../../../../html/alumno_nuevo/usuario/login_nuevo/usuario.php',
    compras_usuario: '../../../../../html/perfil_usuario/compras/perfil_usuario_compras.html',
    resultados: '../../../../../php/resultados_usuario/resultados.php',
    test_electoralio: '../../../../../html/test_electoralio/menu_test_electoralio.html',
    test_examenes: '../../../../../html/test_examenes/menu_examenes.html',
    login_profesores: '../../../../../html/login_area_profesores/login_area_profesores.php',
  };

  // Asocia un evento a cada enlace
  enlaces.forEach(enlace => {
    enlace.addEventListener('click', (e) => {
      e.preventDefault(); // Prevenir la acción predeterminada del enlace
      const destino = enlace.getAttribute('data-enlace');

      // Verifica si existe una ruta definida para este enlace
      if (rutas[destino]) {
        window.location.href = rutas[destino];
      } else {
        console.warn(`Ruta no definida para: ${destino}`);
      }
    });
  });
});
```

```
document.addEventListener('DOMContentLoaded', () => {
  const imagenes = document.querySelectorAll('[data-src]');

  const rutas = [
    reseñas_google: '../../../../../img/logo/629383e30fb025780ee2970.png',
    coche_autoescuela_grande: '../../../../../img/logo/citroen-auto.png',
    coche_pequeño: '../../../../../img/logo/coche_blanco_auto.png',
    fb: '../../../../../img/logo/facebook.png',
    gmail: '../../../../../img/logo/Gmail_Icon.original.png',
    instagram: '../../../../../img/logo/instagram.png',
    logo_autoescuela: '../../../../../img/logo/logo-autoescuela.png',
    estrellas_google: '../../../../../img/logo/reviews-google.png',
    profesor: '../../../../../img/logo/profesor_auto.png',
    profesora: '../../../../../img/logo/profesora_auto.png',
    gif_estadisticas: '../../../../../img/gif/estadisticas.gif',
    gif_compras: '../../../../../img/gif/compras.gif',
    gif_test: '../../../../../img/gif/test.gif',
    test_2_gif: '../../../../../img/gif/test_2.gif',
  ];

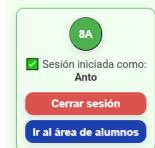
  if (imagenes.length === 0) {
    console.warn('No se encontraron elementos con el atributo [data-src].');
  } else {
    imagenes.forEach(imagen => {
      const destino = imagen.getAttribute('data-src');
      if (destino && rutas[destino]) {
        imagen.src = rutas[destino];
        console.log(`Ruta asignada: ${rutas[destino]} para ${destino}`);
      } else if (!destino) {
        console.error(`El atributo data-src está ausente o vacío en un elemento.`);
      } else {
        console.warn(`Ruta no definida para: ${destino}`);
      }
    });
  }
});
```

Un poco mas abajo y dentro del header, encontramos el inicio y cierre de sesión de los alumnos.

Consiste en un archivo php que muestra el avatar si la sesión está cerrada como se puede ver en las capturas de pantalla y si está iniciada la sesión, muestra un botón con las iniciales de su nombre y su ID (por ejemplo: "12JP").



Además con la sesión iniciada tiene dos botones, uno que va directamente a cerrar sesión y otro que va al área del usuario.



Para realizar esto, uso **la variable \$_SESSION['usuario']** para saber si hay un usuario conectado.

- Usa **\$_GET['cerrado']** para saber si el usuario cerró sesión hace un momento.
- El botón del "avatar" muestra las iniciales del nombre del usuario (por ejemplo, Juan Pérez → JP).
- Si el nombre viene como varias palabras, las junta y toma solo la primera letra de cada una.
- También tiene un formulario que lleva al archivo **logout.php**, que es el encargado de cerrar la sesión.

Dentro del **logout.php**, encontramos:

```
<?php
session_start(); // Iniciar sesión

// Ajustamos la conexión con la base de datos
$rutaConexion = __DIR__ . '/../conexion/conexion.php';

if (!file_exists($rutaConexion)) {
    require_once($rutaConexion);
} else {
    die("⚠ Error: No se encontró el archivo de conexión en " . $rutaConexion);
}

if (isset($_SESSION['usuario']) && isset($_SESSION['email'])) {
    $usuario = mysqli_real_escape_string($conexion, $_SESSION['usuario']);
    $email = $_SESSION['email'];

    // **Cerrara la sesión en la base de datos**
    $query = "UPDATE clientes SET sesion_activa = 0 WHERE correo = '$usuario'";
    if (!mysqli_query($conexion, $query)) {
        die("⚠ Error al cerrar sesión en la base de datos: " . mysqli_error($conexion));
    } else {
        echo "<p>✅ Sesión cerrada correctamente en la base de datos.</p>";
    }
} else {
    echo "<p>⚠ No hay un usuario logueado en PHP, pero verificamos en la base de datos...</p>";
}

// **Buscar usuarios con sesión activa en la base de datos**
$stmt = $conexion->prepare("SELECT correo FROM clientes WHERE sesion_activa = 1");
$stmt->execute();
$result = $stmt->get_result();

if ($result->num_rows > 0) {
    while ($row = $result->fetch_assoc()) {
        $usuario_db = $row['correo'];
        $query = "UPDATE clientes SET sesion_activa = 0 WHERE correo = '$usuario_db'";
        mysqli_query($conexion, $query);
        echo "<p>✅ Se cerró la sesión para el usuario: $usuario_db</p>";
    }
} else {
    echo "<p><strong>No hay sesiones activas en la base de datos.</strong></p>";
}

// **Limpiar todas las variables de sesión**
session_unset();
$_SESSION = [];

// **Eliminar la cookie PHPSESSID**
if (!isset($_COOKIE[session_name()])) {
    setcookie(session_name(), '', time() - 3600, '/', '', false, true);
}

// **Destruir la sesión**
session_destroy();

// **Regenerar ID de sesión**
session_start();
session_regenerate_id(true);

// **Redirigir a index.php**
header("Location: http://localhost/Proyecto_Integrado_2025/Tecnologias/html/Vista_Principal/index.php?cerrado=1");
exit;
?>
```

El cierre de sesión de un usuario en **PHP** y en la base de datos. Primero, inicia la sesión con **session_start()** y carga los datos de conexión. Si hay un usuario activo, actualiza la tabla **clientes** para marcar su sesión como cerrada (**sesion_activa = 0**). Si no hay sesión iniciada, busca y cierra cualquier sesión activa existente.

Luego, elimina todas las variables de sesión (**session_unset()**), borra la cookie de sesión y destruye la sesión en el servidor (**session_destroy()**). Para mayor seguridad, regenera un nuevo **ID** de sesión (**session_regenerate_id(true)**).

Finalmente, redirige al usuario a la página principal (**index.php**) con un mensaje de sesión cerrada.

Luego, para iniciar la sesión tenemos el inicio tenemos el **login_usuario.php** y el **inicio.php** con sus validaciones en JavaScript, que se explicaran más adelante.

Para que en el **index.php** se siga manteniendo la sesión tanto del profesor como alumno, tenemos:

Este fragmento de código **recupera y limpia un mensaje de sesión en PHP**

```
<?php
session_start();

$mensaje = '';
if (isset($_SESSION['mensaje'])) {
    $mensaje = $_SESSION['mensaje'];
    unset($_SESSION['mensaje']);
}
?>
```

Tal y encontramos los botones de iniciar sesión o inscribirte en la autoescuela.



Uno te mandará al **formulario de registro del cliente** y el otro al **login del usuario**:



Vamos a empezar por el **formulario de registro del cliente.php**:

En este caso contamos un formulario básico en HTML precedido por una etiqueta <section> a modo de container, que hará conexión con la base de datos, gracias a la **conexión.php**.

```
<?php
$host = "localhost:3310";
$usuario = "root";
$contrasena = "";
$base_datos = "autoescuela";

$conexion = new mysqli($host, $usuario, $contrasena, $base_datos);

// Verificar la conexión
if ($conexion->connect_error) {
    die("Error de conexión: " . $conexion->connect_error);
}

// Establecer codificación
$conexion->set_charset("utf8");

?>
```

```
<section id="registro-cliente">
<div><div><div>
<form id="formulario" action=".../.../php/registro_cliente/registrar_cliente.php" method="post">
<!-- Token CSRF oculto : Este código genera un campo oculto en el Formulario que incluye un token CSRF almacenado en la sesión para proteger contra ataques de falsificación de solicitudes entre sitios (CSRF). -->
<input type="hidden" name="csrf_token" value=<?php echo $_SESSION['csrf_token']; ?>>
```

...

```
<div>
<label for="nombre">Nombre:</label>
<input id="nombre" name="nombre" type="text" placeholder="Tu nombre" required>
<div><small>Error-nombre</small><br><small>Error-mensaje</small></div>
</div>
```

```
<div>
<label for="apellidos">Apellidos:</label>
<input id="apellidos" name="apellidos" type="text" placeholder="Tus apellidos" required>
<div><small>Error-apellidos</small><br><small>Error-mensaje</small></div>
</div>
```

```
<div>
<label for="fecha_nacimiento">Fecha de Nacimiento:</label>
<input id="fecha_nacimiento" name="Fecha_nacimiento" type="date" required oninvalid="return false;" oninput="setCustomValidity('Formato incorrecto')" onfocus="setCustomValidity('')"/>
<div><small>Error-fecha_nacimiento</small><br><small>Error-mensaje</small></div>
</div>
```

```
<div>
<label for="dni">DNI:</label>
<input id="dni" name="dni" type="text" pattern="[0-9]{8}[A-Z-a-z]{1}" placeholder="12345678A" required>
<div><small>Error-dni</small><br><small>Error-mensaje</small></div>
</div>
```

```
<div>
<label for="telefono">Teléfono:</label>
<input id="telefono" name="telefono" type="tel" placeholder="123456789" pattern="(0-9){9}" required>
<div><small>Error-telefono</small><br><small>Error-mensaje</small></div>
</div>
```

```
<div>
<small>Por favor! Corrige el formulario y vuelve a intentar.</small>
<input id="correo" name="correo" type="email" placeholder="ti@mail.com" required>
<div><small>Error-correo</small><br><small>Error-mensaje</small></div>
</div>
```

```
<div>
<label for="contraseña">Contraseña:</label>
<input id="contraseña" name="contraseña" type="password" placeholder="Tu contraseña" required>
<div><small>Error-contraseña</small><br><small>Error-mensaje</small></div>
<script>
function mostrarContraseña() {
    let contraseña = document.getElementById("contraseña");
    if(contraseña.type == "password") { contraseña.type = "text"; }
    else { contraseña.type = "password"; }
}
</script>
<div><small>Error-contraseña</small><br><small>Error-mensaje</small></div>
</div>
```

```
<div class="contenedor-botones">
<button type="reset" id="reset-datos"><small>Borrar campos</small></button>
<button type="submit" id="enviar-datos">Registrarse</button>
</div>
</div>
</div>
</div>
</div>
```

```
<?php
session_start();
$_SESSION['csrf_token'] = bin2hex(random_bytes(32));
?>
```

```
root {
    --ancho-formulario: 90%;
    --margen-formulario: 60px auto 100px;
    --padding-formulario: 24px;
    --fondo-formulario: #fff7fafd;
    --borde-formulario: 1.5px solid #ddeeaf;
    --radio-formulario: 14px;
    --sombra-formulario: 0 6px 24px rgba(0, 0, 0, 0.10);
    --color-titulo: #midnightblue;
    --tamaño-titulo: 28px;
    --espacio-titulo: 30px;
    --espacio-entre-campos: 22px;
    --borde-campo: 1.5px solid #bfcbe0;
    --borde-campo-focus: 2px solid #eeee22;
    --radio-campo: 7px;
    --radio-campo-focus: 9px;
    --tamaño-fuente-campo: 16px;
    --color-texto-campo: #midnightblue;
    --color-fondo-campo: #fff;
    --color-placeholder: #a9a9a9;
    --transicion-campo: 0.2s cubic-bezier(.4,0,.2,1);
    --escalado-campo-focus: 1.0;
    --padding-botón: 10px 22px;
    --tamaño-botón: 35px;
    --tamaño-fuente-botón: 15px;
    --fondo-botón-enviado: #midnightblue;
    --color-botón-enviado: #eeee22;
    --fondo-botón-reset: #fff;
    --color-botón-reset: #midnightblue;
    --transicion-botón: 0.22s cubic-bezier(.4,0,.2,1);
    --escalado-botón: 1.05;
    --color-error: #c00;
    --color-botón-mostrar: #eeee22;
    --color-botón-mostrar-texto: #midnightblue;
}
```

Este sistema permite que un usuario se registre completando un formulario web. Antes de enviar los datos al servidor, se verifica que estén bien escritos y que cumplan con ciertas reglas, tanto para evitar errores como para asegurar que la información es válida. Esta verificación se hace desde el navegador usando JavaScript, y una vez que pasa esta revisión, los datos se envían al servidor, donde se revisan de nuevo con PHP antes de guardarse en la base de datos MySQL. El diseño del formulario está hecho en HTML y la lógica del lado del cliente y del servidor se maneja con JavaScript y PHP, respectivamente.

El formulario incluye los siguientes campos: nombre, apellidos, fecha de nacimiento, DNI, teléfono, correo electrónico y contraseña. También hay un campo oculto con un token de seguridad (**CSRF**) que protege el formulario de envíos no autorizados. Además, hay un botón para mostrar u ocultar la contraseña y botones para enviar o limpiar el formulario.

En la parte de **JavaScript**, cada campo se valida paso a paso antes de permitir que se envíe. Por ejemplo, el campo **nombre** solo acepta letras (mayúsculas y minúsculas) y espacios, y no permite números ni símbolos. También se comprueba que el nombre no esté vacío y que contenga al menos una vocal, para evitar registros con datos falsos. Lo mismo se aplica al campo de **apellidos**. Para el **DNI**, se revisa que tenga exactamente 8 números seguidos de una sola letra al final, como "12345678A", y que no haya caracteres adicionales. El **teléfono** debe contener exactamente 9 cifras numéricas, sin letras ni espacios, y sin signos. En el campo de **correo electrónico**, se verifica que tenga un formato correcto con una arroba y un dominio, y que el dominio sea uno de los permitidos: **@gmail.com**, **@hotmail.com** o **@outlook.com**. Si no cumple, se muestra un mensaje de error y no se permite continuar.

En el caso de la **contraseña**, JavaScript se asegura de que cumpla con varios requisitos de seguridad: debe tener al menos 8 caracteres en total, contener al menos una letra mayúscula,

un número y un símbolo especial (como @, !, #, etc.). Esto se hace usando expresiones regulares que analizan el contenido de la contraseña y comprueban si cumple con todas esas condiciones. Si no es así, se le muestra al usuario un mensaje claro indicando qué falta.

```

document.addEventListener("DOMContentLoaded", () => {
  const form = document.getElementById("Form-datos");
  const fechaInput = document.getElementById("fecha_nacimiento");

  // Calcular la fecha mínima permitida (18 años atrás)
  const hoy = new Date();
  const edadMinima = new Date(hoy.getFullYear() - 18, hoy.getMonth(), hoy.getDate());
  fechaInput.setAttribute("max", edadMinima.toISOString().split("T")[0]);

  form.addEventListener("submit", function (e) {
    let errores = false;

    // Limpiar mensajes anteriores
    document.querySelectorAll(".error-message").forEach(p => p.textContent = "");

    // Obtener valores
    const nombre = form.nombre.value.trim();
    const apellido = form.apellidos.value.trim();
    const dni = form.dni.value.trim();
    const telefono = form.telefono.value.trim();
    const correo = form.correo.value.trim();
    const contraseña = form.contraseña.value;
    const fechaNacimiento = new Date(form.fecha_nacimiento.value);

    // Validar nombre real
    if (nombre === "") {
      mostrarError("nombre", "El nombre es obligatorio.");
      errores = true;
    } else if (!esNombreValido(nombre)) {
      mostrarError("nombre", "Introduce un nombre válido.");
      errores = true;
    }

    // Validar apellidos reales
    if (apellido === "") {
      mostrarError("apellidos", "Los apellidos son obligatorios.");
      errores = true;
    } else if (!esNombreValido(apellido)) {
      mostrarError("apellidos", "Introduce apellidos válidos.");
      errores = true;
    }

    // Validar DNI (8 números + 1 letra)
    if (!/\^([A-Z]{1}[0-9]{7})$/.test(dni)) {
      mostrarError("dni", "El DNI debe tener 8 números y una letra.");
      errores = true;
    }

    // Validar teléfono (9 dígitos)
    if (!/\^([0-9]{9})$/.test(telefono)) {
      mostrarError("telefono", "El teléfono debe tener 9 dígitos.");
      errores = true;
    }
  });

  // Valida correo electrónico
  const correoInput = document.getElementById("correo");
  const correoRegex = /^[^\s@]+@[^\s@]+\.[^\s@]{2,}$/.test(correo);
  if (!correoRegex) {
    correoInput.setCustomValidity("El correo electrónico no es válido.");
  } else {
    correoInput.setCustomValidity("");
  }
});

// Función para validar el nombre
function esNombreValido(nombre) {
  // Eliminar espacios extra y convertir a minúsculas
  nombre = nombre.trim().toLowerCase();

  // Verificar longitud (mínimo 2 caracteres, máximo 40)
  if (nombre.length < 2 || nombre.length > 40) return false;

  // Verificar si contiene solo letras y espacios
  if (/^([a-zA-Záéíúññ\s]+)$/.test(nombre)) return false;

  // Verificar que tenga al menos una vocal
  if (/^([aeiouáéíúññ]+)$/.test(nombre)) return false;

  // Evitar secuencias de consonantes sin vocales
  if (/^(?![aeiouáéíúññ][aeiouáéíúññ][aeiouáéíúññ][aeiouáéíúññ])[aeiouáéíúññ]+$/i.test(nombre)) return false;

  return true;
}

function mostrarError(idCampo, mensaje) {
  document.getElementById(`error-${idCampo}`).textContent = mensaje;
}

```

Otro punto importante es la validación de la **edad**. A partir de la **fecha de nacimiento** que introduce el usuario, JavaScript calcula su edad y verifica que tenga al menos 18 años. Si es menor de edad, se muestra una alerta y se impide enviar el formulario. Esta comprobación se hace restando el año actual con el año de nacimiento y ajustando el cálculo dependiendo del mes y día actuales.

Además, el formulario cuenta con un botón que permite **mostrar u ocultar la contraseña** escrita. Este botón funciona con JavaScript y lo que hace es cambiar el tipo del campo de "password" a "text" cuando el usuario desea ver lo que escribió, y lo vuelve a cambiar a "password" cuando quiere ocultarla otra vez.

```

<script>
function mostrarContraseña() {
  const input = document.getElementById("contraseña");
  input.type = input.type === "password" ? "text" : "password";
}
</script>

```

Esto es útil para evitar errores al escribir y también da más comodidad al usuario. Generalmente este botón aparece como un ícono o con un texto como "Mostrar" / "Ocultar", y cambia su estado cada vez que el usuario hace clic.

Una vez que todas estas comprobaciones se realizan y todo está correcto, se permite enviar el formulario al servidor. Si algún campo no pasa la validación, no se envía y se muestran mensajes de error justo debajo o al lado del campo que está mal, para que el usuario sepa qué corregir.

En el servidor, PHP se encarga de repetir las validaciones para asegurarse de que no se ha evitado ninguna protección. También limpia los datos para quitar espacios y posibles códigos dañinos. Luego revisa si el correo o el DNI ya están registrados. Si no lo están, guarda los datos y cifra la contraseña con **password_hash()** antes de guardarla en la base de datos. Además, se crea una cookie segura para identificar al cliente y se redirige al usuario a una página de confirmación.

