



---

# PROYECTO FIN DE GRADO: THE BEAST BARBER

---

Manual de administración



JORGE ALFONSO ALCALDE  
2ºDAW

## Contenido

1.	Introducción .....	3
1.1	Tecnologías utilizadas .....	3
2.	Instalación de tecnologías necesarias .....	4
2.1	Instalación de XAMPP.....	4
2.2	Instalación de Visual Studio Code .....	13
3.	Configuración de la base de datos .....	18
4.	Despliegue del proyecto en XAMPP.....	20
4.1	Descarga del proyecto.....	21
4.2	Despliegue del proyecto.....	24
5.	Estructura del proyecto.....	27
5.1	Header y Footer.....	34
5.2	Sección “Inicio” .....	35
5.3	Sección “Servicios”.....	36
5.4	Sección “Contacto” .....	36
5.5	Sección “Iniciar sesión” .....	37
5.6	Sección “Mis citas” .....	39
5.7	Sección “Reservar” .....	42
6.	Aspectos importantes a tener en cuenta .....	47
6.1	Conexión a la base de datos .....	47
6.2	Directorio “img”.....	47
6.3	Directorio “documentación” .....	47
7.	Bibliografía .....	48

## 1. Introducción

En este manual se va a explicar todo lo necesario para poder desplegar el proyecto, entender su estructura, los directorios y archivos que lo forman, y el funcionamiento del proyecto en su conjunto.

La arquitectura de este proyecto está basada en el patrón de diseño modelo-vista-controlador, donde todo pasa por el archivo “index.php”, el cual es el “enrutador” del proyecto y el encargado de la navegación del proyecto, aunque todo esto se explicará más adelante. El proyecto en sí se basa en un sistema de gestión de citas en el que se ha desarrollado un CRUD de estas mismas, junto con una parte de marketing digital para promocionar la web y digitalizar el negocio para poder llegar a un mayor número de clientes.

El proyecto finalmente no está destinado a ser desplegado en internet, por lo que se va a realizar un tutorial de instalación y configuración de las tecnologías necesarias para el despliegue del proyecto en local, la configuración de la base de datos y los pasos para poder abrir el proyecto en el navegador.

### 1.1 Tecnologías utilizadas

Este proyecto ha sido realizado utilizando las siguientes tecnologías:

- HTML: para realizar la estructura de la interfaz de usuario.
- CSS: para dar estilo a la estructura de la interfaz de usuario.
- Javascript: para dar dinamismo y funcionalidad a la web, validaciones de formularios, etc.
- PHP: para la parte backend de la web, y para crear una interfaz de usuario dinámica.
- Mysql: es el gestor de base de datos relacionales utilizado.
- phpMyAdmin: la interfaz de usuario para conectarnos a Mysql.
- Visual Studio Code: es el IDE de desarrollo utilizado para el desarrollo del proyecto.
- Git-github: utilizado para el control de versiones del proyecto.
- Apache HTTP Server: para poder desplegar la web y poder ver el resultado del código, hacer pruebas, etc.

Este proyecto ha sido desarrollado sobre el sistema operativo Windows 11, por lo que el manual se va a basar en este sistema operativo, y se ha utilizado el navegador Google Chrome, pero puede utilizarse cualquier navegador acceder al proyecto.

## 2. Instalación de tecnologías necesarias

Para mayor simplicidad en vez de instalar PHP, Apache HTTP Server y Mysql por separado vamos a proceder a instalar XAMPP, que es un paquete de software gratuito que nos brinda un entorno de desarrollo completo e integra estas tecnologías.

En el caso de HTML, CSS y Javascript no es necesario instalar nada puesto que no se han utilizado librerías ni elementos externos de terceros, y no es necesaria ninguna instalación extra.