```
<?php
session_start();

// Incluir el archivo de conexión a la base de datos
require_once './conexion/conexion.php';

// Generar un CSRF token para el formulario : Esto es para la seguridad. Un token CSRF protege contra ataques en los
// que un atacante intenta enviar solicitudes maliciosas en nombre de un usuario legítimo.
$_SESSION['csrf_token'] = bin2hex(random_bytes(32));

// Recoger los datos del formulario y validárselos
$nombre = isset($_POST['nombre']) ? htmlspecialchars(trim($_POST['nombre'])) : '';
$apellidos = isset($_POST['apellidos']) ? htmlspecialchars(trim($_POST['apellidos'])) : '';
$dni = isset($_POST['dni']) ? strtroupper(htmlspecialchars(trim($_POST['dni']))) : '';
$telefono = isset($_POST['telefono']) ? htmlspecialchars(trim($_POST['telefono'])) : '';
$correo = isset($_POST['correo']) ? htmlspecialchars(trim($_POST['correo'])) : '';
$contraseña = isset($_POST['contraseña']) ? $_POST['contraseña'] : '';
$fecha_nacimiento = isset($_POST['fecha_nacimiento']) ? $_POST['fecha_nacimiento'] : '';

// Validaciones
$errores = [];

// Validación de nombre (debe tener al menos 1 carácter)
if (empty($nombre)) {
    $errores['nombre'] = "El nombre es obligatorio.";
}

// Validación de apellidos (debe tener al menos 1 carácter)
if (empty($apellidos)) {
    $errores['apellidos'] = "Los apellidos son obligatorios.";
}

// Validación de DNI (debe tener 8 dígitos + una letra)
if (!preg_match('/^(0|9){3}[A-Z]{1}[0|9]{3}[A-Z]{1}$', $dni)) {
    $errores['dni'] = "El DNI no tiene el formato válido (8 números seguidos de una letra).";
} else {
    // Verificar si el DNI ya está registrado en la base de datos
    $query = "SELECT * FROM clientes WHERE dni = ?";
    $stmt = $conexion->prepare($query);
    $stmt->bind_param("s", $dni);
    $stmt->execute();
    $result = $stmt->get_result();

    if ($result->num_rows > 0) {
        $errores['dni'] = "El DNI ya está registrado.";
    }
}

// Validación de teléfono (debe ser un número de 9 dígitos)
if (!preg_match('/^([0-9]{9})$/i', $telefono)) {
    $errores['telefono'] = "El teléfono debe tener 9 dígitos.";
}

// Verificar si el correo ya está registrado en la base de datos
$query = "SELECT * FROM clientes WHERE correo = ?";
$stmt = $conexion->prepare($query);
$stmt->bind_param("s", $correo);
$stmt->execute();
$result = $stmt->get_result();

if ($result->num_rows > 0) {
    $errores['correo'] = "El correo electrónico ya es válido.";
}

// Verificar si el correo ya está registrado en la base de datos
$query = "SELECT * FROM clientes WHERE correo = ?";
$stmt = $conexion->prepare($query);
$stmt->bind_param("s", $correo);
$stmt->execute();
$result = $stmt->get_result();

if ($result->num_rows > 0) {
    $errores['correo'] = "El correo ya está registrado.";
}

// Valificación de contraseña (debe tener al menos 8 caracteres)
if (empty($contraseña)) {
    $errores['contraseña'] = "La contraseña debe tener al menos 8 caracteres.";
}

// Validación en fecha de nacimiento (debe ser válida)
if (empty($fecha_nacimiento)) {
    $errores['fecha_nacimiento'] = "La fecha de nacimiento es obligatoria.";
}

// Si hay errores, mostrarlos en el formulario
if (empty($errores)) {
    // No hacer nada, los errores se mostrarán dentro del formulario
} else {
    // Si no hay errores, procesar el registro

    // Hashear la contraseña
    $contraseña_hash = password_hash($contraseña, PASSWORD_DEFAULT);

    // Preparar la consulta SQL de inserción
    $query = "INSERT INTO clientes (nombre, apellidos, dni, telefono, correo, contraseña, fecha_nacimiento, profesor_id) VALUES (?, ?, ?, ?, ?, ?, ?, ?)";
    $stmt = $conexion->prepare($query);
    $stmt->bind_param("ssssssss", $nombre, $apellidos, $dni, $telefono, $correo, $contraseña_hash, $fecha_nacimiento);

    // Ejecutar la consulta
    if ($stmt->execute() === false) { // Si la consulta se ejecuta correctamente, se inserta el nuevo cliente
        // Almacenar el ID del cliente en una cookie (opcional), por ejemplo para mantener la sesión
        setcookie("cliente_id", $stmt->insert_id, time() + 3600, '/', '', false, true); // Cookie segura, expira en 1 hora

        // Redirigir a la página de agradecimiento
        header("Location: /Proyecto_Integrado_2025/Tecnologias/html/agradecimiento_registro/agradecimiento_registro.html");
        exit();
    } else {
        // Mostrar un mensaje de éxito
        echo "Error al registrar: " . $stmt->error;
    }

    $stmt->close(); // Cierre la consulta.
}
}
```

En cuanto a la seguridad, el formulario usa **tokens CSRF** para evitar que alguien lo use desde otro sitio. Las contraseñas se cifran, se validan en el navegador y en el servidor, y las consultas a la base de datos están protegidas contra inyecciones **SQL**. Todo esto garantiza que el sistema sea seguro, confiable y fácil de usar.

También tenemos que saber es que por defecto TODOS los clientes, tendrán asignado al profesor 2 que es María López, encargada de la teórica.

Enrique	García	Pérez	32145217Y	698012347	enriqueGar@gmail.com	S2yS10S/VlekSKDnFJanZpDl0B44em2n056BRK_zlscOa4on...	2025-06-05	13:35:50	2005-06-04	0	2	0	0	2
---------	--------	-------	-----------	-----------	----------------------	---	------------	----------	------------	---	---	---	---	---

No podemos olvidar que los **CSS** de cuentan con sus variables para resumir y no repetir código. Al igual que los enlaces de JavaScript, todos tienen el de las rutas universales y el de las imágenes expliqué anteriormente.

En la página de confirmación, te agradece por haberte registrado y tiene varias opciones:



```
<!-- ----- AGRADECIMIENTO DEL REGISTRO ----- -->
<h1>¡Gracias por registrarte en Autoescuela Almansa.es!</h1>
<div class="agradecimiento-contenedor">
  <p>Estamos encantados de darte la bienvenida a nuestra comunidad. Tu registro ha sido exitoso y ahora formas parte de una autoescuela comprometida con tu aprendizaje y éxito.</p>
  <p>En Autoescuela Almansa.es, nos esforzamos por ofrecerte la mejor experiencia de aprendizaje, con recursos de calidad, profesores altamente capacitados y un ambiente amigable.</p>
  <p>Si tienes alguna pregunta o necesitas ayuda, no dudes en ponerte en contacto con nosotros a través de nuestro correo electrónico <a href="mailto:autoescuelaalmansa@hotmail.com">autoescuelaalmansa@hotmail.com</a> o nuestras redes sociales.</p>
  <p>¡Estamos aquí para ayudarte a alcanzar tus metas y obtener tu carnet de conducir!</p>
  <a data-enlace="inicio" class="btn-volver-inicio">Volver al inicio</a>
  <a data-enlace="test" class="btn-test">Ir a los tests</a>
  <a data-enlace="contacto" class="btn-contacto">Ir al contacto</a>
</div>
```

```
/* agradecimiento_registro.css */
:root {
  --color-fondo: #f0f0f0;
  --color-borde: #midnightblue;
  --radio-borde: 10px;
  --relleno: 30px;
  --ancho-maximo: 600px;
  --color-texto: #midnightblue;
  --tamaño-fuente: 1em;
  --tamaño-parrafos: 16px;
  --altura-linea-parrafos: 1.6;
  --color-texto-enlace: #eeeeee;
  --color-fondo-enlace: #midnightblue;
  --color-botón-hover-enlace: #eeee22;
  --color-texto-hover-enlace: #midnightblue;
  --radio-borde-enlace: 5px;
  --duracion-transicion: 0.3s;
}
```

Aquí directamente podemos mandar un correo al email personal de la autoescuela, irnos a la parte de contacto que mostraremos más adelante, volver al inicio para hacer login en nuestro usuario o ir a los test. Cabe destacar que por el simple hecho de registrarte, no tienes acceso a los Test, para esto, tenemos que pagarlos.

Accedemos al Login de Usuarios o Alumnos desde el inicio. Y nos encontramos con un formulario, que te pregunta por tu correo y contraseña que has puesto en el registro previo y que ya están almacenadas en la base de datos, gracias a la conexión con el servidor de **PHP** con **XAMPP** propiamente dicha.



```
<!-- ----- FORMULARIO LOGIN ----- -->
<h2 id="titulo-login-usuario">Login de usuario</h2>
<form action="../../php/login_usuarios/login_usuario.php" method="POST" id="formulario-login">
  <div id="campo">
    <label for="usuario">Usuario:</label>
    <input type="text" name="usuario" id="usuario" placeholder="Introduce tu usuario" required><br><br>
    <xp class="mensaje-error" id="error-usuario"></p>
  </div>

  <div id="campo">
    <label for="contrasena">Contraseña:</label>
    <input type="password" name="contrasena" id="contrasena" placeholder="Introduce tu contraseña" required>
    <button type="button" id="mostrar-contraseña">>Mostrar</button>
    <xp class="mensaje-error" id="error-contrasena"></p>
  </div>

  <label>
    <input type="checkbox" name="recordar" value="1" checked> Recordarme
  </label>
  <input type="submit" value="Iniciar sesión">
</form>

<p><a href="../../html/recuperar_contraseña/recuperar_contraseña.php">Olvidaste tu contraseña?</a></p>
<p><a href="../../html/crear-cuenta">No tienes una cuenta?</a> <a href="#">Crear cuenta</a></p>
```

```
/* root */
--color-principal: #005F73;
--color-secundario: #40D2B0;
--color-fondo-formulario: #fff;
--color-borde: 1px solid #ccc;
--color-texto: #333;
--color-botón: #a9a9a9;
--color-botón-hover: #007F89;
--radio-borde: 10px;
--espaciado: 15px;
--sombra: 0 4px 10px rgba(0, 0, 0, 0.1);
--fuente: 'Reboto', sans-serif;
```

El formulario de login permite que los usuarios registrados puedan iniciar sesión en su cuenta.

Está creado con **HTML**, **JavaScript** y **PHP**, y se conecta a la base de datos de **Clients** para comprobar si los datos introducidos son correctos. En este formulario, el usuario debe escribir su correo electrónico (que actúa como nombre de usuario) y su contraseña. También puede marcar una casilla para que el sistema lo recuerde en futuras visitas.

Además, cuenta con un botón para mostrar u ocultar la contraseña, lo cual ayuda a evitar errores al escribirla. Si el usuario ha olvidado su contraseña, tiene un enlace disponible para recuperarla.

Para recuperar y cambiar la contraseña, contamos primero con el formulario **recuperar_contraseña.php**, acompañado por un archivo **PHP** que se encarga de la parte lógica. Este archivo se conecta a la base de datos y revisa si el correo electrónico que introduce el usuario está registrado en alguna de las dos tablas: **clientes** o **profesores**. Si el correo se encuentra en alguna de ellas, se muestra un mensaje indicando que se ha enviado un correo, y tras 5 segundos el usuario es redirigido automáticamente a la página donde podrá establecer su nueva contraseña. Si el correo no está registrado, se muestra un mensaje de error informando de ello.

```
#!/usr/bin/php
header("Content-Type: text/html; charset=UTF-8"); // Establece el tipo de contenido y la codificación
require_once("../php/conexion/conexion.php"); // Incluye el archivo de conexión a la base de datos

$correo = $_POST['correo'] ?? ''; // Obtiene el correo del formulario o asigna cadena vacía

if ($_SERVER['REQUEST_METHOD'] == "POST" && !empty($correo)) { // Si el método es POST y el correo no está vacío
    // Verifica si el correo existe en la base de datos
    $stmt = $conexion->prepare("SELECT `profesor` AS tipo_usuario FROM profesores WHERE correo = ?;
        UNION
        SELECT `alumno` AS tipo_usuario FROM clientes WHERE correo = ?"); // Prepara la consulta para buscar el correo en profesores o clientes
    $stmt->bind_param("ss", $correo, $correo); // Asocia los parámetros a la consulta
    $stmt->execute(); // Ejecuta la consulta
    $resultado = $stmt->get_result(); // Obtiene el resultado de la consulta
    $tipo_usuario = $resultado->fetch_assoc()['tipo_usuario'] ?? null; // Obtiene el tipo de usuario o null si no existe

    if ($tipo_usuario) { // Si el correo existe
        $mensaje_error = "⚠ Este correo no está registrado."; // Muestra mensaje de error
    } else { // Si el correo no existe
        echo "<p style='color: green;'>Tu correo enviado. Redirigiendo en 5 segundos...</p>"; // Mensaje de éxito
        echo "script=setTimeout(function(){ window.location.href = '../html/establecer_contraseña/establecer_contraseña.php?correo=$correo&tipo=$tipo_usuario'; }, 5000);";
        exit(); // Finaliza la ejecución del script
    }
}

```



```
<!-- ===== REESTABLECER CONTRASEÑA ===== -->
<h2>Recuperar contraseña</h2>

<form method="POST" id="formulario-recuperar">
    <label for="correo">Introduce tu correo:</label>
    <input type="email" name="correo" id="correo" placeholder="Correo electrónico" required>
    <p class="mensaje-error">?= $mensaje_error ?? '' ?</p>
    <input type="submit" value="Recuperar contraseña">
</form>
```

```
:root {
    color-fondo: #f7faf4;
    color-principal: #midnightblue;
    color-secundario: #eeee22;
    color-texto: #1a1a1a;
    color-error: #e74c3c;
    borde-radio: 10px;
    sombra: 0 6px 18px rgba(0, 0, 0, 0.1);
    transicion: 0.3s ease;
}
```

Una vez pasados los 5 segundos, te pasa al recuperar_contraseña.

Este código muestra el formulario que permite al usuario establecer una nueva contraseña después de haber solicitado recuperarla. El formulario incluye dos campos: uno para escribir la nueva contraseña y otro para confirmarla. Además, se envían de forma oculta (con **input type="hidden"**) el correo del usuario y el tipo de usuario (cliente o profesor), para que el sistema sepa a quién pertenece la cuenta.

También incluye un botón para **mostrar u ocultar la contraseña**, facilitando al usuario comprobar lo que escribe. Esta función se activa con JavaScript: si el botón dice "Mostrar", los campos se cambian para que muestren el texto; si dice "Ocultar", vuelven a ocultarlo. Por último, se carga un archivo JavaScript externo que se encarga de **validar** que las contraseñas sean correctas (por ejemplo, que coincidan o tengan el formato adecuado) antes de que se envíe el formulario para cambiar la contraseña definitivamente.

```
<!-- ===== RECLAMAR CONTRASEÑA Y CAMBIAR ===== -->
<h2>Establecer nueva contraseña</h2>

<form id="formulario-cambio" method="POST" action="../../php/procesar_cambio_contraseña/procesar_cambio_contraseña.php">
    <input type="hidden" name="correo" value=>?> htmlspecialchars($correo) ?>>
    <input type="hidden" name="tipo_usuario" value=>?> htmlspecialchars($tipo_usuario) ?>>
    <label for="nueva_contraseña">Nueva contraseña:</label>
    <input type="password" name="nueva_contraseña" id="nueva_contraseña" required>
    <p class="mensaje-error" id="error-contraseña"></p>
    <label for="confirmar_contraseña">Confirmar nueva contraseña:</label>
    <input type="password" name="confirmar_contraseña" id="confirmar_contraseña" required>
    <p class="mensaje-error" id="error-confirmar"></p>
    <input type="submit" value="Cambiar contraseña">
    <button type="button" id="mostrar-contraseña" style="margin-top:10px;">Mostrar contraseña</button>
</form>

<script>
document.addEventListener("DOMContentLoaded", function() {
    const formularioCambio = document.getElementById("formulario-cambio");
    const inputNuevaContraseña = document.getElementById("nueva_contraseña"); // Campo de nueva contraseña
    const inputConfirmarContraseña = document.getElementById("confirmar_contraseña"); // Campo de confirmación
    const mensajeErrorContraseña = document.getElementById("error-contraseña"); // Mensaje de error de contraseña
    const mensajeErrorConfirmar = document.getElementById("error-confirmar"); // Mensaje de error de confirmación

    formularioCambio.addEventListener("submit", function(event) { // Evento al enviar el formulario
        event.preventDefault(); // Previene el envío por defecto

        const nuevaContraseña = inputNuevaContraseña.value.trim(); // Obtiene y limpia la nueva contraseña
        const confirmarContraseña = inputConfirmarContraseña.value.trim(); // Obtiene y limpia la confirmación

        let errores = false; // Variable para controlar si hay errores

        if (nuevaContraseña.length < 8) { // Verifica longitud mínima
            mensajeErrorContraseña.textContent = "▲ La contraseña debe tener al menos 8 caracteres.";
            errores = true;
        } else if (/([A-Z]).test(nuevaContraseña)) { // Verifica mayúscula
            mensajeErrorContraseña.textContent = "▲ Debe incluir al menos una letra mayúscula.";
            errores = true;
        } else if (/(\d)/.test(nuevaContraseña)) { // Verifica número
            mensajeErrorContraseña.textContent = "▲ Debe incluir al menos un número.";
            errores = true;
        } else if (/[^0-9A-Za-z]/.test(nuevaContraseña)) { // Verifica carácter especial
            mensajeErrorContraseña.textContent = "▲ Debe incluir al menos un carácter especial.";
            errores = true;
        } else {
            mensajeErrorContraseña.textContent = ""; // Limpia mensaje de error de contraseña
        }

        if (nuevaContraseña !== confirmarContraseña) { // Verifica coincidencia de contraseñas
            mensajeErrorConfirmar.textContent = "▲ Las contraseñas no coinciden.";
            errores = true;
        } else {
            mensajeErrorConfirmar.textContent = ""; // Limpia mensaje de error de confirmación
        }

        if (errores) return; // Si hay errores, no envía el formulario
    });
});
<script src="../../js/validacion_recuperacion_contraseña/validacion_recuperacion_contraseña.js" defer></script>
</body>
</html>
```

```
document.addEventListener("DOMContentLoaded", function () { // Espera a que el DOM esté cargado
    const formularioCambio = document.getElementById("formulario-cambio"); // Obtiene el formulario
    const inputNuevaContraseña = document.getElementById("nueva_contraseña"); // Campo de nueva contraseña
    const inputConfirmarContraseña = document.getElementById("confirmar_contraseña"); // Campo de confirmación
    const mensajeErrorContraseña = document.getElementById("error-contraseña"); // Mensaje de error de contraseña
    const mensajeErrorConfirmar = document.getElementById("error-confirmar"); // Mensaje de error de confirmación

    formularioCambio.addEventListener("submit", function (event) { // Evento al enviar el formulario
        event.preventDefault(); // Previene el envío por defecto

        const nuevaContraseña = inputNuevaContraseña.value.trim(); // Obtiene y limpia la nueva contraseña
        const confirmarContraseña = inputConfirmarContraseña.value.trim(); // Obtiene y limpia la confirmación

        let errores = false; // Variable para controlar si hay errores

        if (nuevaContraseña.length < 8) { // Verifica longitud mínima
            mensajeErrorContraseña.textContent = "▲ La contraseña debe tener al menos 8 caracteres.";
            errores = true;
        } else if (/([A-Z]).test(nuevaContraseña)) { // Verifica mayúscula
            mensajeErrorContraseña.textContent = "▲ Debe incluir al menos una letra mayúscula.";
            errores = true;
        } else if (/(\d)/.test(nuevaContraseña)) { // Verifica número
            mensajeErrorContraseña.textContent = "▲ Debe incluir al menos un número.";
            errores = true;
        } else if (/[^0-9A-Za-z]/.test(nuevaContraseña)) { // Verifica carácter especial
            mensajeErrorContraseña.textContent = "▲ Debe incluir al menos un carácter especial.";
            errores = true;
        } else {
            mensajeErrorContraseña.textContent = ""; // Limpia mensaje de error de contraseña
        }

        if (nuevaContraseña !== confirmarContraseña) { // Verifica coincidencia de contraseñas
            mensajeErrorConfirmar.textContent = "▲ Las contraseñas no coinciden.";
            errores = true;
        } else {
            mensajeErrorConfirmar.textContent = ""; // Limpia mensaje de error de confirmación
        }

        if (errores) return; // Si hay errores, no envía el formulario
    });
});

formularioCambio.submit(); // Envía el formulario si no hay errores
});
```

```
:root {
    --color-principal: #midnightblue;
    --color-acento: #eeee22;
    --color-fondo: #f7faf4;
    --color-texto: #222831;
    --color-error: #d00000;
    --radio-borde: 10px;
    --sombra: 0 4px 16px rgba(0, 0, 0, 0.1);
    --transicion: 0.3s ease;
}
```

Por último, para validar este formulario en el lado servidor, tenemos el **procesar_cambio_contraseña.php**

```
<?php
header("Content-type: text/html; charset=UTF-8");
require_once('../../php/conexion/conexion.php');

$correo = $_POST['correo'] ?? '';
$tipo_usuario = $_POST['tipo_usuario'] ?? '';
$nueva_contraseña = $_POST['nueva_contraseña'] ?? '';

if (empty($correo) || empty($tipo_usuario) || empty($nueva_contraseña)) {
    echo "<p style='color: red;'>▲ Datos inválidos.</p>";
    exit();
}

// Hashear la nueva contraseña
$hsh_contraseña = password_hash($nueva_contraseña, PASSWORD_DEFAULT);
$tabla = $tipo_usuario === 'profesores' ? 'profesores' : 'clientes';

$stmt = $conexion->prepare("UPDATE $tabla SET contraseña = ? WHERE correo = ?");
$stmt->bind_param("ss", $hsh_contraseña, $correo);
$stmt->execute();

// Redirigir según el tipo de usuario
$url_redireccion = $tipo_usuario === "profesor" ? "../../html/login_area_profesores/login_area_profesores.php" : "../../html/login_usuario/login_usuario.html";

// Mostrar mensaje de éxito antes de la redirección
echo "<p style='color: green; font-size: 10px;'>Tu contraseña ha sido actualizada correctamente! En breves serás redirigido al login.</p>";
echo "<script>setTimeout(function(){ window.location.href = '$url_redireccion'; }, 3000);</script>";
?>
```

Este código procesa el cambio de contraseña. Primero verifica que se hayan enviado correctamente el correo, tipo de usuario y nueva contraseña. Luego, **protege la nueva contraseña cifrándola (hash)** y **actualiza la base de datos** en la tabla correspondiente (profesores o clientes). Finalmente, **muestra un mensaje de confirmación de que todo ha salido correctamente y redirige automáticamente al formulario de inicio de sesión del usuario en 3 segundos.**

Ya explicado como funciona el recuperar y establecer contraseña, seguimos con el inicio de sesión.

Nos quedamos en que pasa antes de que se envíen los datos al servidor, se realiza una validación usando **JavaScript**. Esto significa que se comprueba si los campos están vacíos. Si alguno lo está, se muestra un mensaje de advertencia sin necesidad de recargar la página, lo cual mejora la experiencia del usuario y evita enviar información incompleta al servidor.

```
document.addEventListener('DOMContentLoaded', function () {
  const formulario = document.getElementById('formulario-login');

  if (formulario) {
    formulario.addEventListener('submit', async function (e) {
      e.preventDefault();

      // Limpiar errores anteriores
      document.getElementById('error-usuario').textContent = '';
      document.getElementById('error-contraseña').textContent = '';

      const usuario = document.getElementById('usuario').value.trim();
      const contraseña = document.getElementById('contraseña').value.trim();

      let hayError = false;

      if (!usuario) {
        document.getElementById('error-usuario').textContent = '⚠ El usuario es obligatorio.';
        hayError = true;
      }

      if (!contraseña) {
        document.getElementById('error-contraseña').textContent = '⚠ La contraseña es obligatoria.';
        hayError = true;
      }

      if (hayError) return;

      try {
        const respuesta = await fetch('../php/verificar_login_usuarios/verificar_login_usuarios.php', {
          method: 'POST',
          headers: { 'Content-Type': 'application/json' },
          body: JSON.stringify({ usuario, contraseña })
        });

        const datos = await respuesta.json();

        if (datos.success) {
          window.location.href = '../html/areas_alumnos/area_alumnos.php';
        } else {
          if (datos.error === 'usuario') {
            document.getElementById('error-usuario').textContent = '⚠ Usuario no válido.';
          } else if (datos.error === 'contraseña') {
            document.getElementById('error-contraseña').textContent = '⚠ Contraseña incorrecta.';
          } else {
            document.getElementById('error-contraseña').textContent = '⚠ Usuario o contraseña incorrectos.';
          }
        }
      } catch (error) {
        console.error('Error al verificar login:', error);
        document.getElementById('error-contraseña').textContent = '⚠ Error de conexión con el servidor.';
      }
    });
  }
});
```

Una de las funciones más importantes de este sistema es el uso de **fetch**. Esta función de JavaScript permite enviar datos al servidor sin recargar la página, lo que se conoce como una petición asíncrona. En este caso, se utiliza **fetch** para mandar el usuario y la contraseña al archivo **PHP** que se encarga de verificar si los datos son correctos. Se envía la información en formato **JSON**, y luego se espera una respuesta del servidor. Si los datos son correctos, el usuario es redirigido al área de alumnos. Si no lo son, se muestran mensajes de error específicos, como por ejemplo si el usuario no existe o si la contraseña es incorrecta.

Del lado del servidor, el archivo **PHP** realiza varias acciones de seguridad. Primero, se inicia una sesión y se comprueba que no haya ya una sesión activa de alumno o profesor, para evitar conflictos. Despues, se utiliza una consulta preparada para buscar en la base de datos al usuario, lo que protege contra ataques conocidos como inyección **SQL**. Las contraseñas están cifradas, por lo tanto, se utiliza la función **password_verify()** para compararlas de forma segura. Si el usuario es válido, se guarda la información en una sesión, se genera un **nuevo ID de sesión** y se redirige al usuario a su área privada.

```

<?php
// Iniciar la sesión de manera segura
session_start();
session_regenerate_id(true);

// Bloquear acceso si ya hay una sesión de profesor activa
if (isset($_SESSION['profesor_id'])) {
    $_SESSION['mensaje'] = "⚠ Ya tienes una sesión activa como profesor. Para iniciar sesión como alumno, primero debes cerrar sesión.";
    header("Location: ../../login_usuario/login_usuario.html?error=Ya_tienes_sesion_profesor");
    exit();
}

// Conexión a la base de datos
require_once("../php/conexion/conexion.php");

// Si ya hay una sesión activa, mostramos un mensaje antes de redirigir
if (isset($_SESSION['mensaje'])) {
    $_SESSION['mensaje'] = "⚠ Ya hay una sesión iniciada. Cierra sesión antes de iniciar otra.";
    header("Location: ../../login_usuario/login_usuario.html?sesion_activa=1");
    exit();
}

// Verificar si se han enviado los datos del formulario
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    // Eliminar espacios en blanco
    $usuario = filter_var(trim($_POST['usuario']), FILTER_SANITIZE_EMAIL);
    $contraseña = trim($_POST['contraseña']);

    // Uso de 'prepared statements' para evitar inyecciones SQL
    $query = "SELECT id, nombre, apellidos, correo, contraseña FROM clientes WHERE correo = ?";
    $stmt = $conexion->prepare($query);
    if ($stmt) {
        die("Error al preparar la consulta: " . $conexion->error);
    }
    $stmt->bind_param("s", $usuario);
    $stmt->execute();
    $result = $stmt->get_result();
}

```

```

if ($result->num_rows > 0) {
    $row = $result->fetch_assoc();

    if (password_verify($contraseña, $row['contraseña'])) {

        // Bloquea que otro usuario inicie sesión si ya hay una cuenta activa en esta sesión
        if (isset($_SESSION['usuario'])) {
            $_SESSION['mensaje'] = "⚠ No puedes iniciar otra cuenta mientras hay una sesión activa.";
            header("Location: ../../login_usuario/login_usuario.html?sesion_activa=1");
            exit();
        }

        // Guarda datos del usuario en la sesión
        $_SESSION['usuario'] = [
            'id' => $row['id'],
            'nombre' => $row['nombre'],
            'apellidos' => $row['apellidos'],
            'correo' => $row['correo']
        ];
        $_SESSION['ultimo_acceso'] = time();

        session_regenerate_id(true); // Seguridad extra para evitar el robo de sesión

        // Redirige al área de alumnos
        header("Location: ../../area_alumnos/area_alumnos.php");
        exit();
    } else {
        $_SESSION['mensaje'] = "⚠ Correo o contraseña incorrectos.";
        header("Location: ../../login_usuario/login_usuario.html?error=1");
        exit();
    }
}

$stmt->close();
$conexion->close();
}

```

Además, si ya hay una sesión iniciada (independiente si es el profesor el que le tiene iniciada o el alumno), el sistema muestra una alerta al usuario o al profesor con un mensaje que le informa que debe cerrar la sesión actual antes de iniciar otra. Esto se hace revisando un parámetro en la URL con JavaScript, y mostrando el mensaje automáticamente si es necesario.

Por último, el formulario incluye un botón que permite mostrar u ocultar la contraseña. Esto se logra con un pequeño script que cambia el tipo del campo entre "password" (oculto) y "text" (visible), para que el usuario pueda revisar lo que ha escrito si lo desea.

Si todo ha salido bien, hemos iniciado la sesión y entramos aquí.



```

<!-- ===== AREA DEL ALUMNO ===== -->


<h1 id="titulo-alumno">Bienvenid@ a tu área de alumno de la Autoescuela Almansa.es</h1>
    <p id="texto-alumno">Aquí podrás gestionar tu cuenta, ver tus clases, consultar tus notas y mucho más.</p>

    <div class="botones-alumno">
        
        <button data-enlace="test" class="btn-alumno">Vamos a hacer test?</button>
    </div>

    
    <button data-enlace="estadisticas" class="btn-alumno">Vamos a hacer test?</button>

    
    <button data-enlace="compras_usuario" class="btn-alumno">¿Qué has comprado?</button>


```

Este código muestra la **zona privada del alumno** en la web de la autoescuela. En esta área, el alumno ve un mensaje de bienvenida y puede acceder a diferentes funciones mediante botones con íconos animados (GIFs):

- **Hacer test:** con un botón que redirige a la sección de tests.
- **Consultar notas:** con un botón que redirige a sus resultados, usando el ID guardado en la sesión. El `session_start()` gestiona que la sesión pueda estar activa.

```
<?php
session_start();

$mensaje = '';
if (isset($_SESSION['mensaje'])) {
    $mensaje = $_SESSION['mensaje'];
    unset($_SESSION['mensaje']);
}
?>
```

- **Ver compras:** con otro botón para revisar lo que ha adquirido.

Todo está organizado de forma visual y accesible, facilitando al alumno la gestión de su cuenta.

En la imagen del código se aprecian varios caracteres en rojo y no son fallos, es que para no hacer la captura excesivamente larga, he hecho un salto de linea para que se viese más cortado y por eso mismo se ve en rojo.

Llegados a esta parte, lo más importante es hacer test. Para ello, si intentamos acceder a los test sin haber pagado el teórico o el pack completo nos llevará a la página de precios.html.

The image consists of three parts. The left part is a screenshot of a website's student area with various course options like 'Teórico', 'Clase Práctica Suelta', 'Pack Completo', etc. The middle part is a snippet of CSS code for a class named 'root'. The right part is a screenshot of a browser's developer tools showing a message about a purchase being added to the cart.

Precios Carnet B

- Teórico**
Precio: 150€
Incluye:
 - Material didáctico
 - Acceso a teoría online
 - Acceso a teoría escrita**Añadir al carrito - Teórico**
- Clase Práctica Suelta**
Precio: 30€/clase
Incluye:
 - Clases prácticas de 1 hora
 - Vehículo moderno y seguro
 - Instructor certificado**Añadir al carrito - Clase Suelta**
- Pack Completo**
Precio: 950€
Incluye:
 - Teórico completo
 - Clases prácticas de 1 hora
 - Vehículo moderno y seguro
 - Instructor certificado**Añadir al carrito - Pack Completo**
- Curso Intensivo**
Precio: 350€
Incluye:
 - Clases teóricas intensivas
 - Material de estudio
 - Acceso a teoría online**Añadir al carrito - Curso Intensivo**
- Pack de 10 Clases Prácticas**
Precio: 280€
Incluye:
 - 10 clases prácticas de 1 hora
 - Vehículo moderno y seguro
 - Instructor certificado**Añadir al carrito - Pack de 10 Clases**
- Pack de 20 Clases Prácticas**
Precio: 540€
Incluye:
 - 20 clases prácticas de 1 hora
 - Vehículo moderno y seguro
 - Instructor certificado**Añadir al carrito - Pack de 20 Clases**
- Oportunidad Extra de Examen**
Precio: 175€
Incluye:
 - Un examen adicional para obtener el carné de conducir
 - Gestión administrativa del examen**Añadir al carrito - Oportunidad Extra**

VARIABLES - PRECIO

```
.root {
    --color-principal: #1e232e;
    --color-blanco: #fff;
    --color-azul: #1e232e;
    --color-borde: #ddd;
    --color-texto-secundario: #555;
    --color-shadow-fuerte: #rgba(0, 0, 0, 0.8);
    --color-shadow-suave: #rgba(0, 0, 0, 0.2);
    --color-shadow-menu: #rgba(0, 0, 0, 0.25);
    --color-hover-icon: #2563eb;
    --color-icon-bg: #f1e3a8;
    --color-entorno: #f1f1f1;
    --color-texto-header: #1f7297;
    --color-texto-subtle: #6b7288;
    --color-bg-item: #f3f4f6;
}
```

localhost dice
 'Pack Completo' añadido al carrito.
Aceptar

Aquí tendrás que añadir el Teórico o el Pack Completo para hacer los test. Si por ejemplo intentas añadir el Teórico estando el Pack Completo, te dará error al igual por lo que sólo podrás añadir un producto de los dos y eso es gracias a esta función del archivo **carritoFinal.js** que hace que la compra funcione correctamente con los archivos de **registrar_comprar.php** y **verificar_usuario.php**

```
// ===== AGREGAR PRODUCTO AL CARRITO =====
function agregarAlCarrito(nombre, precio, tipo) {
  fetch("../php/verificar_usuario.php")
    .then(response => response.json())
    .then(data => {
      let productosRecurrentes = ["Práctico", "Pack de 10 Clases Prácticas", "Pack de 20 Clases Prácticas", "Curso Intensivo"];
      let productosComprados = data.productos_comprados || [];

      if (!productosRecurrentes.includes(nombre) && productosComprados.includes(nombre)) {
        alert(`❌ Ya has comprado '${nombre}'. No puedes añadirlo nuevamente.`);
        return;
      }

      precio = parseFloat(precio);
      if (isNaN(precio)) {
        alert(`❌ Precio inválido para '${nombre}'`);
        return;
      }

      let carrito = JSON.parse(localStorage.getItem("carrito")) || [];

      // Verificar restricción entre "Pack Completo" y "Teórico"
      let tienePackCompleto = carrito.some(item => item.nombre === "Pack Completo");
      let tieneTeorico = carrito.some(item => item.nombre === "Teórico");

      if ((nombre === "Pack Completo" && tieneTeorico) || (nombre === "Teórico" && tienePackCompleto)) {
        alert(`❌ No puedes añadir '${nombre}' porque ya tienes en el carrito un producto que es compatible con el teórico.`);
        return;
      }

      // No permitir añadir dos veces "Teórico" o "Pack Completo"
      if (
        (nombre === "Teórico" && carrito.some(item => item.nombre === "Teórico")) ||
        (nombre === "Pack Completo" && carrito.some(item => item.nombre === "Pack Completo"))
      ) {
        alert(`❌ No puedes añadir '${nombre}' más de una vez al carrito.`);
        return;
      }

      let existente = carrito.find(item => item.nombre === nombre);

      if (existente) {
        existente.cantidad += 1;
      } else {
        carrito.push({ nombre, precio: Number(precio), tipo, cantidad: 1 });
      }

      localStorage.setItem("carrito", JSON.stringify(carrito));
      actualizarCarrito();
      alert(`✅ '${nombre}' añadido al carrito.`);
    })
    .catch(error => console.error(`❌ Error al verificar usuario: ${error}`));
}
```

Primero, realiza una solicitud con **fetch** al archivo **verificar_usuario.php** para obtener información sobre los productos que el usuario ya ha comprado. La respuesta del servidor se convierte en un objeto **JavaScript** utilizando **.json()**, lo que permite verificar si el producto seleccionado es recurrente o si ya ha sido adquirido anteriormente. Luego, la función verifica que el precio ingresado sea válido, convirtiéndolo en número y asegurándose de que no sea **NaN**. Posteriormente, recupera el carrito almacenado en **localStorage**, donde la información se encuentra guardada en formato **JSON**.

Se incluyen reglas para evitar conflictos entre productos, impidiendo que el usuario agregue "**Pack Completo**" si ya tiene "**Teórico**" y viceversa. Además, la función evita que se añadan más de una vez estos productos específicos.

Finalmente, si el producto puede ser agregado, se actualiza el carrito en **localStorage** y se llama a **actualizarCarrito()** para reflejar los cambios, mostrando una alerta confirmando la acción. Este mecanismo garantiza que los productos se gestionen correctamente en el carrito, evitando duplicaciones y combinaciones incompatibles.

Para la solicitud que he mencionado anteriormente del **verificar_usuario.php**, voy a explicar que es lo que hace

Este código PHP genera una respuesta en formato JSON con información sobre el estado de la sesión del usuario y los productos que ha comprado. Primero, verifica si el usuario tiene una sesión activa y, si no la tiene, devuelve un JSON indicando que no puede acceder. Luego, consulta la base de datos para obtener los productos comprados, asegurándose de eliminar duplicados. También determina si el usuario puede acceder a un test basándose en su historial de compras. Finalmente, toda esta información se estructura en un objeto JSON y se envía al cliente, permitiendo que funciones como **agregarAlCarrito(nombre, precio, tipo)** en JavaScript utilicen estos datos para restringir la compra de productos repetidos o incompatibles. Esto garantiza una comunicación efectiva entre el servidor y el cliente.