### 2.1 Instalación de XAMPP

Lo primero de todo es descargar XAMPP desde su página web oficial:

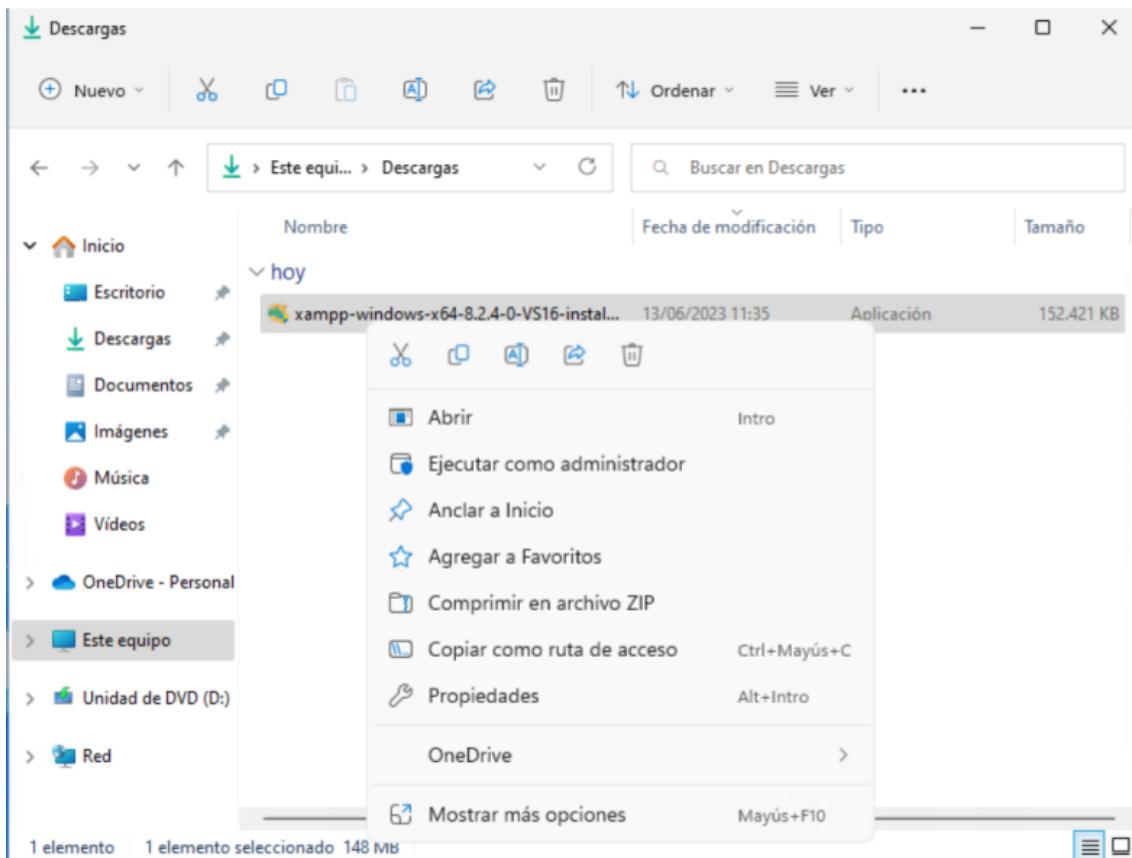
<https://www.apachefriends.org/es/download.html>

The screenshot shows the Apache Friends download page. The main heading is "Descargar". Below it, there's a section for "XAMPP para Windows 8.0.30, 8.1.25 & 8.2.12". This section lists three versions with their respective file sizes and download links:

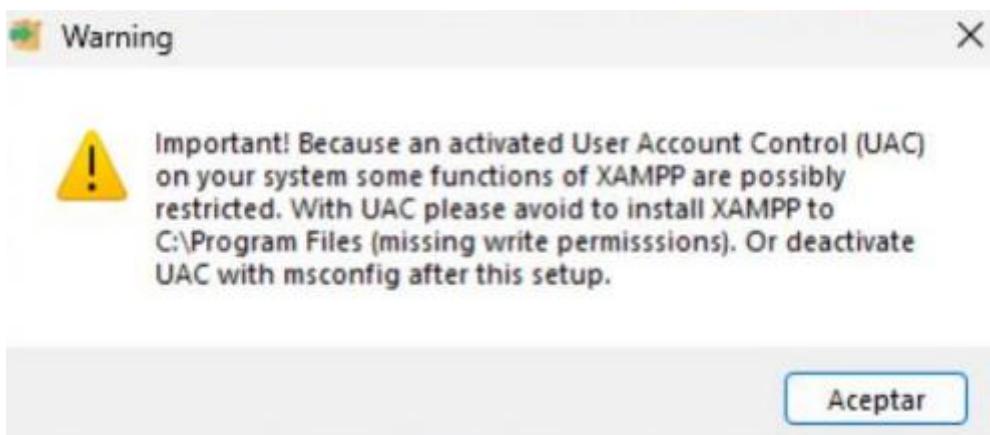
Versión	Suma de comprobación	Tamaño
8.0.30 / PHP 8.0.30	md5 sha1	144 MB
8.1.25 / PHP 8.1.25	md5 sha1	148 MB
8.2.12 / PHP 8.2.12	md5 sha1	149 MB

Each row has a "Descargar (14 MB)" button. To the right of the table is a "Documentación/FAQs" sidebar with links to Linux, Windows, and OS X FAQs. At the bottom of the table, it says "Windows XP or 2003 are not supported. You can download a compatible version of XAMPP for these platforms here."

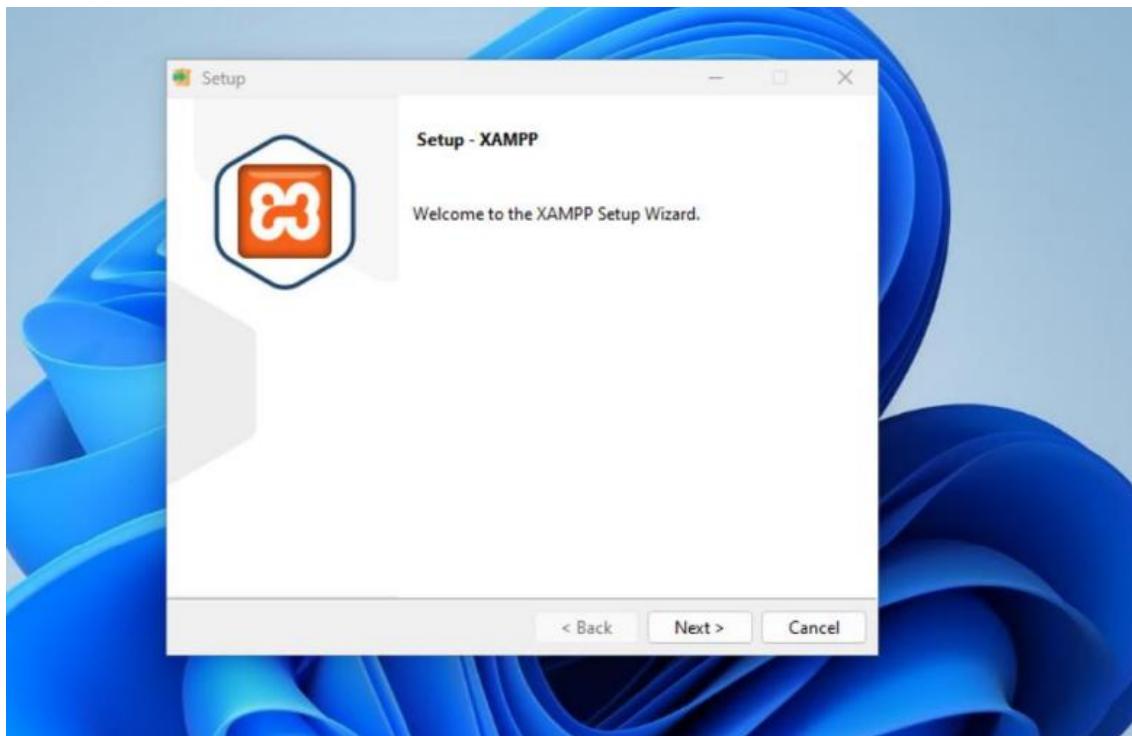
Una vez que se ha descargado ejecutamos como administrador:



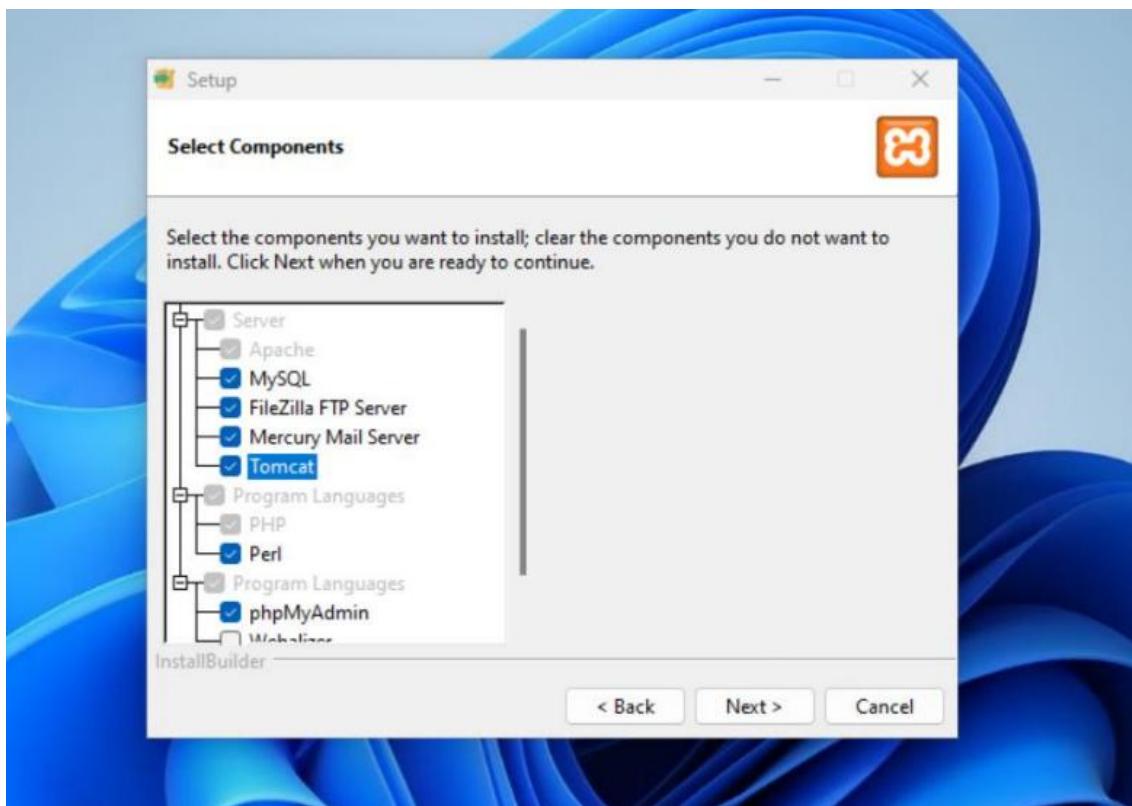
Nos aparecerá el siguiente aviso:



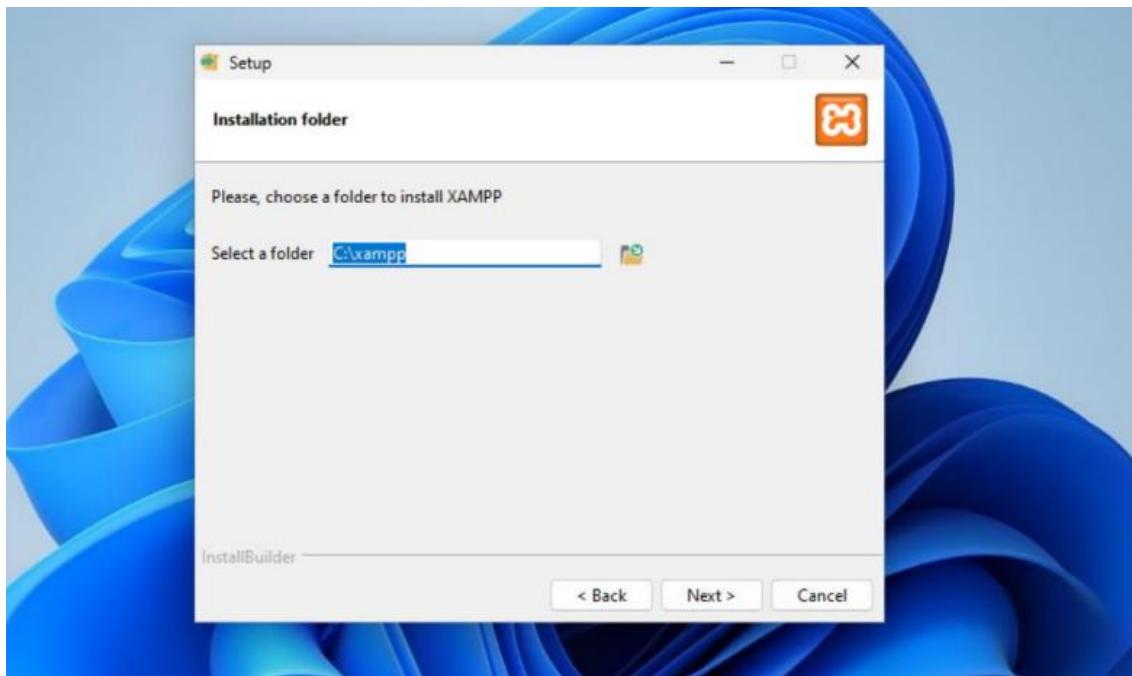
Aceptamos y ahora nos aparecerá la siguiente ventana y pulsaremos en siguiente:



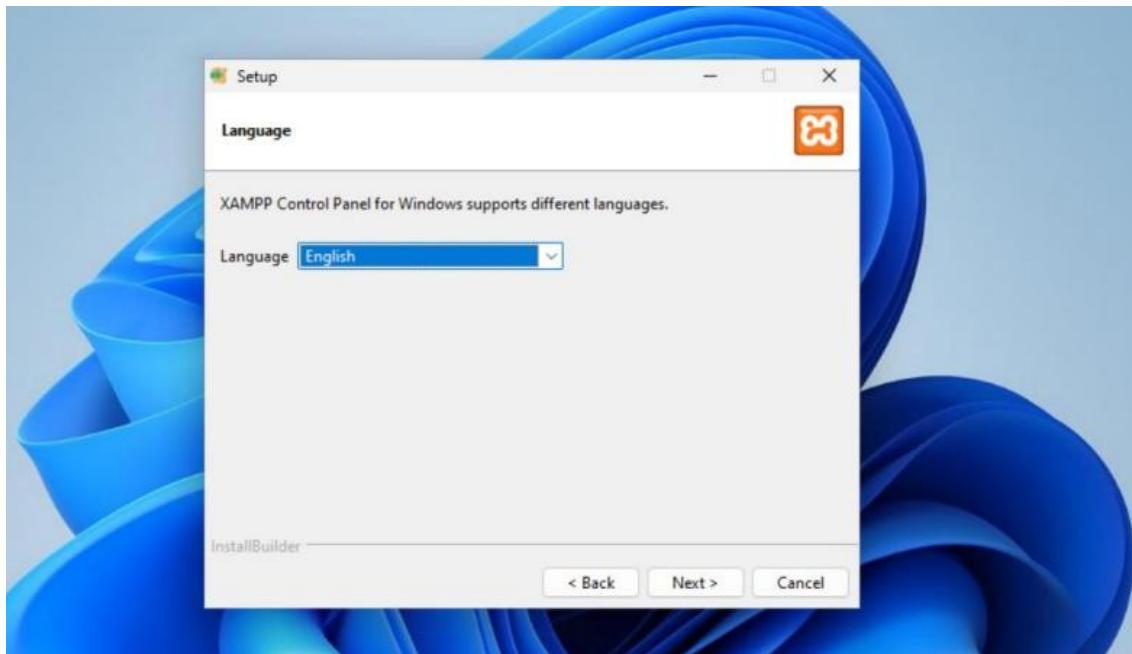
Pulsamos “Next” y ahora nos pedirá marcar los componentes que queremos que XAMPP instale. Principalmente los que se van a necesitar son Apache y MySQL además de PHP. Marcamos las que queramos y pulsamos siguiente (yo lo dejaré por defecto):