```
<?php
// Inicia la sesión para acceder a las variables de sesión del usuario
session_start();

// Establece el tipo de contenido de la respuesta como JSON
header("Content-Type: application/json");

// Incluye el archivo de conexión a la base de datos
require_once("../php/conexion/conexion.php");

// Inicializa la respuesta indicando si la sesión está activa y los productos comprados
$response = [
    "sesion_activa" => isset($_SESSION["usuario"]),
    "puede_acceder_test" => false,
    "productos_comprados" => [] // Guarda todos los productos comprados sin duplicados
];

// Si no hay sesión activa, devuelve la respuesta y termina la ejecución
if (!isset($_SESSION["usuario"])) {
    echo json_encode($response);
    exit();
}

// Obtiene el ID del usuario desde la sesión
$usuario_id = $_SESSION["usuario"]["id"];

// Consulta los productos comprados sin duplicados
$query = "SELECT DISTINCT producto FROM compras WHERE usuario_id = ?";
$stmt = $conexion->prepare($query);
$stmt->bind_param("i", $usuario_id);
$stmt->execute();
$result = $stmt->get_result();

// Guarda los productos comprados en el array, eliminando duplicados
while ($row = $result->fetch_assoc()) {
    $response["productos_comprados"][] = $row["producto"];
}

// Si el usuario ha comprado "Pack Completo", puede acceder al test
$response["puede_acceder_test"] = in_array("Pack Completo", $response["productos_comprados"]);

// Cierra la consulta y la conexión a la base de datos
$stmt->close();
$conexion->close();

// Devuelve la respuesta en formato JSON
echo json_encode($response);
?>
```

Dentro del **carritoFinal.js** encontramos más funciones que vamos a utilizar seguidamente porque son las encargadas de borrar un producto, comprobar que ese producto no esté en comprado ya por el usuario, actualizar el carrito con el navegador para que siempre tengamos los productos por todos los html o php en los que nos movamos y tengan linkeado el **carritoFinal.js**

La función **eliminarDelCarrito(nombre)** se encarga de eliminar un producto específico del carrito de compras. Para ello, primero recupera los productos almacenados en el **localStorage**, los convierte a un array mediante **JSON.parse**, y luego aplica un filtro para excluir el producto cuyo nombre coincide con el proporcionado como parámetro. Una vez actualizado el **array**, se guarda nuevamente en **localStorage** con **JSON.stringify** (función de JavaScript que convierte un objeto o un array en una cadena de texto con formato JSON). Finalmente, se llama a **actualizarCarrito()** para reflejar los cambios en la interfaz de usuario.

```
// ===== ELIMINAR PRODUCTO DEL CARRITO =====
function eliminarDelCarrito(nombre) {
    let carrito = JSON.parse(localStorage.getItem("carrito")) || [];
    carrito = carrito.filter(item => item.nombre !== nombre);
    localStorage.setItem("carrito", JSON.stringify(carrito));
    actualizarCarrito();
}
```

La función **actualizarCarrito()** se encarga de sincronizar la vista del carrito con los datos almacenados en localStorage. Recupera los elementos HTML relevantes como la **lista de productos (carrito-items)**, el **total (total-carrito)**, el **contador (carrito-count)** y el mensaje de carrito vacío. Si el carrito está vacío, oculta la lista de productos, muestra el mensaje correspondiente y reinicia los contadores. Si contiene elementos, oculta el mensaje vacío y genera dinámicamente los productos en pantalla, sumando sus precios y cantidades para mostrar un resumen actualizado. También se incluye un botón para eliminar productos directamente desde la vista.

```
/*----- ACTUALIZAR VISUAL DEL CARRITO -----*/
function actualizarCarrito() {
    let carrito = JSON.parse(localStorage.getItem("carrito") || []);
    let carritoItems = document.getElementById("carrito-items");
    let totalElement = document.getElementById("total-carrito");
    let contadorCarrito = document.getElementById("carrito-count");
    let mensajeVacio = document.getElementById("carrito-vacio");

    if (!carritoItems || !totalElement || !contadorCarrito || !mensajeVacio) return;

    carritoItems.innerHTML = "";
    let total = 0, cantidadTotal = 0;

    if (carrito.length === 0) {
        mensajeVacio.style.display = "block";
        carritoItems.style.display = "none";
        totalElement.textContent = "0.00€";
        contadorCarrito.textContent = "0";
        return;
    } else {
        mensajeVacio.style.display = "none";
        carritoItems.style.display = "block";
    }

    carrito.forEach(item => {
        let itemElement = document.createElement("div");
        itemElement.classList.add("carrito-item");
        itemElement.innerHTML =
            `<span>${item.nombre} <span>${(item.precio.toFixed(2)) * item.cantidad}</span>
            <button class="eliminar" onclick="eliminarDelCarrito('${item.nombre.replace(/\s/g, '\\\\')}')>X</button>`;
        carritoItems.appendChild(itemElement);
        total += item.precio * item.cantidad;
        cantidadTotal += item.cantidad;
    });

    totalElement.textContent = `${total.toFixed(2)}€`;
    contadorCarrito.textContent = `${cantidadTotal}`;
}
```

La función **verificarSesionYPagar()** comprueba si el usuario tiene una sesión activa antes de permitirle proceder con el pago. Utiliza **fetch()** para hacer una solicitud a un archivo **verificar_usuario.php** encargado de validar la sesión. Si el usuario no ha iniciado sesión, se le redirige a la página de **login**. También verifica que el carrito no esté vacío; en caso contrario, redirige a la página de precios. Si ambas condiciones se cumplen, se almacena el carrito y se dirige al usuario hacia la pasarela de pago.

```
/*----- VERIFICAR Y PROCEDER CON EL PAGO -----*/
function verificarSesionYPagar() {
    fetch("../php/verificar_usuario_pago/verificar_usuario.php")
        .then(response => response.json())
        .then(data => {
            if (!data.sesion_activa) {
                alert("⚠ Debes iniciar sesión para realizar la compra.");
                window.location.href = "../html/login_usuario/login_usuario.html";
                return;
            }

            let carrito = JSON.parse(localStorage.getItem("carrito") || []);

            if (carrito.length === 0) {
                alert("⚠ El carrito está vacío.");
                window.location.href = "../html/precio/precio.html";
                return;
            }

            // Guardamos el carrito antes de redirigir
            localStorage.setItem("carrito", JSON.stringify(carrito));

            // Redirigir a la pasarela de pago
            window.location.href = "../html/pasarela_pago/pasarela_pago.html";
        })
        .catch(error => console.error("✗ Error al verificar sesión:", error));
}
```

La función **registrarCompra(carrito)** toma los productos del carrito y los transforma para incluir solo los datos esenciales: nombre, precio unitario, cantidad y precio total por producto. Luego, **envía esa información al servidor** usando **fetch** y el archivo **PHP registrar_comprar.php**.

Si la respuesta indica éxito, se borra el carrito y el usuario es redirigido a una página de agradecimiento. Si ocurre un error, se muestra un mensaje para facilitar su detección. Así se asegura que la compra quede correctamente registrada en el servidor.

```
// ===== REGISTRAR LA COMPRA EN PHP =====
function registrarCompra(carrito) {
    let carritoProcesado = carrito.map(item => ({
        nombre: item.nombre,
        precioUnitario: item.precio,
        cantidad: item.cantidad,
        precioTotal: item.precio * item.cantidad
    }));
    fetch("../php/registrar_compra/registrar_comprar.php", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({ carrito: carritoProcesado })
    })
    .then(response => response.json())
    .then(data => {
        if (data.success) {
            alert("✓ Compra registrada correctamente.");
            localStorage.removeItem("carrito");
            window.location.href = "../html/agradecimiento_pago/agradecimiento_pago.html";
        } else {
            alert("✗ Error al registrar la compra: " + data.error);
        }
    })
    .catch(error => console.error("✗ Error al enviar datos al servidor:", error));
}
```

La función **toggleCarrito()** alterna la visibilidad del menú del carrito en la interfaz de usuario. Si el elemento del carrito está presente en el **DOM**, se cambia su clase con **classList.toggle("visible")**, lo que permite abrir o cerrar el menú de forma interactiva al hacer clic.

```
// ===== MOSTRAR / OCULTAR CARRITO =====
function toggleCarrito() {
    const menu = document.getElementById("carrito-menu");
    if (menu) menu.classList.toggle("visible");
}
```

La función **procesarPago()** valida que el usuario haya seleccionado un método de pago antes de continuar. En el caso de que se elija "**Tarjeta**", también verifica que todos los campos (**número, fecha y CVV**) estén correctamente completados. Si alguna validación falla, se muestra un mensaje de advertencia. Si todo está en orden, se muestra un mensaje indicando que el pago se está procesando y, tras una pequeña espera simulada con **setTimeout**, se llama a **realizarPago(carrito)** para completar la operación.

```
function procesarPago() {
    let carrito = JSON.parse(localStorage.getItem("carrito")) || [];
    let metodoPago = localStorage.getItem("metodoPago");

    if (!metodoPago) {
        alert("⚠ Selecciona un método de pago antes de proceder.");
        return;
    }

    if (carrito.length === 0) {
        alert("⚠ No hay productos en el carrito.");
        window.location.href = "../../html/precio/precio.html";
        return;
    }
    if (metodoPago === "Tarjeta") {
        let numero = document.getElementById("numero").value.trim();
        let fecha = document.getElementById("fecha").value.trim();
        let cvv = document.getElementById("cvv").value.trim();

        if (numero === "" || fecha === "" || cvv === "") {
            alert("⚠ Completa todos los campos de la tarjeta para proceder.");
            return;
        }
    }

    alert("⏳ Procesando pago con ${metodoPago}...");

    setTimeout(() => {
        alert("✅ Pago realizado con éxito usando ${metodoPago}.");
        realizarPago(carrito);
    }, 2000);
}
```

Gracias a esta función se gestiona el carrito de compras: al cargar la página, se muestran los productos guardados en **localStorage**, incluyendo nombre, cantidad, precio total e IVA. También permite eliminar productos y muestra el total a pagar.

La función **realizarPago(carrito)** se encarga de **enviar el carrito al servidor** mediante una solicitud **POST** en formato **JSON** al archivo **registrar_comprar.php**. Si la respuesta del servidor confirma el registro de la compra, se borra el carrito y se redirige al usuario a una página de agradecimiento. Si hay un error, se informa por consola o mediante una alerta. Así, se asegura que la compra quede registrada correctamente tanto en el navegador como en el servidor.

```
// ===== ENVIAR LA COMPRA AL SERVIDOR =====
function realizarPago(carrito) {
    console.log("➡ Enviando compra al servidor:", carrito);

    fetch("../../php/registrar_compra/registrar_comprar.php", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({ carrito })
    })
    .then(response => response.json())
    .then(data => {
        console.log("➡ Respuesta del servidor:", data);

        if (data.success) {
            alert("✅ Compra registrada correctamente.");
            localStorage.removeItem("carrito");
            window.location.href = "../../html/agradecimiento_pago/agradecimiento_pago.html";
        } else {
            alert("❌ Error al registrar la compra: " + data.error);
        }
    })
    .catch(error => console.error("❌ Error al enviar datos al servidor:", error));
}
```

Por último, el evento **DOMContentLoaded** asegura que, una vez cargado el contenido del DOM, se ejecute **actualizarCarrito()** si existe el elemento correspondiente en la página. Además, añade un evento al botón de pago (si está presente) para que al hacer clic se active la función **verificarSesionYPagar()**, asegurando así una correcta secuencia antes de permitir el acceso a la compra.

```
// ====== EVENTOS AL CARGAR ======
document.addEventListener("DOMContentLoaded", () => {
  if (document.getElementById("carrito-items")) actualizarCarrito();

  const btnPagar = document.querySelector(".btn-pago");
  if (btnPagar) btnPagar.addEventListener("click", verificarSesionYPagar);
});
```

Una vez explicada la lógica del carrito y de la compra en general, llegaríamos al resumen de la compra, donde encontraríamos lo que queremos comprar, calculando el 21% de IVA y lo que nos vamos a gastar.

Seguidamente los métodos de pagos que son PayPal, Bizum o Tarjeta. Es decir, esto es la pasarela de pago de mi autoescuela digital.



```
<!-- ===== RESUMEN DE TU COMPRA =====
<div id="resumen-compra">
  <div>Resumen de tu compra</div>
  <div id="carrito-contenedor"></div>
</div>

<!-- Botones para métodos de pago -->
<div id="metodos-pago">
  <h2>Elige tu método de pago</h2>
  <button class="boton-pago paypal" onclick="mostrarFormulario('PayPal')">PayPal</button>
  <button class="boton-pago bizum" onclick="mostrarFormulario('Bizum')">Bizum</button>
  <button class="boton-pago tarjeta" onclick="mostrarFormulario('Tarjeta')">Tarjeta</button>
</div>

<!-- Contenedor para el formulario de pago dinámico -->
<div id="formulario-pago" style="display: none;"></div>

<!-- Botón para finalizar el pago -->
<button id="finalizarPago" style="display: none;" onclick="procesarPago()">Finalizar Pago </button>
```

```
:root {
  /* Colores principales */
  --color-primario: #midnightblue;
  --color-secundario: #fafffa;
  --color-terciario: #white;
  --color-hover: #eeee22;
  --color-texto-secundario: #gray;
  --color-fondo-total: #e0e7ff;

  /* Sombras */
  --sombra-suave: 0 4px 10px #rgba(0, 0, 0, 0.1);
  --sombra-fuerte: 0 6px 15px #rgba(0, 0, 0, 0.2);
  --sombra-hover: 0 6px 15px #rgba(0, 0, 0, 0.3);

  /* Otros */
  --borde-radio: 10px;
  --borde-radio-grande: 15px;
  --transicion-btn: background-color 0.3s ease, transform 0.2s ease, box-shadow 0.3s ease;
}
```

```
function mostrarFormulario(tipo) {
  document.querySelector("#resumen-compra").style.display = "block";
  document.querySelector("#carrito-contenedor").style.display = "block";

  const formularioPago = document.getElementById("formulario-pago");
  const finalizarPago = document.getElementById("finalizarPago");

  formularioPago.innerHTML = "";
  formularioPago.style.display = "block";
  finalizarPago.style.display = "block";

  if (tipo === "Bizum") {
    formularioPago.innerHTML =
      `

<h3>PAGO CON TARJETA</h3>
        <label>Número de Tarjeta:</label>
        <input type="text" id="numero" placeholder="1234 5678 9012 3456" />
        <label>Expiración:</label>
        <input type="text" id="expiracion" placeholder="MM/AA" />
        <label>CVV:</label>
        <input type="text" id="cvv" placeholder="123" />

`;
    formularioPago.querySelector("button").style.display = "block";
  } else if (tipo === "Tarjeta") {
    formularioPago.innerHTML =
      `

<h3>PAGO CON TARJETA</h3>
        <label>Número de Telefono:</label>
        <input type="text" id="telefono" maxlength="9" placeholder="Introduce tu móvil" />

`;
    formularioPago.querySelector("button").style.display = "block";
  } else if (tipo === "PayPal") {
    formularioPago.innerHTML =
      `

<h3>PAGO CON PAYPAL</h3>
        <label>Correo electrónico de PayPal:</label>
        <input type="text" id="correo-email" placeholder="usuariop@mplo.com" />
        <label>Contraseña:</label>
        <input type="password" id="paypal-password" placeholder="Contraseña" />

`;
    formularioPago.querySelector("button").style.display = "block";
  }

  // Agrega el evento para mostrar/cultar el CVV
  setTimeout(() => {
    const cvvInput = document.getElementById("cvv");
    const toggleCvv = document.getElementById("toggle-cvv");
    if (cvvInput) {
      toggleCvv.addEventListener("click", function() {
        if (cvvInput.type === "password") {
          cvvInput.type = "text";
          toggleCvv.textContent = "•••";
        } else {
          cvvInput.type = "password";
          toggleCvv.textContent = "•••";
        }
      });
    }
  }, 0);
}

else if (tipo === "PayPal") {
  formularioPago.innerHTML =
    `

<h3>PAGO CON PAYPAL</h3>
      <label>Correo electrónico de PayPal:</label>
      <input type="text" id="correo-email" placeholder="usuariop@mplo.com" />
      <label>Contraseña:</label>
      <input type="password" id="paypal-password" placeholder="Contraseña" />

`;
}

document.addEventListener("input", function(event) {
  if (event.target.id === "telefono" || event.target.id === "numero" || event.target.id === "cvv") {
    event.target.value = event.target.value.replace(/\D/g, ""); // solo numeros
  }
});
```

La función **mostrarFormulario(tipo)** genera y muestra dinámicamente un formulario según el método de pago seleccionado: **Bizum**, **Tarjeta** o **PayPal**. Guarda la opción elegida en **localStorage** para futuras referencias. Según el tipo, inserta los campos correspondientes en el formulario: número de teléfono para Bizum, datos bancarios para Tarjeta (incluyendo un botón para mostrar u ocultar el CVV), o correo y contraseña para PayPal. Además, se activa un evento global (input) que impide introducir letras en los campos que deben tener solo números (teléfono, tarjeta y CVV). Esto mejora la validación del formulario antes del envío.

Y no podemos olvidarnos de como se hace la lógica del resumen de la compra,

El script de resumen_compra.js gestiona el carrito de compras de la autoescuela . Al cargarse la página (**DOMContentLoaded**), recupera los productos guardados en **localStorage** y los muestra en pantalla con la función **renderizarCarrito()**. Si el carrito está vacío, se informa al usuario. Si contiene productos, se crean elementos **HTML** con el nombre, cantidad, precio total e IVA incluido de cada producto. Además, cada artículo incluye un botón para eliminarlo del carrito, lo cual actualiza los datos tanto en pantalla como en **localStorage**. Al final, se muestra el total a pagar. Por otro lado, la función **realizarPago()** valida si el carrito tiene productos; si está vacío, alerta al usuario y lo redirige a la página de precios. Si hay productos, simula el procesamiento del pago, borra el carrito y redirige al usuario a una página de agradecimiento.

```

document.addEventListener("DOMContentLoaded", function () {
  const carritoContenedor = document.getElementById("carrito-contenedor");
  let carrito = JSON.parse(localStorage.getItem("carrito")) || [];

  function renderizarCarrito() {
    carritoContenedor.innerHTML = "";

    if (carrito.length === 0) {
      const vacio = document.createElement("p");
      vacio.id = "carrito-vacio";
      vacio.textContent = "Tu carrito está vacío. ⚡";
      carritoContenedor.appendChild(vacio);
      return;
    }

    const carritoItems = document.createElement("div");
    carritoItems.id = "carrito-items";

    let total = 0;
    const ivaPorcentaje = 0.21;

    carrito.forEach((item, index) => {
      const itemDiv = document.createElement("div");
      itemDiv.classList.add("carrito-item", "fade-in");

      const nombre = document.createElement("p");
      nombre.textContent = `${item.nombre} (${item.cantidad})`;

      const precioTotal = item.precio * item.cantidad;
      const ivaIncluido = precioTotal - (precioTotal / (1 + ivaPorcentaje));

      const precio = document.createElement("p");
      precio.textContent = `Precio Total: ${precioTotal.toFixed(2)}€`;

      const ivaElemento = document.createElement("p");
      ivaElemento.textContent = `IVA Incluido: ${ivaIncluido.toFixed(2)}€`;

      const eliminar = document.createElement("button");
      eliminar.textContent = "Eliminar ✕";
      eliminar.onclick = () => {
        carrito.splice(index, 1);
        localStorage.setItem("carrito", JSON.stringify(carrito));
        renderizarCarrito();
      };

      itemDiv.appendChild(nombre);
      itemDiv.appendChild(precio);
      itemDiv.appendChild(ivaElemento);
      itemDiv.appendChild(eliminar);
      carritoItems.appendChild(itemDiv);

      total += precioTotal;
    });

    const totalDiv = document.createElement("div");
    totalDiv.id = "total-carrito";
    totalDiv.classList.add("fade-in");

    totalDiv.innerHTML =
      `

<p><strong>Total a Pagar: ${total.toFixed(2)}€</strong></p>

`;

    carritoContenedor.appendChild(carritoItems);
    carritoContenedor.appendChild(totalDiv);
  }
}

// Esta es la función para controlar el pago
function realizarPago() {
  const carrito = JSON.parse(localStorage.getItem("carrito")) || [];

  if (carrito.length === 0) {
    alert("Tu carrito está vacío. No puedes realizar el pago. ⚡ X");
    window.location.href = "../precio/precio.html"; // redirige a la página principal
    return;
  }

  alert("Procesando pago... por favor espere.");
  localStorage.removeItem("carrito");
  window.location.href = "../agradecimiento_pago/agradecimiento_pago.html";
}
}

```

Ya con esto sólo queda explicar como se registra la compra que antes hemos mencionado con el archivo **registrar_comprar.php**

El archivo registrar_comprar.php es el encargado de **procesar y guardar la información de la compra enviada desde el JavaScript** mediante una petición POST en formato JSON. Primero, verifica que exista una sesión activa de usuario para permitir la compra; si no, devuelve un error en formato JSON.

Luego, recibe los datos del carrito enviados desde **realizarPago(carrito)** y **registrar_Compra(carrito)** y, si hay productos, inicia una transacción en la base de datos para registrar cada artículo en la tabla compras. Además, actualiza datos específicos del usuario según el tipo de producto comprado: por ejemplo, controla que solo pueda comprar una “**Oportunidad Extra de Examen**” si no tiene intentos disponibles, y actualiza la cantidad de clases prácticas o paquetes adquiridos en su perfil. El código usa sentencias preparadas para **evitar inyección SQL** y garantiza que todos los cambios se confirmen juntos con **commit()**. Finalmente, devuelve una respuesta **JSON** indicando éxito o error, que el JavaScript usa para notificar al usuario y limpiar el carrito o mostrar mensajes de fallo. Esta comunicación directa con el **script JS** permite una experiencia dinámica y segura en el proceso de pago.

```
<?php
session_start();
header("Content-Type: application/json");
require_once("../php/conexion/conexion.php");

$response = ["success" => false];

if (!isset($_SESSION["usuario"])) {
    $response["error"] = "No hay sesión activa.";
    echo json_encode($response);
    exit();
}

// Obtener datos del carrito enviados por "registrar_compra.js"
$data = json_decode(file_get_contents("php://input"), true);

if (isset($data["carrito"]) && count($data["carrito"]) > 0) {
    $usuario_id = $_SESSION["usuario"]["id"];
    $conexion->begin_transaction();

    foreach ($data["carrito"] as $item) {
        $producto = $item["nombre"];
        $precio = $item["precio"];
        $cantidad = isset($item["cantidad"]) ? $item["cantidad"] : 1; // Capturamos la cantidad comprada

        // Insertar la compra en la tabla 'compras'
        $query = "INSERT INTO compras (usuario_id, producto, precio, cantidad, fecha_compra) VALUES (?, ?, ?, ?, NOW())";
        $stmt = $conexion->prepare($query);
        $stmt->bind_param("isii", $usuario_id, $producto, $precio, $cantidad);
        $stmt->execute();

        // Si compra "Oportunidad Extra", aumentamos intentos de examen
        if ($producto === "Oportunidad Extra de Examen") {
            // Consultar cuántas oportunidades tiene el usuario
            $query_check = "SELECT oportunidades_examenes FROM clientes WHERE id = ?";
            $stmt_check = $conexion->prepare($query_check);
            $stmt_check->bind_param("i", $usuario_id);
            $stmt_check->execute();
            $stmt_check->fetch();
            $stmt_check->close();

            // Permitir compra sólo si tiene 0 oportunidades
            if ($stmt_check->fetch_all() == 0) {
                $query_update = "UPDATE clientes SET oportunidades_examenes = oportunidades_examenes + 1 WHERE id = ?";
                $stmt_update = $conexion->prepare($query_update);
                $stmt_update->bind_param("i", $cantidad, $usuario_id);
                $stmt_update->execute();
            } else {
                $response["error"] = "X No puedes comprar una Oportunidad Extra si aún tienes intentos disponibles.";
            }
        }
    }

    // Si compra "Oportunidad Extra", aumentamos intentos de examen
    if ($stmt_update->fetch_all() == 0) {
        $query_update = "UPDATE clientes SET oportunidades_examenes = oportunidades_examenes + 1 WHERE id = ?";
        $stmt_update = $conexion->prepare($query_update);
        $stmt_update->bind_param("i", $cantidad, $usuario_id);
        $stmt_update->execute();
    }
}
else {
    $response["success"] = true;
}

echo json_encode($response);
?>
```

Una vez finalizada toda la lógica de la compra y como se conecta entre la parte visual y la parte de código, vamos a mostrar que pasa cuando ya puedes acceder a los test y puedes ver tu área de usuario.



```
:root {
    --ancho-maximo-contenedor: 800px;
    --margen-vertical-contenedor: 60px;
    --relleno-contenedor: 30px 25px;
    --color-fondo-contenedor: #ffffff;
    --radio-borde-contenedor: 16px;
    --sombra-contenedor: 0 10px 30px rgba(0, 0, 0, 0.1);
    --alineacion-texto: center;
    --color-texto-principal: #midnightblue;

    --tamano-titulo: 2.5rem;
    --margen-inferior-titulo: 1.5rem;
    --sombra-texto-titulo: 0 2px 4px rgba(0, 0, 0, 0.08);

    --tamano-parrafo: 1.15rem;
    --altura-linea-parrafo: 1.7;
    --margen-inferior-parrafo: 1.5rem;

    --espacio-entre-botones: 16px;
    --margen-superior-botones: 20px;

    --relleno-botón: 12px 24px;
    --color-fondo-botón: #midnightblue;
    --color-texto-botón: #white;
    --tamano-texto-botón: 1rem;
    --peso-texto-botón: 600;
    --radio-borde-botón: 8px;
    --transición-botón: background-color 0.3s ease, transform 0.2s ease;
    --sombra-botón: 0 4px 10px rgba(0, 0, 0, 0.15);

    --color-fondo-botón-hover: #2c3e90;
    --escala-botón-hover: 1.05;

    --borde-focus-botón: 2px solid #2c3e90;
    --separación-borde-focus: 4px;
}
```

Entramos en la parte de los test y encontramos un menú que podemos seleccionar que tipo de test queremos hacer; Test Temáticos, aleatorios o de examen. Seguidamente, encontramos un menú en cada tipo de test y sólo tenemos que seleccionar cual queremos hacer y nos pondríamos a responder preguntas.



```
<!-- ----- MENU DE LOS TEST ----- -->
<h1 id="titulo-test">Menú de Test</h1>
<section class="menu-test">
    <div class="test-options">
        <div class="test-option">
            <h2>Test Temáticos </h2>
            <p>Realiza test organizados por temas específicos para reforzar tus conocimientos en áreas concretas.</p>
            <button data-enlace="test_temáticos" class="btn-test">Ir a Test Temáticos</button>
        </div>
        <div class="test-option">
            <h2>Test Aleatorios </h2>
            <p>Pon a prueba tus conocimientos con preguntas seleccionadas al azar de todos los temas.</p>
            <button data-enlace="test_aleatorio" class="btn-test">Ir a Test Aleatorios</button>
        </div>
        <div class="test-option">
            <h2>Test de Exámenes </h2>
            <p>Simula un examen real con preguntas similares a las que encontrarás en la prueba oficial.</p>
            <button data-enlace="test_examen" class="btn-test">Ir a Test de Exámenes</button>
        </div>
    </div>
</section>
```



```
<!-- ----- LISTA DE TEST ----- -->
<div> Test Temáticos </div>
<ul class="lista-test-temáticos">
    <li><a href="../../php/test/test temáticos/test tema 1.php"> Señales de tráfico </a></li>
    <li><a href="../../php/test/test temáticos/test tema 1.php"> Prioridad y normas de circulación </a></li>
    <li><a href="../../php/test/test temáticos/test tema 1.php"> Uso de la vía y comportamiento </a></li>
    <li><a href="../../php/test/test temáticos/test tema 1.php"> Velocidades máximas y mínimas </a></li>
    <li><a href="../../php/test/test temáticos/test tema 1.php"> Distancias de seguridad </a></li>
    <li><a href="../../php/test/test temáticos/test tema 1.php"> Alcohol y drogas en la conducción </a></li>
    <li><a href="../../php/test/test temáticos/test tema 1.php"> Uso del cinturón y sistemas de seguridad </a></li>
    <li><a href="../../php/test/test temáticos/test tema 1.php"> Aluminbrado y visibilidad </a></li>
    <li><a href="../../php/test/test temáticos/test tema 1.php"> Maniobras y adelantamientos </a></li>
    <li><a href="../../php/test/test temáticos/test tema 1.php"> Señalización de vehículos </a></li>
    <li><a href="../../php/test/test temáticos/test tema 1.php"> Documentación obligatoria </a></li>
    <li><a href="../../php/test/test temáticos/test tema 1.php"> Mantenimiento del vehículo </a></li>
    <li><a href="../../php/test/test temáticos/test tema 1.php"> Transporte de mercancías y pasajeros </a></li>
    <li><a href="../../php/test/test temáticos/test tema 1.php"> Factores de riesgo en la conducción </a></li>
    <li><a href="../../php/test/test temáticos/test tema 1.php"> Conducción en condiciones adversas </a></li>
    <li><a href="../../php/test/test temáticos/test tema 1.php"> Mecánica básica del automóvil </a></li>
    <li><a href="../../php/test/test temáticos/test tema 1.php"> Normas de circulación para ciclistas y peatones </a></li>
    <li><a href="../../php/test/test temáticos/test tema 1.php"> Primeros auxilios en accidentes </a></li>
</ul>
```



```
<!-- ----- TEST ALEATORIOS ----- -->
<h2> Test Aleatorios </h2>
<ul class="lista-test">
    <li><a href="../../php/test/test aleatorios/test aleatorio 1.php"> Test Aleatorio 1 </a></li>
    <li><a href="../../php/test/test aleatorios/test aleatorio 2.php"> Test Aleatorio 2 </a></li>
    <li><a href="../../php/test/test aleatorios/test aleatorio 3.php"> Test Aleatorio 3 </a></li>
    <li><a href="../../php/test/test aleatorios/test aleatorio 4.php"> Test Aleatorio 4 </a></li>
    <li><a href="../../php/test/test aleatorios/test aleatorio 5.php"> Test Aleatorio 5 </a></li>
    <li><a href="../../php/test/test aleatorios/test aleatorio 6.php"> Test Aleatorio 6 </a></li>
    <li><a href="../../php/test/test aleatorios/test aleatorio 7.php"> Test Aleatorio 7 </a></li>
    <li><a href="../../php/test/test aleatorios/test aleatorio 8.php"> Test Aleatorio 8 </a></li>
    <li><a href="../../php/test/test aleatorios/test aleatorio 9.php"> Test Aleatorio 9 </a></li>
    <li><a href="../../php/test/test aleatorios/test aleatorio 10.php"> Test Aleatorio 10 </a></li>
</ul>
```



```
<!-- ===== TEST EXAMEN ===== -->
<h1> Test de Exámenes Oficiales</h1>
<ul class="lista-test">
<li><a href="../../php/test/test_examen/test_examen_1.php"> Test Examen 1 </a></li>
<li><a href="../../php/test/test_examen/test_examen_1.php"> Test Examen 2 </a></li>
<li><a href="../../php/test/test_examen/test_examen_1.php"> Test Examen 3 </a></li>
<li><a href="../../php/test/test_examen/test_examen_1.php"> Test Examen 4 </a></li>
<li><a href="../../php/test/test_examen/test_examen_1.php"> Test Examen 5 </a></li>
<li><a href="../../php/test/test_examen/test_examen_1.php"> Test Examen 6 </a></li>
<li><a href="../../php/test/test_examen/test_examen_1.php"> Test Examen 7 </a></li>
<li><a href="../../php/test/test_examen/test_examen_1.php"> Test Examen 8 </a></li>
<li><a href="../../php/test/test_examen/test_examen_1.php"> Test Examen 9 </a></li>
<li><a href="../../php/test/test_examen/test_examen_1.php"> Test Examen 10 </a></li>
</ul>
```

Con el fin de hacer el proceso más eficiente, se ha creado un único test que se ajusta de forma dinámica al tipo de examen elegido por el usuario. Por ejemplo, si se selecciona el examen temático, se mostrará únicamente el test del Tema 1 centrado en "Señales de Tráfico". Esta estrategia evita tener que construir una plataforma extensa con un gran volumen de preguntas, ayudas e imágenes, lo cual implicaría un desarrollo largo y complejo. La clave ha sido reutilizar el mismo test base, haciendo consultas distintas desde varios archivos PHP para adaptar el contenido según el tipo de prueba. Así, se logra una solución práctica que permite ofrecer un entorno de aprendizaje claro, organizado y sin necesidad de duplicar esfuerzos en múltiples versiones del examen.

Así que para optimizar la cantidad de hojas que me quedan, vamos hacer referencia solamente a los temáticos ya que, los aleatorios y de examen son los mismo pero con diferentes consultas.

TEST 1: SEÑALES DE TRÁFICO

Tiempo restante: 29:22

Pregunta 1: ¿Qué indica una señal de triángulo con borde rojo y un coche derrepando?

A. Peligro por cierre de carril

B. Obligación de reducir la velocidad

C. Fin de vía prohibida

SIGUIENTE

ANTERIOR	SIGUIENTE	ELIMINAR RESPUESTA				
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
FINALIZAR TEST						

```
:root {
    --color-primario: #1e40af;
    --color-secundario: #60a5fa;
    --color-acento: #ef4444;
    --color-fondo: #f0f4ff;
    --color-blanco: #ffffff;
    --color-gris-claro: #dbeafe;
    --color-texto: #1e293b;
    --sombra-suave: 0 8px 24px rgba(96, 165, 250, 0.25);
    --transicion-suave: all 0.35s cubic-bezier(0.4, 0, 0.2, 1);
    --radio-bordes: 20px;
}
```

El sistema de test temático funciona mediante la combinación entre varios archivos PHP y una base de datos que almacena las preguntas, respuestas y registros de los usuarios. Cuando el usuario accede al archivo **test_tema_1.php**, el sistema realiza una consulta a la base de datos de autoescuela para obtener todas las preguntas del test en la tabla **preguntas_test_temáticos**, junto con sus opciones de respuesta, imágenes asociadas y textos de ayuda (obtenidos de la tabla **ayudas**).

```

<?php
// incluye la conexión
include("C:/xampp/htdocs/Proyecto_Integrado_2025/Tecnologias/php/conexion/conexion.php");

// Verifica conexión
if (!$conexion) {
    die("Error: La conexión a la base de datos no está definida.");
}

// Consulta preguntas
$consulta_id = "SELECT id, numero_preguntas, pregunta, opcion_a, opcion_b, opcion_c FROM preguntas_test_tematicos WHERE tema_id = 1";
$resultado = $conexion->query($consulta);

if ($resultado) {
    die("Error en la consulta de preguntas: " . $conexion->error);
}

$preguntas = $resultado->fetch_all(MYSQLI_ASSOC);

// Consulta ayudas
$consulta_ayuda = "SELECT pregunta_id, texto_ayuda FROM ayudas";
$resultado_ayuda = $conexion->query($consulta_ayuda);

$ayudas = [];
if ($resultado_ayuda) {
    while ($fila = $resultado_ayuda->fetch_assoc()) {
        $ayudas[$fila['pregunta_id']] = $fila['texto_ayuda'];
    }
}
}

```

```

<!DOCTYPE HTML>
<html lang="es">
<head>
    <meta charset="UTF-8" />
    <title>Sistemas de tráfico</title>
    <link rel="stylesheet" href="https://Proyecto_Integrado_2025/Tecnologias/css/test_tematicos.css" />
    <link rel="preconnect" href="https://fonts.googleapis.com" />
    <link rel="preconnect" href="https://fonts.gstatic.com" />
    <link href="https://Proyecto_Integrado_2025/Tecnologias/img/logo-autoscuola.png" type="image/x-icon" rel="icon" />
    <script src="https://Proyecto_Integrado_2025/Tecnologias/js/timer.js" type="text/javascript" />
    <script src="https://Proyecto_Integrado_2025/Tecnologias/js/validacion_test_resuestas.js" type="text/javascript" />
</head>
<body>

<button class="boton-cerrar" onclick="if(confirm('Has a volver al menú de test temáticos y no se va a guardar el intento. (Deseas continuar?)')){location.href='../../..../Tecnologias/html/test_tematicos/menu_test_tematicos.html';} else {alert('Cerrando')}</button>

<div style="text-align:center; font-size:1.2rem; margin-bottom: 1rem;>
    Tiempo restante: <span id="tempo">30</span> segundos
</div>

<div id="pregunta">
    <img alt="Imagen de la pregunta" data-preguntas="1" />
</div>

<form action="https://Proyecto_Integrado_2025/Tecnologias/php/resuestas_usuario/respuesta_usuario_test.php" method="post" id="formularioTest">
    <input type="hidden" name="tempo_id" value="1" />
    <div>
        <input type="radio" name="resposta" value="A" /> A: <?php echo $ayudas[1]; ?>
        <input type="radio" name="resposta" value="B" /> B: <?php echo $ayudas[2]; ?>
        <input type="radio" name="resposta" value="C" /> C: <?php echo $ayudas[3]; ?>
    </div>
</form>

```

```

<?php if (isset($ayudas[$pregunta['id']])): ?>
    <button type="button" class="boton-ayuda" onclick="toggleAyuda(this);"><?php echo $pregunta['id']; ?> Mostrar ayuda</button>
    <div id="ayuda_<?php echo $pregunta['id']; ?>" class="ayuda-texto"><?php echo $ayudas[$pregunta['id']]; ?></div>
</php endif; ?>
</div>
</div>
</div>

<div class="nav-buttons">
    <button type="button" onclick="anteriorPregunta()">< Anterior</button>
    <button type="button" onclick="siguientePregunta()">> Siguiente </button>
    <button type="button" onclick="eliminarRespuesta(this)">< Eliminar respuesta </button>
</div>

<div class="navegacion" id="cuadrosNavegacion"></div>

<div style="text-align:center; margin-top:2rem">
    <button type="submit">Finalizar test</button>
</div>
</form>

<script>
let indiceActual = 0;

function cambiarImagen(numeroPregunta) {
    if (numeroPregunta < 1) numeroPregunta = 1;
    const ruta = "http://localhost/Proyecto_Integrado_2025/Tecnologias/img/test_tematicos/tema_1/" + numeroPregunta + ".png";
    document.getElementById("ImagenPregunta").src = ruta;
}

function mostrarPregunta(indice) {
    const preguntas = document.querySelectorAll('.pregunta');
    preguntas.forEach((p, i) => {
        p.style.display = i === indice ? 'block' : 'none';
    });
}

const numero = preguntas[indice].dataset.numero;
if (numero) cambiarImagen(numero);

const botones = document.querySelectorAll('#cuadrosNavegacion button.cuadro-navegacion');
botones.forEach((bt, i) => {
    if (i === indice) bt.classList.add('active');
    else bt.classList.remove('active');
});
}

</script>

```

Estas preguntas se presentan en un formulario HTML que incluye un temporizador en cuenta regresiva de 30 minutos implementado con **JavaScript**. El temporizador comienza en cuanto se carga la página y, si el tiempo llega a cero antes de que el usuario envíe el formulario, se realiza un envío automático de las respuestas disponibles en ese momento, lo que garantiza que el intento sea registrado incluso si el usuario no actúa.