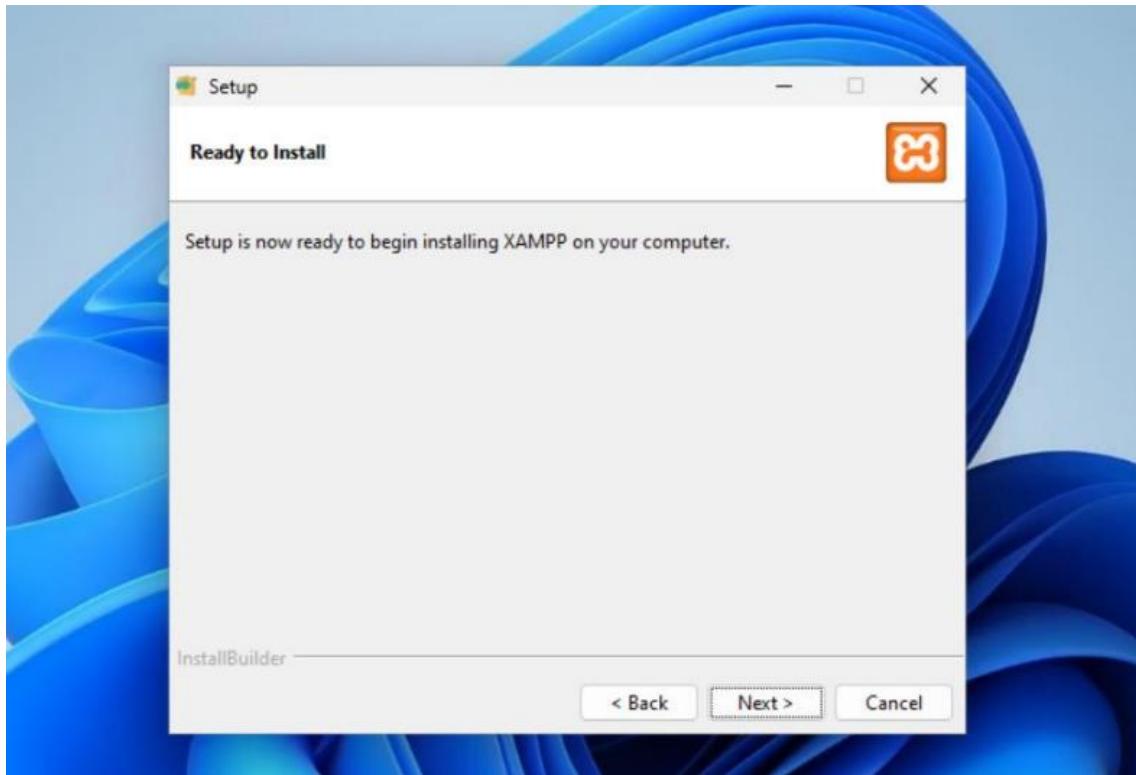
Pulsamos “Next” y la siguiente ventana nos permitirá cambiar el directorio raíz de XAMPP, aunque lo recomendable es dejarlo por defecto:



Pulsamos “Next”, la siguiente opción que da a elegir es el idioma del interfaz de XAMPP (sólo se puede elegir inglés o alemán):