```

<?php
// Se ejecuta cuando se ha completamente cargado
document.addEventListener("DOMContentLoaded", function() {
    // Inicializa el tiempo total en segundos (30 minutos = 1800 segundos)
    let tiempoRestante = 1800;

    // Inicia el temporizador que se ejecuta cada 1 segundo o 3000 milisegundos
    // Utiliza setinterval para llamar a la función actualizarTemporizador cada segundo
    let intervalo = setInterval(actualizarTemporizador, 3000);

    // Función que actualiza el temporizador cada segundo
    function actualizarTemporizador() {
        // Calcula los minutos dividiendo los segundos totales entre 60
        let minutos = Math.floor(tiempoRestante / 60);

        // Calcula los segundos restantes que no forman un minuto completo
        let segundos = tiempoRestante % 60;

        // Coge el id="tempo" y actualiza su contenido
        // Formatea los minutos y segundos para que solo muestren 2 dígitos (como un reloj normal 09:05)
        const elementoTiempo = document.getElementById("tempo");
        elementoTiempo.textContent = `${minutos.toString().padStart(2, '0'))}:${segundos.toString().padStart(2, '0'))}`;

        // Si el tiempo restante es de 60 segundos o menos, cambia el color del texto a rojo para advertir al usuario
        if (tiempoRestante <= 60) {
            elementoTiempo.style.color = "red";
        }

        // Si aún queda tiempo, lo baja a 1 segundo
        if (tiempoRestante > 0) {
            tiempoRestante--;
        } else {
            // Si el tiempo ha llegado a cero, detiene el intervalo
            clearInterval(intervalo);

            // Muestra una alerta indicando que se agotó el tiempo
            alert("El tiempo agotado! Envíando el test...");

            // Envía automáticamente el formulario con id="formularioTest"
            document.getElementById("formularioTest").submit();
        }
    }
});

```

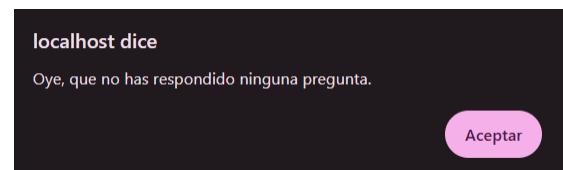
Tiempo restante: **29:57**

Durante la interacción con el formulario, se han implementado validaciones del lado del cliente mediante JavaScript para mejorar la experiencia del usuario y evitar errores comunes. Por ejemplo, si el usuario intenta enviar el test sin haber respondido ninguna pregunta, se muestra un mensaje de advertencia que impide el envío hasta que al menos una respuesta haya sido seleccionada.

```
// Espera a que el contenido del documento esté completamente cargado
document.addEventListener("DOMContentLoaded", function () {
    // Añade un listener al formulario con id "formularioTest" para el evento submit
    document.getElementById("formularioTest").addEventListener("submit", function(event) {
        // Selecciona todas las respuestas marcadas (inputs tipo radio seleccionados)
        const respuestasMarcadas = document.querySelectorAll('input[type="radio"]:checked');

        // Si no hay ninguna respuesta marcada
        if (respuestasMarcadas.length === 0) {
            // Previene el envío del formulario
            event.preventDefault();

            // Muestra una alerta indicando que no se ha respondido ninguna pregunta
            alert("Oye, que no has respondido ninguna pregunta.");
            // Pregunta al usuario si está seguro de enviar el test sin responder
            const confirmar = confirm("¿Estás seguro que quieres enviar el test?");
            if (confirmar) {
                // si confirma, envía el formulario
                this.submit();
            } else {
                // Si cancela, no hace nada más
                return false;
            }
        }
    });
});
```



Asimismo, si el usuario pulsa el botón "Finalizar test" sin haber interactuado con el formulario, se le alerta para confirmar su intención, evitando envíos accidentales. Estas validaciones son clave para asegurar la integridad del intento y mejorar la usabilidad general del sistema.

Una vez que el usuario envía sus respuestas, ya sea manualmente o por agotamiento del tiempo, el formulario las envía mediante método **POST** al archivo **respuestas_usuario_test.php**. Este archivo verifica que el usuario esté autenticado y consulta la base de datos para identificar cuántos intentos previos ha realizado ese usuario en el tema, incrementando el número de intento para el nuevo registro.

```
<?php
session_start();
include("C:/xampp/htdocs/Proyecto_Integrado_2025/Tecnologías/php/conexion/conexion.php");

// Verificación de autenticación del usuario
if (!isset($_SESSION["usuario"])[0]) {
    die("Error: Usuario no autenticado.");
}

$usuario_id = intval($_SESSION["usuario"][0]["id"]);

// Obtener número de intento actual del usuario
$consulta_intento = "SELECT IFNULL(MAX(intento), 0) + 1 AS nuevo_intento FROM respuestas_usuario WHERE usuario_id = ?";
$stmt_intento = $conexion->prepare($consulta_intento);
$stmt_intento->bind_param("i", $usuario_id);
$stmt_intento->execute();
$resultado_intento = $stmt_intento->get_result();
$intento_actual = $resultado_intento->fetch_assoc()["nuevo_intento"];

// Comprobar si se envió el formulario y 'tema_id'
if ($_SERVER['REQUEST_METHOD'] == "POST" && isset($_POST["tema_id"])) {
    $tema_id = intval($_POST["tema_id"]);
    $respuestas_procesadas = 0;

    foreach ($_POST["respuesta"] as $pregunta_id => $respuesta_usuario) {
        $pregunta_id = intval($pregunta_id);

        // Validar que la respuesta existe antes de procesarla
        $consulta = "SELECT respuesta_correcta, tema_id FROM preguntas_test_tematicos WHERE id = ?";
        $stmt = $conexion->prepare($consulta);
        $stmt->bind_param("i", $pregunta_id);
        $stmt->execute();
        $resultado = $stmt->get_result();
        $fila = $resultado->fetch_assoc();

        if ($fila) {
            continue; // Ignora preguntas inexistentes
        }

        $respuesta_correcta = $fila["respuesta_correcta"];
        $tema_id = $fila["tema_id"];
        $se_correcta = ($respuesta_usuario == $respuesta_correcta) ? 1 : 0;
        $fecha_resposta = date("Y-m-d H:i:s");

        // REDIRECTIR AL EXAMEN: Identificar correctamente el test**
        if ($tema_id == 203) {
            $stipo_test = 'examen';
        } elseif ($tema_id >= 101 && $tema_id <= 107) {
            $stipo_test = 'aleatorio';
        } else {
            $stipo_test = 'tematico';
        }
    }
}
```

```
// Insertar respuesta del usuario
$tema_id = $_POST["tema_id"] ?? 101; // Si no existe, establece 101 por defecto - ÚLTIMA MODIFICADA PARA LEA LAS PREGUNTAS DEL TIPO TEMÁTICO Y ALEATORIO.
$insert = "INSERT INTO respuestas_usuario (usuario_id, pregunta_id, respuesta, es_correcta, fecha_resposta, intento, tema_id) VALUES (?, ?, ?, ?, ?, ?, ?)";
$stmt_insert = $conexion->prepare($insert);
$stmt_insert->bind_param("isssiii", $usuario_id, $pregunta_id, $respuesta_usuario, $es_correcta, $fecha_resposta, $intento, $tema_id);
$stmt_insert->execute();

if ($es_correcta) {
    // Insertar el error en errores_usuario
    $insert_error = "INSERT INTO errores_usuario (usuario_id, pregunta_id, respuesta_usuario, respuesta_correcta, intento) VALUES (?, ?, ?, ?, ?)";
    $stmt_error = $conexion->prepare($insert_error);
    $stmt_error->bind_param("isssii", $usuario_id, $pregunta_id, $respuesta_usuario, $respuesta_correcta, $intento);
    $stmt_error->execute();
}

$respuestas_procesadas++;

// Registrar el intento aunque no haya respuestas
if ($respuestas_procesadas == 0) {
    $fecha_resposta = date("Y-m-d H:i:s");
    $insert_vacio = "INSERT INTO respuestas_usuario (usuario_id, intento, fecha_resposta, tema_id) VALUES (?, ?, ?, ?)";
    $stmt_vacio = $conexion->prepare($insert_vacio);
    $stmt_vacio->bind_param("isii", $usuario_id, $intento, $fecha_resposta, $tema_id);
    $stmt_vacio->execute();
}

// Redirigir a la página de resultados
header("Location: ././php/resultados_usuario/resulado.php?usuario_id=" . urlencode($usuario_id));
exit();
} else {
    die("No se recibieron respuestas válidas.");
}
```

A continuación, analiza cada respuesta recibida: si la opción seleccionada por el usuario coincide con la respuesta correcta almacenada en la base de datos, se registra como un acierto; en caso contrario, como un error. Todos los resultados se guardan en la tabla **respuestas_usuario**, incluyendo los datos de usuario, pregunta, fecha, intento y acierto o error. Además, si la respuesta fue incorrecta, también se inserta un registro en la tabla **errores_usuario**, lo cual permite llevar un seguimiento detallado de los errores cometidos.

respuesta_id	usuario_id	pregunta_id	respuesta	es_correcta	fecha_respuesta	intento
1	8	1	B	0	2025-05-21 12:52:03	1
error_id	usuario_id	pregunta_id	respuesta_usuario	respuesta_correcta	intento	
1	8	1	B	A	7	

Incluso en los casos en los que el usuario no responde a ninguna pregunta, el sistema registra igualmente ese intento vacío en la base de datos. Esto permite mantener una trazabilidad completa del historial del usuario y contabilizar cada sesión de evaluación, independientemente del resultado.

Finalizado el procesamiento, el sistema redirige al archivo **resultado.php**, pasando el ID del usuario a través de la URL. En esta página se hace una consulta a la base de datos para mostrar un resumen detallado del intento más reciente. La información incluye el número del intento, el tema, el número de preguntas respondidas, los aciertos, los errores, el porcentaje de aciertos y el estado final (Aprobado o Suspenso).

```
<?php
session_start();
include("C:/xampp/htdocs/Proyecto_Integrado_2025/Tecnologias/php/conexion/conexion.php");

// Validar que "usuario_id" ha sido pasado por URL
if (!isset($_GET["usuario_id"]) && !isset($_GET["alumno_id"])) {
    die("Error: Usuario no especificado.");
}

$usuario_id = isset($_GET["usuario_id"]) ? intval($_GET["usuario_id"]) : intval($_GET["alumno_id"]);

// Verificar conexión con la base de datos
if (!$conexion) {
    die("Error de conexión con la base de datos.");
}

// **Verificación si el usuario tiene test realizados**
$consulta_test = "SELECT COUNT(*) AS total_tests FROM respuestas_usuario WHERE usuario_id = ?";
$stmt_test = $conexion->prepare($consulta_test);
$stmt_test->bind_param("i", $usuario_id);
$stmt_test->execute();
$resultado_test = $stmt_test->get_result();
$filas_test = $resultado_test->fetch_assoc();
$total_tests = ($filas_test["total_tests"] == 0);

// **Consulta original sin modificar**
$consulta_general = "
SELECT r.intentos,
       r.tema_id,
       CASE
           WHEN r.tema_id BETWEEN 201 AND 205 THEN 'Test de Examen'
           WHEN r.tema_id BETWEEN 101 AND 107 THEN 'Test Aleatorio'
           ELSE COALESCE(t_p.nombre_tema, t_r.nombre_tema, 'Sin título')
       END AS nombre_tema,
       COUNT(r.pregunta_id) AS total_preguntas,
       SUM(CASE WHEN r.es_correcta = 1 THEN 1 ELSE 0 END) AS total_aciertos,
       SUM(CASE WHEN r.es_correcta = 0 AND r.pregunta_id IS NOT NULL THEN 1 ELSE 0 END) AS total_errores
FROM respuestas_usuario r
LEFT JOIN preguntas_test_tematicos p ON r.pregunta_id = p.id
LEFT JOIN temas t_p ON p.tema_id = t_p.tema_id
LEFT JOIN temas t_r ON r.tema_id = t_r.tema_id
WHERE r.usuario_id = ?
GROUP BY r.intentos, r.tema_id
ORDER BY r.intentos ASC
";

$stmt_general = $conexion->prepare($consulta_general);
$stmt_general->bind_param("i", $usuario_id);
$stmt_general->execute();
$resultado_general = $stmt_general->get_result();
?>
```

```
<h1>Resultados del Test</h1>

<?php if ($sin_tests) :>
    <p style="text-align:center; font-size:1.2rem; color:#ff4d4d;">
        Aún no ha realizado ningún test.
    </p>
</?php else: >
<table>
    <tr>
        <th>Intento</th>
        <th>Número del Tema</th>
        <th>Número de Preguntas contestadas</th>
        <th>Aciertos</th>
        <th>Errores</th>
        <th>Porcentaje de aciertos</th>
        <th>Estado</th>
    </tr>
<?php while ($fila = $resultado_general->fetch_assoc()):>
<?php
$porcentaje = ($fila["total_preguntas"] > 0)
    ? ($fila["total_aciertos"] / $fila["total_preguntas"]) * 100
    : 0;

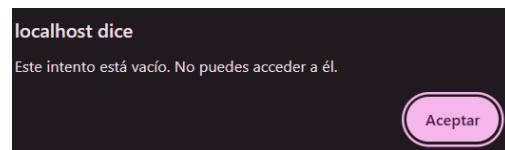
// Lógica de aprobado: mínimo 27 preguntas y 27 aciertos
$estado = "Suspensos";
$clase_estado = "estado-suspensos";
if ($fila["total_preguntas"] >= 27 && $fila["total_aciertos"] >= 27) {
    $estado = "Aprobado";
    $clase_estado = "estado-aprobado";
}
?>
<tr>
    <td>
        <a href="#" title="Ver tus respuestas" onclick="validarIntento(event, <?=$fila['total_preguntas']?>, <?=$fila['intento']?>, <?=$usuario_id?>)">
            <?php htmlspecialchars($fila["intento"]); ?>
        </a>
    </td>
    <td>= htmlspecialchars($fila["numero_tema"]); ?</td>
    <td>= $fila["total_preguntas"]; ?</td>
    <td>= $fila["total_aciertos"]; ?</td>
    <td>= $fila["total_errores"]; ?</td>
    <td>= round($porcentaje, 2); ?</td>
    <td class="<?php echo $clase_estado; ?>"><> $estado </td>
</tr>
<?php endwhile; ?>
</table>
```

87	1	Señales de tráfico	0	0	0	0%	Suspensos
----	---	--------------------	---	---	---	----	-----------

En este punto se aplica la **lógica de aprobado** definida en el sistema: si el usuario ha respondido al menos 27 preguntas y ha acertado al menos 27 de ellas, el estado mostrado será “Aprobado”. De lo contrario, el resultado será “Suspensos”. Este cálculo se realiza directamente en el código **PHP** y se refleja en la tabla de resultados con diferentes estilos visuales (por ejemplo, en color verde para aprobados y rojo para suspensos), facilitando una lectura rápida por parte del usuario o del profesor.

```
// Lógica de aprobado: mínimo 27 preguntas y 27 aciertos
$estado = "Suspensos";
$clase_estado = "estado-suspensos";
if ($fila["total_preguntas"] >= 27 && $fila["total_aciertos"] >= 27) {
    $estado = "Aprobado";
    $clase_estado = "estado-aprobado";
}
```

Cada intento está vinculado a un enlace que permite revisar los detalles de las respuestas. Para evitar errores, si el intento está vacío (sin ninguna pregunta respondida), el sistema muestra un mensaje de advertencia y bloquea el acceso al detalle. Esta verificación se realiza desde **JavaScript**, validando si el total de preguntas respondidas es cero antes de redirigir. En caso contrario, puedes ver los resultados obtenidos con el número de intento.



Detalle del Intento 86			
Número de Pregunta	Pregunta	Tu respuesta	Estado
1	¿Qué indica una señal de tráfico con fondo naranja y un cuadro dorado?	A	Verde
2	¿Qué significa una señal circular con fondo azul y una flecha blanca hacia la derecha?	A	Roja
3	Una señal de tráfico que indica que debes detener tu vehículo.	C	Verde
4	¿Qué indica una señal circular con fondo gris y un cuadro dorado?	B	Roja
5	Una señal de tráfico que indica que debes adelantar a los demás vehículos.	C	Roja
6	¿Qué indica una señal que muestra una flecha blanca sobre fondo azul?	C	Verde

Con esta explicación hemos terminado como funcionan los test, desde la parte visual y lógica. Ahora si continuamos con la parte de los usuarios, en su área encontramos que también pueden que han comprado y además, los resultados que han obtenido en los test realizados (esta parte ya ha sido explicada en la parte de los test). Vamos a centrarnos en como pueden sus compras.

Cuando accede a las compras, el usuario entra en este HTML.



```
<!-- ----- PERFIL DE LAS COMPRAS ----- -->
<h1><!--titulo-compras-->Bienvenido a tu perfil de compras</h1>
<main>
  <section id="compras-usuario">
    <h2>Mis Compras</h2>
    <div id="lista-compras">Cargando compras...</div>
  </section>
</main>

<script>
document.addEventListener("DOMContentLoaded", function () {
  fetch("http://localhost/Proyecto_Integrado_2025/Tecnologias/php/verificar_usuario.php?verificar_usuario.php")
    .then(response => response.json())
    .then(data => {
      let lista = document.getElementById("lista-compras");
      if (data.session_activ == true && data.productos_comprados.length > 0) {
        lista.innerHTML = "<ul>";
        // Mostrar productos tal como están en la respuesta JSON, sin eliminar duplicados
        data.productos_comprados.forEach((producto, index) => {
          lista.innerHTML += "<li> 
```

Al cargar la página, se ejecuta un script que realiza una petición (**fetch**) a un archivo **PHP** ubicado en el servidor (**verificar_usuario.php** ya explicado). Este archivo devuelve un objeto **JSON** con dos datos clave:

- Si la sesión del usuario está activa (**sesion_activ**).
- Un listado de los productos comprados (**productos_comprados**).

Si hay productos en la respuesta, se generan elementos **** para mostrarlos en una lista dentro del contenedor **#lista-compras**. Cada producto aparece con un ícono de verificación y su número de orden. En caso de que no haya compras registradas, se muestra un mensaje indicando que el usuario aún no ha comprado nada. Además, si ocurre algún error al consultar los datos, se muestra un mensaje de error en pantalla.

El usuario después de registrarse, iniciar sesión, comprar los tests, ver sus compras, hacer tests y ver sus resultados, ya sólo le faltaría ver los eventos que están en el calendario y dejar una reseña en la gran autoescuela a la que se ha apuntado.

Vamos a empezar con el calendario explicando que es lo que hace, de qué está compuesto y por qué es importante para los alumnos y profesores.

El calendario se ejecuta cuando la página carga completamente gracias al evento **DOMContentLoaded**.

Dentro de este evento, se inicializan las variables del DOM, como el contenedor de días (.dias), el encabezado del mes (.encabezado h2) y los botones de navegación.

También se define el array **nombresMeses** para mostrar correctamente los nombres de los meses, y se carga la lista de eventos almacenados en **localStorage**.

La función principal es **generarCalendario**, encargada de construir el calendario de forma dinámica. Primero limpia el contenido anterior, luego calcula el número de días del mes, determina en qué día de la semana empieza, y crea celdas vacías si es necesario para alinear correctamente los días. Recorre todos los días del mes y crea elementos visuales (div) para cada uno, añadiendo clases especiales si es el día actual. Además, verifica si hay eventos para ese día y, si existen, los muestra dentro del día correspondiente con el color definido. Cada día es clicable y al hacer clic ejecuta **abrirModalEvento**.

La función **cambiarMes** permite cambiar el mes actual y también ajusta el año si es necesario (por ejemplo, si se pasa de diciembre a enero o viceversa). Esta función se activa mediante los botones de navegación del calendario.

La función **abrirModalEvento** se ejecuta cuando el usuario hace clic sobre un día del calendario. Esta función filtra los eventos que coincidan con esa fecha y, si existen, muestra sus detalles mediante una alerta (alert).

Una de las funciones más importantes es **añadirEvento**, la cual se activa desde un botón flotante que muestra un menú de opciones. Dentro de esta función, se le pregunta al usuario si desea añadir un evento. Luego solicita el **título** del evento, y solo se aceptan opciones como “Examen práctico”, “Curso intensivo”, “Cumpleaños”, “Clase teórica” o “Otro”. Después pide la **fecha** en formato dd/mm/yyyy, la cual es validada por la función **validarFecha**, que se encarga de comprobar que la fecha sea válida (existente en el calendario). Luego solicita un **color** para el evento, limitado a ciertos colores específicos (rojo, verde, azul, naranja, morado, rosa, marrón, celeste o gris). Si el título del evento es “Examen práctico” y cae en fin de semana, se cancela el evento mediante una validación que verifica si el día corresponde a sábado o domingo.

Además, para evitar saturación en un solo día, se valida que no haya más de dos eventos ya programados para esa fecha. Si todo es correcto, se crea un objeto con los datos del evento y se agrega al arreglo eventos. Luego, se guarda el arreglo actualizado en **localStorage** y se ejecuta **generarCalendario** de nuevo para reflejar el nuevo evento en pantalla.

La función **validarFecha** toma una cadena con formato dd/mm/yyyy, la divide, y utiliza el constructor Date para comprobar si la fecha es válida, devolviendo true o false.

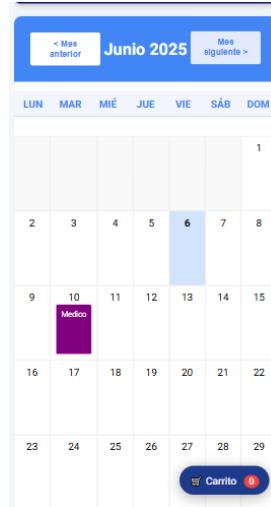
Por último, el calendario incluye un botón flotante con funcionalidad para mostrar y ocultar el menú de acciones. Este botón cambia de clase y visibilidad mediante un **addEventListener** que alterna clases como **mostrar** y **mostrar-menu**.

```
<!-- CALENDARIO ----->
<div class="calendario">
  <div class="encabezado">
    <div class="botones-mes">
      <button></button> Mes anterior </button>
    </div>
    <div>Junio 2025</div>
    <div class="botones-mes">
      <button>Mes siguiente &gt;</button>
    </div>
  </div>

  <div class="dias-semana">
    <div>Jue</div> <div>Vie</div> <div>Sáb</div> <div>Dom</div>
  </div>

  <div class="dias">
    <div class="vacio"></div> <div class="vacio"></div> <div class="vacio"></div>
    <div>1</div> <div>2</div> <div>3</div> <div>4</div>
    <div>5</div> <div>6</div> <div>7</div> Examen teórico
    <div>8</div> <div>9</div> Examen práctico
    <div>10</div> <div>11</div> Cursillo intensivo
    <div>12</div> <div>13</div> Examen teórico
    <div>14</div> <div>15</div> Examen práctico
    <div>16</div> <div>17</div> Cursillo intensivo
    <div>18</div>
    <div>19</div> Examen teórico
    <div>20</div> Complejada
    <div>21</div>
    <div>22</div> Examen práctico
    <div>23</div>
    <div>24</div> Cursillo intensivo
    <div>25</div>
    <div>26</div> Examen teórico
    <div>27</div>
    <div>28</div>
    <div>29</div> Examen
    <div>30</div> medico
    <div>31</div>
  </div>
</div>
<!-- Botón solamente visible para profesores -->
<php if ($esProfesor): >
  <div class="boton-flotante">
    <div>
      <button id="menuAcciones" type="button">≡</button>
      <button id="btñAñadir" type="button">Añadir</button>
      <button id="btñBorrar" type="button">Borrar</button>
      <button id="btñModificar" type="button">Modificar</button>
    </div>
  </div>
<php endif; >
<!-- FOOTER O PIE DE PAGINA ----->
```

```
:root {
  --ancho-maximo: 1000px;
  --color-fondo: #ffffff;
  --radio-borde: 12px;
  --sombra-caja: 0 4px 16px #rgba(0, 0, 0, 0.1);
  --color-primario: #4285f4;
  --color-primario-oscuro: #357aa8;
  --color-primario-mas-oscuro: #2c66d4;
  --color-primario-claro: #e3eaf3;
  --color-texto-blanco: #fff;
  --color-texto-primario: #4285f4;
  --color-fondo-semana: #ef2f6c;
  --color-borde-semana: #e0e0e0;
  --color-borde-dia: #eee;
  --color-hover-dia: #ff0000;
  --color-hoy: #d2e3fc;
  --color-vacio: #ffff99;
  --color-evento: #ffcc00;
  --color-evento-rojo: #red;
  --color-evento-verde: #green;
  --color-evento-oro: #gold;
  --color-evento-azul: #blue;
  --color-evento-morado: #purple;
  --color-botón-Flotante: #007bff;
  --color-sombra-botón: 0 4px 8px #rgba(0, 0, 0, 0.3);
  --color-botón-evento: #aca5f0;
  --color-botón-evento-hover: #454649;
  --color-botón-evento-activo: #388e3c;
  --color-borde-menu: #ccc;
  --color-fondo-menu: #fff;
  --color-hover-menu: #ff00ff;
}
```



```
document.addEventListener("DOMContentLoaded", function () {
  const Calendario = document.querySelector("#calendario");
  const mesActual = document.querySelector(".encabezado h2");
  const botonesMes = document.querySelectorAll(".botones-mes button");
  const diasMes = document.querySelectorAll(".dias div");
  const menuAcciones = document.getElementById("menuAcciones");

  let eventos = JSON.parse(localStorage.getItem("eventos")) || [];
  let añoActual = 2025;
  let mesActualIndex = 5; // Estamos en junio (mes 5, ya que enero es 0) como los array empiezan en 0

  const meses = [
    "Enero", "Febrero", "Marzo", "Abril",
    "Mayo", "Junio", "Julio", "Agosto",
    "Septiembre", "Octubre", "Noviembre", "Diciembre"
  ];

  function formatearFechaCorrectamente(año, mes, dia) {
    return `${año}-${mes}-${dia}`.padStart(4, "0");
  }

  function generarCalendario(año, mes) {
    calendario.innerHTML = "";
    mesActual.textContent = `(mes ${mes}) ${año}`;
    const diaMes = new Date(año, mes, 1).getDay();
    const diasMes = new Date(año, mes + 1, 0).getUTCDate();
    for (let i = 0; i < (primerDia === 0 ? 0 : 1) + (mes - 1); i++) {
      const divVacio = document.createElement("div");
      divVacio.classList.add("vacio");
      calendario.appendChild(divVacio);
    }
    const hoy = new Date();
    const estoyMes = hoy.getFullYear() === año && hoy.getMonth() === mes;
    for (let dia = 1; dia <= diasMes - diaMes; dia++) {
      const divDia = document.createElement("div");
      divDia.textContent = dia;
      divDia.classList.add("dia");
      if (estoyMes && hoy.getDate() === dia) {
        divDia.classList.add("hoy");
      }
      const fechaFormatada = formatearFechaCorrectamente(año, mes, dia);
      const eventosDelDia = eventos.filter(e => e.fecha === fechaFormatada);
      eventosDelDia.forEach(ev => {
        const eventoElemento = document.createElement("div");
        eventoElemento.classList.add("evento");
        eventoElemento.style.backgroundColor = ev.color;
        eventoElemento.textContent = ev.titulo;
        divDia.appendChild(eventoElemento);
      });
      divDia.addEventListener("click", () => abrirModalEvento(año, mes + 1, dia));
      calendario.appendChild(divDia);
    }
  }
});
```

```
function cambiarMes(dirección) {
  mesActualIndex += dirección;
  if (mesActualIndex < 0) {
    mesActualIndex = 11;
    añoActual--;
  } else if (mesActualIndex > 11) {
    mesActualIndex = 0;
    añoActual++;
  }
  generarCalendario(añoActual, mesActualIndex);
}

botonesMes[0].addEventListener("click", () => cambiarMes(-1));
botonesMes[1].addEventListener("click", () => cambiarMes(1));

function toggleMenu() {
  menuAcciones.classList.toggle("visible");
}

botónFlotante.addEventListener("click", function (e) {
  if (e.target === botónFlotante || e.target.textContent.trim() === "≡") {
    toggleMenu();
  }
});

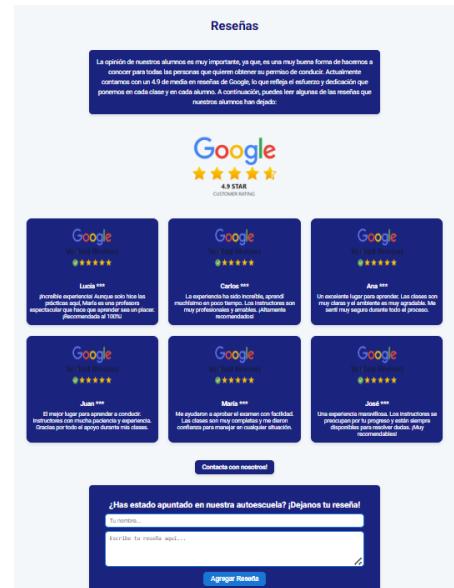
document.addEventListener("click", function (event) {
  if (!botónFlotante.contains(event.target) && !menuAcciones.contains(event.target)) {
    menuAcciones.classList.remove("visible");
  }
});

function abrirModalEvento(año, mes, dia) {
  const fechaFormatada = formatearFechaCorrectamente(año, mes, dia);
  const eventosDelDia = eventos.filter(e => e.fecha === fechaFormatada);
  if (eventosDelDia.length > 0) {
    const lista = eventosDelDia.map(e => ` - ${e.titulo}`).join("\n");
    alert(`Eventos del ${fechaFormatada}:\n${lista}`);
  } else {
    alert(`No hay eventos para el ${dia} de ${meses[mes - 1]} ${año}`);
  }
}
```

Dada la explicación de como funciona el calendario, hay que diferenciar en que el calendario tenemos la parte visual de los alumnos y de los profesores, me explico. En la parte de los alumnos, tenemos el calendario como tal pero sin poder añadir, modificar o borrar eventos, es decir, sin el botón de las acciones y en las reseñas pasa igual. En los profesores si aparecerá el botón para borrar las reseñas pero para los alumnos, no.

De este modo controlamos que la parte del usuario sea solamente visual mientras que para los profesores, tengan acciones.

Dicho esto, esta es la parte de las reseñas:



```
:root {
    --color-fondo: #ffffff;
    --color-principal: #1a237e;
    --color-secundario: #1976d2;
    --color-texto: #212121;
    --color-hover: #e3f2fd;
    --color-sombra: rgba(0, 0, 0, 0.1);
    --color-sombra-hover: rgba(0, 0, 0, 0.2);
    --tamano-fuente-titulo: 36px;
    --tamano-fuente-subtitulo: 24px;
    --tamano-fuente-parrafo: 16px;
    --tamano-fuente-pequeno: 18px;
    --tamano-fuente-grande: 28px;
    --radio-borde: 10px;
    --sombra-caja: 0 4px 8px rgba(0, 0, 0, 0.1);
    --sombra-caja-hover: 0 8px 15px rgba(0, 0, 0, 0.15);
    --relleno: 20px;
    --margen-superior: 50px;
    --relleno-pequeno: 10px;
    --relleno-grande: 30px;
    --radio-borde-grande: 15px;
}
```

```
<a data-enlace="contacto" id="boton-contactar">Contacta con nosotros</a>

<!-- ===== AGREGAR RESEÑA ===== -->

<div id="agregar-reseña">
    <h2>Has estado apuntado en nuestra autoescuela? ¡Déjanos tu reseña!</h2>
    <input type="text" id="nombre-usuario" placeholder="Tu nombre..."/>
    <br>
    <textarea id="nueva-reseña" placeholder="Escribe tu reseña aquí... rows="4" cols="50"></textarea>
    <br>
    <button id="boton-agregar-reseña" onclick="agregarReseña()">Agregar Reseña</button>
</div>
```

```
<div id="etiquetas-reseñas">
    <div id="etiqueta-reseñas">
        <input checked="" type="checkbox"/> Los profesores son muy importantes, ya que, en una muy buena forma de haceremos a conocer para todas las personas que quieren obtener su permiso de conducir. Actualmente contamos con un 4.9 de media en Google, lo que refleja el esfuerzo y dedicación que ponemos en cada clase. A continuación, puedes leer algunas de las reseñas que nuestros alumnos han dejado:</div>
        <div id="etiqueta-reseñas">
            <img alt="estrella-google" width="4.9 estrellas" height="200"/>
        </div>
        <div id="etiquetas-reseñas">
            <div id="etiqueta-reseñas">
                <img alt="reseña-verificada" width="100" height="100"/>
                <div>Lucía ***</div>
                <div>La experiencia que hace que aprender sea un placer. ¡Recomendadísimo!</div>
            </div>
            <div id="etiqueta-reseñas">
                <img alt="reseña-verificada" width="100" height="100"/>
                <div>Carlos ***</div>
                <div>La experiencia ha sido increíble, aprendí muchísimo y con mucha paciencia. Los profesores son muy profesionales y amables, ¡altamente recomendado!</div>
            </div>
            <div id="etiqueta-reseñas">
                <img alt="reseña-verificada" width="100" height="100"/>
                <div>Ana ***</div>
                <div>Un excelente lugar para aprender. Los chicos son muy amables y profesionales. Me sentí muy segura durante todo el proceso.</div>
            </div>
            <div id="etiqueta-reseñas">
                <img alt="reseña-verificada" width="100" height="100"/>
                <div>Juana ***</div>
                <div>El mejor lugar para aprender a conducir. Muchísima paciencia y experiencia. Gracias por todo el apoyo durante mis clases.</div>
            </div>
            <div id="etiqueta-reseñas">
                <img alt="reseña-verificada" width="100" height="100"/>
                <div>María ***</div>
                <div>Me ayudaron a aprobar el examen con facilidad. Las clases son muy completas y me dieron confianza para manejar en cualquier situación.</div>
            </div>
            <div id="etiqueta-reseñas">
                <img alt="reseña-verificada" width="100" height="100"/>
                <div>José ***</div>
                <div>Una experiencia maravillosa. Los instructores se preocupan por tu progreso y están siempre disponibles para resolver dudas. ¡Muy recomendable!</div>
            </div>
        </div>
        <div id="etiquetas-reseñas">
            <div id="etiqueta-reseñas">
                <img alt="reseña-verificada" width="100" height="100"/>
                <div>Sofía ***</div>
                <div>Un gran lugar para aprender a conducir. Muy buena formación y mucha paciencia y experiencia. Gracias por todo el apoyo durante mis clases.</div>
            </div>
            <div id="etiqueta-reseñas">
                <img alt="reseña-verificada" width="100" height="100"/>
                <div>Pilar ***</div>
                <div>Un gran lugar para aprender a conducir. Los profesores son muy amables y profesionales. Me sentí muy segura durante todo el proceso.</div>
            </div>
            <div id="etiqueta-reseñas">
                <img alt="reseña-verificada" width="100" height="100"/>
                <div>Daniela ***</div>
                <div>Un gran lugar para aprender a conducir. Los profesores son muy amables y profesionales. Me sentí muy segura durante todo el proceso.</div>
            </div>
            <div id="etiqueta-reseñas">
                <img alt="reseña-verificada" width="100" height="100"/>
                <div>Juan ***</div>
                <div>Un gran lugar para aprender a conducir. Los profesores son muy amables y profesionales. Me sentí muy segura durante todo el proceso.</div>
            </div>
        </div>
    </div>
</div>
```

```
// Función para agregar una nueva reseña
function agregarReseña() {
    // Obtiene el nombre del usuario y el texto de la reseña desde los inputs
    const nombreUsuario = document.getElementById("nombre-usuario").value.trim();
    const reseñaTexto = document.getElementById("nueva-reseña").value.trim();

    // Verifica que ambos campos estén completos
    if (nombreUsuario === "" || reseñaTexto === "") {
        alert("Por favor, completa todos los campos antes de enviar tu reseña.");
        return;
    }

    // Verifica si la reseña contiene palabras prohibidas
    if (containsProhibitedWords(reseñaTexto)) {
        alert("Tu reseña contiene palabras inapropiadas. Intentalo de nuevo.");
        return;
    }

    // Agrega el nombre y apellido y coloca el apellido con asteriscos
    const nombreApellido = nombreUsuario + " " + apellido(reseñaTexto);
    const nombreConApellidos = apellido.length > 0 ? `${nombre} *${apellido}` : nombre;

    // Recopila las reseñas guardadas, agrega la nueva y guarda en localStorage
    const reseñas = JSON.parse(localStorage.getItem("reseñas")) || [];
    reseñas.push({ nombre: nombreConApellidos, texto: reseñaTexto });
    localStorage.setItem("reseñas", JSON.stringify(reseñas));

    // Muestra la nueva reseña en pantalla
    mostrarReseña(nombreConApellidos, reseñaTexto);

    // Limpia los campos del formulario
    document.getElementById("nombre-usuario").value = "";
    document.getElementById("nueva-reseña").value = "";

    alert("¡Gracias por dejarnos tu opinión!");
}

// Función para mostrar una reseña en pantalla
function mostrarReseña(nombre, texto) {
    // Crea un nuevo div para la reseña
    const divReseña = document.createElement("div");
    divReseña.classList.add("etiqueta-reseña");

    // Agrega un id único para el texto de la reseña
    const idReseña = `texto-reseña-${Date.now()}`;
    divReseña.innerHTML = `

${texto}



Borrar



Editar

`;

    // Inserta el contenido HTML de la reseña, incluyendo botones de borrar y editar
    nuevoEtiquetas.innerHTML += divReseña.outerHTML;

    // Agade la reseña al contenedor de reseñas en el DOM
    document.getElementById("etiquetas").appendChild(nuevoEtiquetas);
}
```

Reseñas.php está diseñada para mostrar opiniones de alumnos sobre su experiencia en la autoescuela, con el objetivo de generar confianza en futuros clientes. Al principio se muestra un título y una breve introducción que destaca la valoración media en Google (en modo imagen), luego se presentan varias reseñas estáticas con el nombre del alumno parcialmente oculto y un texto que describe su experiencia.

El archivo JavaScript **borrar_reseñas.js** se encarga de manejar el funcionamiento dinámico de esta sección. Al cargar la página, recupera las reseñas guardadas en el navegador mediante **localStorage** y las muestra automáticamente. También permite añadir nuevas reseñas, verificando que los campos estén completos y que no contengan palabras inapropiadas mediante un filtro de palabras prohibidas.

```
// Lista de palabras prohibidas para filtrar reseñas inapropiadas
const palabrasProhibidas = [
    // Masculino singular
    "tonto", "gilipollas", "subnormal", "idiotaz", "imbécil", "estúpido", "inútil", "payaso", "capullo", "bobo",
    "cretino", "tarado", "roqueta", "burro", "bestia", "patán", "necio", "memó", "pelmazo", "cenutrión", "majaderos",
    "miserable", "asqueroso", "cerdo", "baboso", "maldito", "malnacido", "desgraciado", "cabron", "hijo de puta", "perro",
    // Masculino plural
    "tontos", "gilipollas", "subnormales", "idiotas", "imbéciles", "estúpidos", "inutiles", "payosos", "capullos", "bobos",
    "cretinos", "tarados", "roquetas", "burros", "bestias", "patanes", "necios", "memos", "pelmazos", "cenutrios", "majaderos",
    "miserables", "asquerosos", "cerdos", "babosos", "malditos", "malnacidos", "desgraciados", "cabrones", "hijos de puta", "perros",
    // Feminino singular
    "cretina", "gilipollana", "subnormal", "idiotita", "imbécila", "estúpida", "inútil", "payasa", "capulla", "boba",
    "cretina", "tarada", "roqueta", "burra", "bestia", "patana", "necia", "memá", "pelmaza", "cenutria", "majadera",
    "miserable", "asquerosa", "cerda", "babosa", "maldita", "malnacida", "desgraciada", "cabrona", "hija de puta", "perra",
];
};
```

Al agregar una reseña, se oculta el apellido con asteriscos y se guarda la información en **localStorage**, asegurando que se mantenga aunque se recargue la página. Si se pulsa el botón de borrar, se elimina la reseña correspondiente, tanto visualmente como del almacenamiento local.

Todo este sistema permite que los usuarios participen activamente compartiendo su experiencia y que los profesores tengan control sobre el contenido mostrado. Además, la sección de reseñas utiliza archivos **JSON** para almacenar los comentarios de los usuarios de forma estructurada y organizada. Cada reseña se guarda como un objeto con propiedades como el nombre del alumno y su opinión. Estos objetos se convierten a formato **JSON** y se guardan en el navegador con **localStorage**. Al cargar la página, el JavaScript lee ese **JSON**, lo convierte nuevamente en objetos y muestra las reseñas automáticamente. Usar **JSON** permite guardar varias reseñas en un solo lugar y facilita su lectura, modificación y eliminación de manera sencilla y rápida desde el código.

Con esta última explicación queda concluida la parte de los alumnos en cuanto **Manual de Usuarios** y **Manual Técnico**, sólo nos quedaría la parte de los profesores, que vamos a describir de manera muy clara.

5.1.2 Parte Profesores.

Cuando un profesor quiere acceder a su área, tiene que iniciar sesión en **Área de Profesores**, desde el menú de navegación o nav.

Tengo que destacar que el iniciar sesión de los profesores, es prácticamente igual que el de los alumnos pero con diferente consulta para la base de datos. En vez de consultar la tabla de **Clientes**, lo hace con la de **Profesores** y obviamente, la consulta es diferente.

Login de profesores

Correo:

Contraseña:

Recordarme

[Olvidaste tu contraseña?](#)

```
:root {
    --color-principal: #005f73;
    --color-secundario: #94d2bd;
    --color-fondo-formulario: #f0f4f8;
    --color-borde: #ccc;
    --color-texto: #333;
    --color-botón: #0a9396;
    --color-botón-hover: #007ff8;
    --radio-borde: 10px;
    --espaciado: 15px;
    --sombra: 0 4px 10px rgba(0, 0, 0, 0.1);
    --fuente: 'Reboto', sans-serif;
}
```

```
<?php
header("Content-Type: application/json");
include("../xampp/htdocs/Proyecto_Integrado_2025/Tecnologias/php/conexion/conexion.php");
error_reporting(E_ALL);
ini_set("display_errors", 1);

$datos = json_decode(file_get_contents("php://input"), true);
if (!($datos)) {
    echo json_encode(["error" => "No se han recibido datos correctamente."]);
    exit();
}

$correo = $datos["correo"] ?? "";
$contraseña = $datos["contraseña"] ?? "";

// Consultar la base de datos
$stmt = $conexion->prepare("SELECT contraseña FROM profesores WHERE correo = ?");
$stmt->bind_param("s", $correo);
$stmt->execute();
$resultado = $stmt->get_result();

if ($resultado->num_rows === 1) {
    $profesor = $resultado->fetch_assoc();

    if (password_verify($contraseña, $profesor["contraseña"])) {
        echo json_encode(["validación" => "correcta"]);
    } else {
        echo json_encode(["error" => "Contraseña incorrecta."]);
    }
} else {
    echo json_encode(["error" => "Correo no registrado."]);
}

exit();
>

<!-- Este verificar es para verificar que tengo la conexión con la base de datos para que las contraseñas y los correos correctos -->
```

No podemos olvidarnos de validar las sesiones de profesores antes y con las validaciones en JavaScript o lado Cliente (que son prácticamente que la de los alumnos) que voy a explicar a continuación junto el PHP o lado Servidor:

```
<?php
session_start();
include("../xampp/htdocs/Proyecto_Integrado_2025/Tecnologias/php/conexion/conexion.php");

// Si el profesor no tiene sesión activa, evitar que vuelva a iniciar sesión
if (!isset($_SESSION['profesor_id'])) {
    if ($_SESSION['profesor_nombre'] === "Juan") {
        header("Location: ../../area_profesor/area_profesor.php");
    } elseif ($_SESSION['profesor_nombre'] === "María") {
        header("Location: ../../area_profesora/area_profesora.php");
    }
    exit();
}

// Bloquear el acceso al login de profesores si ya hay una sesión activa como alumno
if (isset($_SESSION['usuario_id'])) {
    header("Location: ../../area_alumnos/area_alumnos.php?error=Ya_tienes_sesion");
    exit();
}

// Verificar que se recibió una solicitud POST antes de procesar el login
if ($_SERVER['REQUEST_METHOD'] === "POST") {
    $correo = $_POST['correo'];
    $contraseña = $_POST['contraseña'];

    // Obtener los datos del profesor
    $stmt = $conexion->prepare("SELECT id, nombre, tipo_clase, contraseña FROM profesores WHERE correo = ?");
    $stmt->bind_param("s", $correo);
    $stmt->execute();
    $resultado = $stmt->get_result();

    if ($resultado->num_rows === 1) {
        $profesor = $resultado->fetch_assoc();

        // Verificar la contraseña escrita
        if (password_verify($contraseña, $profesor["contraseña"])) {
            // Establecer sesión correctamente
            $_SESSION['profesor_id'] = $profesor['id'];
            $_SESSION['profesor_nombre'] = $profesor['nombre'];
            $_SESSION['tipo_clase'] = $profesor['tipo_clase'];

            // Asegurar que la sesión dura más tiempo
            setcookie(session_name(), session_id(), time() + 86400, "/");

            // Redirigir automáticamente según el profesor
            if ($_SESSION['profesor_nombre'] === "Juan") {
                header("Location: ../../area_profesor/area_profesor.php");
            } elseif ($_SESSION['profesor_nombre'] === "María") {
                header("Location: ../../area_profesora/area_profesora.php");
            } else {
                echo "Error: Profesor no identificado.";
            }
            exit();
        } else {
            echo "Error: Contraseña incorrecta.";
        }
    } else {
        echo "Error: Correo no registrado.";
    }
} else {
    echo "Error: Método de solicitud no válido.";
}
>
```

```
document.addEventListener("DOMContentLoaded", function () {
    // Obtenemos referencias al formulario y a los campos de entrada
    const formulario = document.getElementById("formulario-login");
    const inputCorreo = document.getElementById("correo");
    const inputContraseña = document.getElementById("contraseña");

    // Referencias a los elementos donde mostraremos los errores (ya separados)
    const mensajeErrorCorreo = document.getElementById("mensaje-error-correo");
    const mensajeErrorContraseña = document.getElementById("mensaje-error-contraseña");

    // Añadimos un manejador de evento para cuando el formulario se intente enviar
    formulario.addEventListener("submit", function (event) {
        event.preventDefault(); // Prevendremos el envío por defecto del formulario

        // Obtenemos los valores introducidos por el usuario, quitando espacios
        const correo = inputCorreo.value.trim();
        const contraseña = inputContraseña.value.trim();

        // Enviamos los datos al servidor utilizando fetch (AJAX)
        fetch("../php/verifica_login_profs/verifica_login_profs.php", {
            method: "POST",
            headers: { "Content-Type": "application/json" },
            body: JSON.stringify({ correo, contraseña }) // Enviamos los datos como JSON
        })
        .then(response => response.text()) // Recibimos la respuesta como texto
        .then(texto => {
            try {
                const data = JSON.parse(texto); // Intentamos convertir el texto a objeto JS

                // Limpiamos mensajes de error anteriores
                mensajeErrorCorreo.textContent = "";
                mensajeErrorContraseña.textContent = "";

                if (data.error) {
                    // Si hay un error, lo mostramos en el campo correspondiente
                    if (data.error.includes("correo")) {
                        mensajeErrorCorreo.textContent = `▲ ${data.error}`;
                        mensajeErrorCorreo.style.color = "red";
                    } else if (data.error.includes("contraseña")) {
                        mensajeErrorContraseña.textContent = `▲ ${data.error}`;
                        mensajeErrorContraseña.style.color = "red";
                    } else {
                        // Si el error no es específico, lo mostramos en ambos
                        mensajeErrorCorreo.textContent = `▲ ${data.error}`;
                        mensajeErrorCorreo.style.color = "red";
                        mensajeErrorContraseña.textContent = `▲ ${data.error}`;
                        mensajeErrorContraseña.style.color = "red";
                    }
                } else {
                    // Si no hay errores, se considera login válido y se envía el formulario
                    formulario.submit();
                }
            } catch (error) {
                // Si hay un error al procesar el JSON, mostramos un mensaje genérico
                console.error("Error al procesar JSON:", error);
                mensajeErrorCorreo.textContent = "";
                mensajeErrorContraseña.textContent = `▲ Error inesperado en el servidor.`;
                mensajeErrorContraseña.style.color = "red";
            }
        })
        .catch(error => {
            // Si no se puede conectar con el servidor, mostramos un mensaje
            console.error("Error en la conexión", error);
            mensajeErrorCorreo.textContent = "";
            mensajeErrorContraseña.textContent = `▲ No se pudo conectar con el servidor.`;
            mensajeErrorContraseña.style.color = "red";
        });
    });
});
```

En primer lugar **verificar_login_profs.php** comprueba si el correo y la contraseña que escribe el profesor son correctos. Recibe los datos en formato JSON que le manda el formulario. Luego, busca en la base de datos si hay un profesor con ese correo. Si lo encuentra, revisa si la contraseña que escribió coincide con la que está guardada. Si todo está bien, responde con un mensaje diciendo que los datos son correctos.

Si hay algún error, como que el correo no existe o la contraseña no es la correcta, también manda un mensaje pero indicando cuál es el problema.

El archivo **validaciones_login_profes.js** es el que se encarga de revisar los datos del formulario en el navegador antes de que se envíen. Cuando el profesor escribe su correo y contraseña, este archivo toma esos datos y los manda a **verificar_login_profes.php** usando **fetch**. Espera una respuesta y, si hay algún error, muestra un mensaje en pantalla, como por ejemplo: "Correo no registrado" o "Contraseña incorrecta". Si no hay errores, deja que el formulario se envíe normalmente.

Después, entra en acción el archivo **validar_login_profesores.php**, que se encarga de iniciar la sesión de verdad. Primero, revisa si ya hay una sesión activa como profesor o como alumno. Si ya hay sesión, no deja volver a iniciar y redirige a la zona que le corresponde. Si no hay sesión y llegan los datos del formulario, busca otra vez al profesor en la base de datos y revisa la contraseña. Si todo está correcto, guarda en la sesión datos del profesor, como su nombre, su ID y el tipo de clase. Luego, según el nombre del profesor, lo manda a su zona personalizada. Por ejemplo, si se llama Juan, lo lleva a **area_profesor.php**; si se llama María, lo lleva a **area_profesora.php**. Si el nombre no es ninguno de esos, muestra un error.

Una vez que el profesor ha iniciado sesión, puede entrar a su zona privada. Por ejemplo, el profesor Juan, que tiene el ID 1, puede ver cosas especiales, como los resultados de los test de los alumnos y las clases prácticas que tiene cada uno de sus alumnos.



```
<?php endif; ?>
</header>
<?php Alumnos de <?php htmlspecialchars($_SESSION['profesor_nombre']); ?> (Prácticas)</?>
<?php Selecciona un alumno para ver su información</?>


<select id="selección-alumno">
<option value="selección-alumno" selected="selected">Seleccionar Alumno -></option>
<?php foreach ($alumnos as $fila) { ?>
<option value="alumno-<?php echo $fila['id']; ?>"><?php echo htmlspecialchars($fila['nombre']); ?> -> . <?php echo $fila['apellidos']; ?></option>
</?php } ?>
</select>


<?php foreach ($alumnos as $fila) { ?>


<?php echo htmlspecialchars($fila['nombre']); ?> -> . <?php echo $fila['apellidos']; ?></div>
<?php if ($fila['telefono']) { ?>
<?php echo htmlspecialchars($fila['telefono']); ?></div>
<?php } ?>
<?php if ($fila['fecha_inscripcion']) { ?>
<?php echo date_format(date_create($fila['fecha_inscripcion']), 'd/m/Y'); ?></div>
<?php } ?>
<?php if ($fila['correo_inscripto']) { ?>
<?php echo htmlspecialchars($fila['correo_inscripto']); ?></div>
<?php } ?>
<?php if ($fila['tecnico_aprobado']) { ?>
 Sí <input type="checkbox"/> No</div>
<?php } ?>
<?php if ($fila['profesor']) { ?>
<?php echo htmlspecialchars($fila['profesor']); ?></div>
<?php } ?>
<?php if ($fila['total_gastado']) { ?>
<?php echo number_format($fila['total_gastado'], 2); ?> <?php echo "€"; ?></div>
<?php } ?>
<?php if ($fila['clases_practicas']) { ?>
<?php echo $fila['clases_practicas']; ?></div>
<?php } ?>
<?php if ($fila['oportunidades_examen']) { ?>
<?php echo $fila['oportunidades_examen']; ?></div>
<?php } ?>
<!-- BOTONES PARA MODIFICAR CLASES PRÁCTICAS -->
<div>
<button class="bt-modificar" onclick="modificarClases(<?php echo $fila['id']; ?>, 1)">+</button>
<button class="bt-modificar" onclick="modificarClases(<?php echo $fila['id']; ?>, -1)">-</button>
<button class="bt-modificar" onclick="modificarClases(<?php echo $fila['id']; ?>, 0)">X</button>
<button class="bt-modificar" onclick="modificarClases(<?php echo $fila['id']; ?>, 0)">Quitar 1 Clase</button>
</div>
<a href="../../../../resultado_usuario/resultado.php?usuario_id=<?php echo $fila['id']; ?>">Ver Resultados de los test</a>



```
<?php
session_start();
include("C:/xampp/htdocs/Proyecto_integrado_2025/tecnologias/php/conexion/conexion.php");
// Verifica que el usuario tenga sesión activa como profesor
if (!isset($_SESSION['profesor_id'])) {
 die("Acceso denegado.");
}

// Obtener alumno asignado al profesor que ha iniciado sesión
$stmt = $conexion->prepare('
 SELECT clientes.id, clientes.nombre, clientes.apellidos, clientes.dni, clientes.telefono, clientes.correo,
 clientes.fecha_inscripto, clientes.fecha_nacimiento, clientes.profesor_id,
 clientes.aprobado, clientes.oportunidades_examenes, clientes.clases_practicas,
 IFNULL(SUM(compras.precio), 0) AS total_gastado
 FROM clientes
 INNER JOIN compras ON clientes.id = compras.usuario_id
 WHERE clientes.profesor_id = ?
 GROUP BY clientes.id
');

// Verificar si ID del profesor en la consulta
$stmt->bind_param("i", $_SESSION['profesor_id']);
$stmt->execute();
$Resultados = $stmt->get_result();

// Guardar los resultados en un array para evitar múltiples bucles
$alumnos = [];
while ($fila = $Resultados->fetch_assoc()) {
 $alumnos[] = $fila;
}
```



54


```

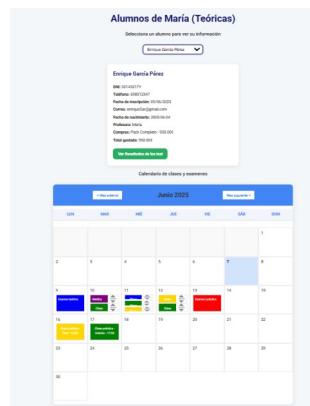
En este caso, el archivo **area_profesores.php** inicia la sesión del usuario y conecta con la base de datos verificando si el usuario ha iniciado sesión como profesor para permitirle el acceso. Luego consulta la base de datos para obtener los alumnos asignados al profesor con detalles como nombre, apellidos, DNI, teléfono, correo, fecha de inscripción, compras realizadas y total gastado, almacenando estos datos en un array.

En el HTML se muestra el nombre del profesor y un selector para elegir a un alumno, permitiendo que al seleccionarlo se despliegue su información en una tarjeta con todos sus datos y opciones para modificar el número de clases prácticas con botones de incremento y decremento.

Se usa JavaScript para controlar la interacción de la selección de alumnos y el manejo del cambio en la cantidad de clases prácticas mediante una solicitud **fetch()** al archivo **modificar_clases_profesor.php**. Este archivo recibe los datos en formato **JSON** desde **area_profesores.php**, valida que los datos sean correctos, y actualiza la base de datos asegurándose de que las clases prácticas nunca sean negativas utilizando una consulta segura con **prepare()** y **bind_param()**.

Finalmente, se envía una respuesta en formato **JSON** indicando si todo salió bien o si hubo algún error.

Con esto, ya tendríamos terminada la parte del profesor, y sólo nos quedaría la parte de la profesora. Prácticamente hace lo mismo, pero sin poder modificar las clases prácticas los alumnos.



The screenshot shows a user interface for managing theoretical students for teacher Maria. It includes a sidebar with teacher information (Maria Garcia Perez, DNI 12345678, etc.) and a main area with a calendar for June 2023. The calendar grid has various colored cells representing different activities or student status. To the right of the calendar is a block of PHP code used to query the database for student data.

```
Opção
session_start();
include("c:/xampp/htdocs/Proyecto_Integrado_2025/Tecnologias/php/conexion/conexion.php");
if (!isset($_SESSION["profesor_id"]) || !$_SESSION["profesor_id"] || 2) {
    die("Acceso denegado.");
}

// Obtiene alumnos asignados a María y sus compras desde la tabla 'compras'
$stmt = $conexion->prepare("
    SELECT clientes.id, clientes.nombre, clientes.apellidos, clientes.dni, clientes.telefono, clientes.correo,
    DATE(cliente.fecha_registro) AS fecha_inscripcion, clientes.fecha_nacimiento, clientes.profesor_id,
    GROUP_CONCAT(CONCAT(compras.producto, ' - ', compras.precio, '€') SEPARATOR ', ') AS productos_comprados,
    SUM(compras.precio) AS total_gastado
    FROM clientes
    LEFT JOIN compras ON clientes.id = compras.usuario_id
    WHERE clientes.profesor_id = 2
    GROUP BY clientes.id
");

$stmt->execute();
$resultados = $stmt->get_result();

// Guarda los resultados en un array para evitar múltiples bucles
$alumnos = [];
while ($fila = $resultados->fetch_assoc()) {
    $alumnos[] = $fila;
}
");

$stmt->close();
```

area_profesora.php funciona de manera similar a **area_profesores.php**, pero está diseñado específicamente para la profesora María, quien imparte clases teóricas en lugar de prácticas.

En la cabecera, se inicia la sesión y se verifica si el usuario es María comparando su ID con 2, bloqueando el acceso si no es ella. Luego, se consulta la base de datos para obtener los alumnos asignados a la profesora y se recopila información como nombre, apellidos, DNI, teléfono, correo, fecha de inscripción, fecha de nacimiento y compras realizadas.

A diferencia del área de profesores de prácticas, aquí no se incluyen datos relacionados con clases prácticas u oportunidades de examen, ya que los alumnos de María solo cursan teoría.

La información se guarda en un **array** y se usa en el **HTML** para mostrar un encabezado con el nombre de la profesora, un selector de alumnos y tarjetas con los detalles de cada uno. En el código JavaScript, se usa **document.addEventListener("DOMContentLoaded", function () {...})** para cargar la interacción cuando la página se inicia, y **addEventListener("change", function () {...})** para mostrar la tarjeta del alumno seleccionado. A diferencia del área de profesores de prácticas, no se implementan botones para modificar clases, ya que aquí solo se gestionan datos generales y compras de los alumnos. Esto marca una diferencia clave con **area_profesores.php**, donde los profesores pueden administrar clases prácticas y oportunidades de examen.

Ya explicada la diferencia entre profesores, se da por concluida la documentación del proyecto, **Autoescuela Almansa.es**.

Respecto a los demás archivos que están en el proyecto de los cuales no he hecho mención. Son archivos con mera información como pueden ser las clases prácticas, cursos intensivos, conoce a tus profesores...

En esta caso, en la presentación del proyecto se verán y diré cómo los he hecho, que son básicamente **HTML** y **CSS** todo.