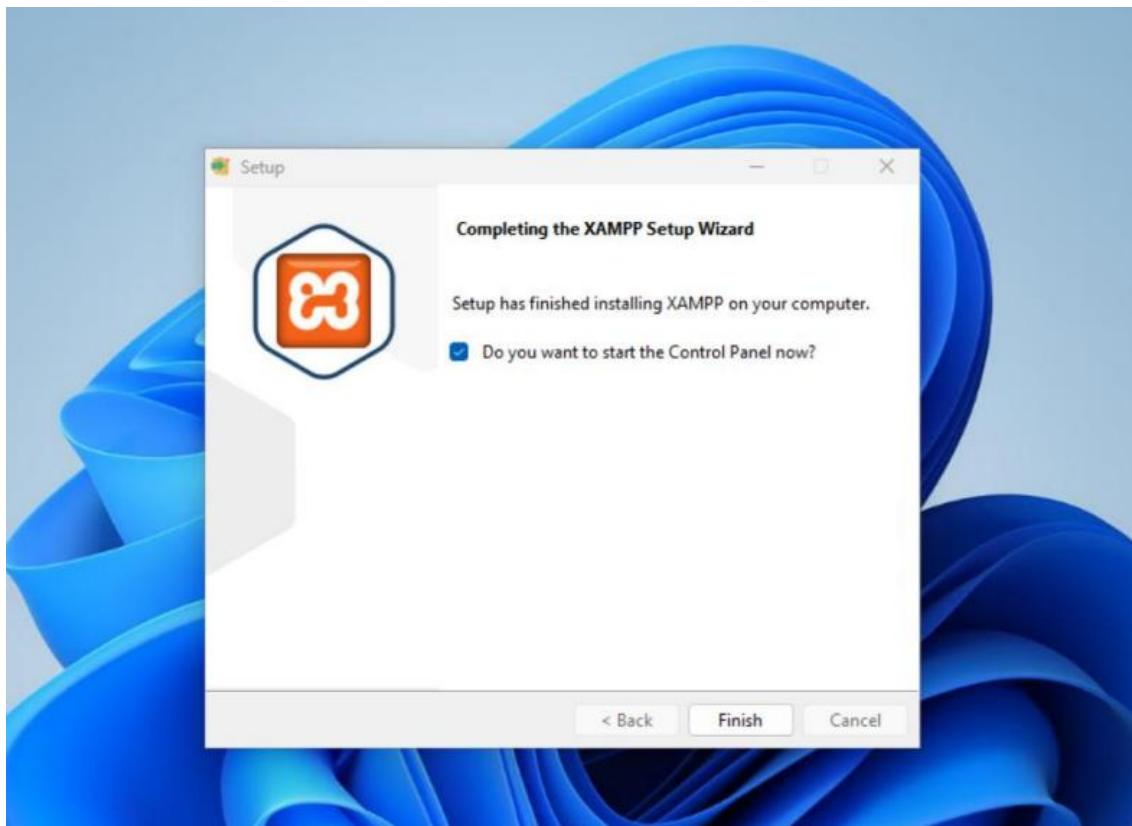
Elegimos uno y pulsamos “Next”. Se nos mostrará un último apartado previo a la instalación, que nos indicará que XAMPP está listo para instalarse:



Pulsamos “Next” y comenzará la instalación:

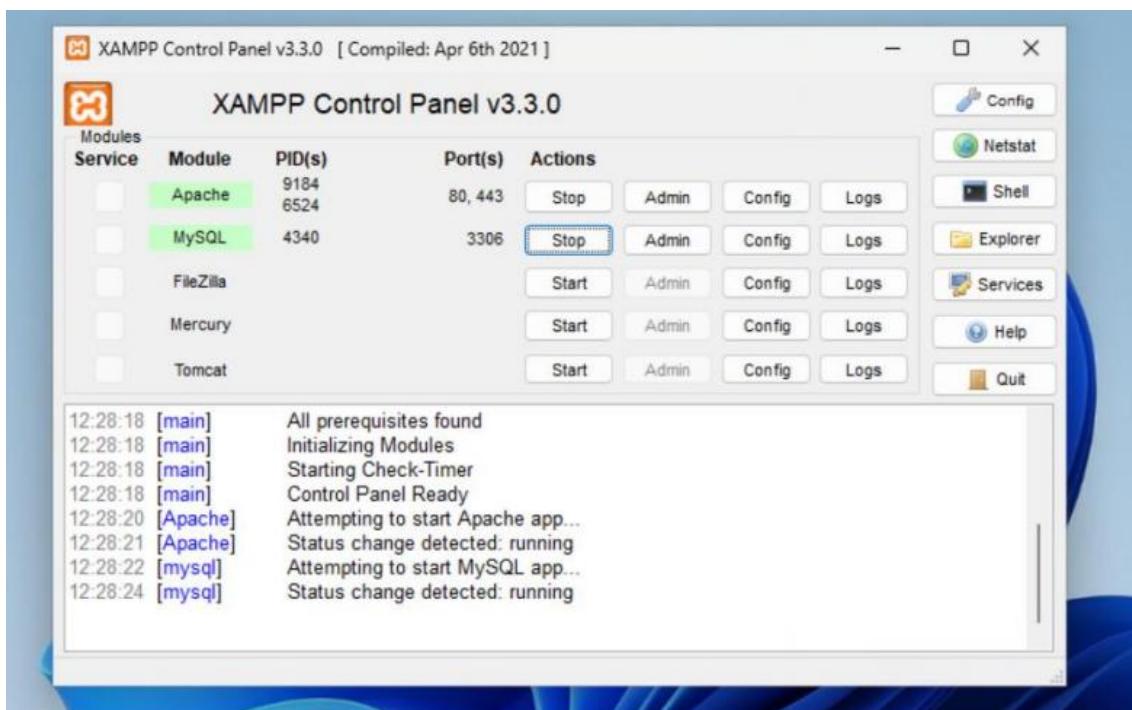


Una vez finalizada se nos mostrará la siguiente ventana:



Si queremos comprobar que la instalación ha sido correcta dejamos la opción marcada y pulsamos “Finish”. Esto nos abrirá el panel de control de XAMPP.

Una vez en el panel iniciamos los servicios de Apache y MySQL:

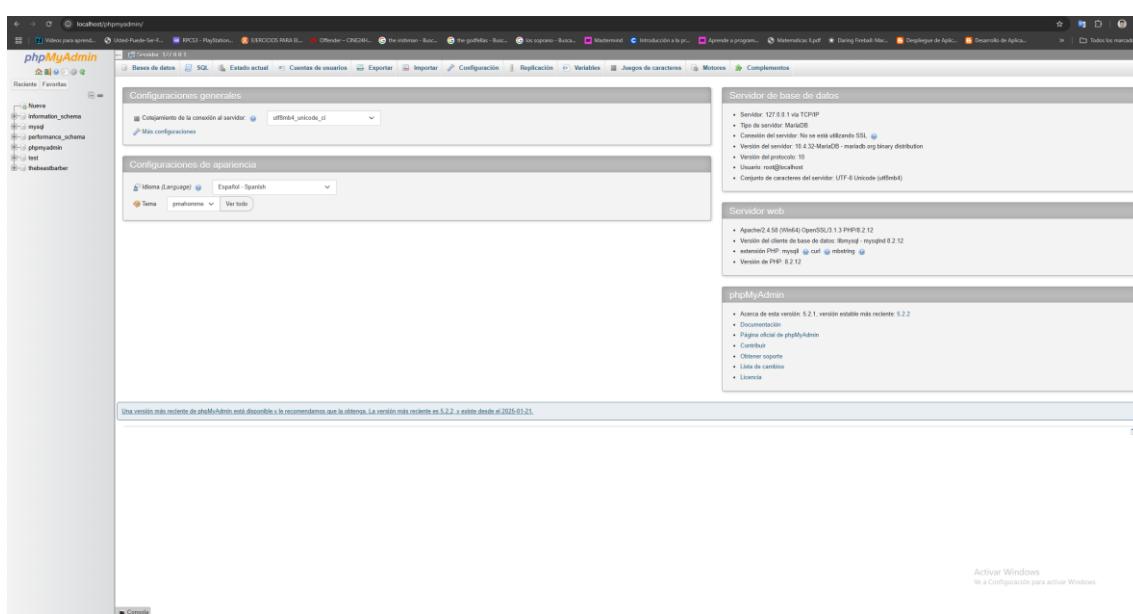




Una vez realizado esto abrimos un navegador y buscamos “localhost”, lo cual nos debe mostrar la página de bienvenida de XAMPP:



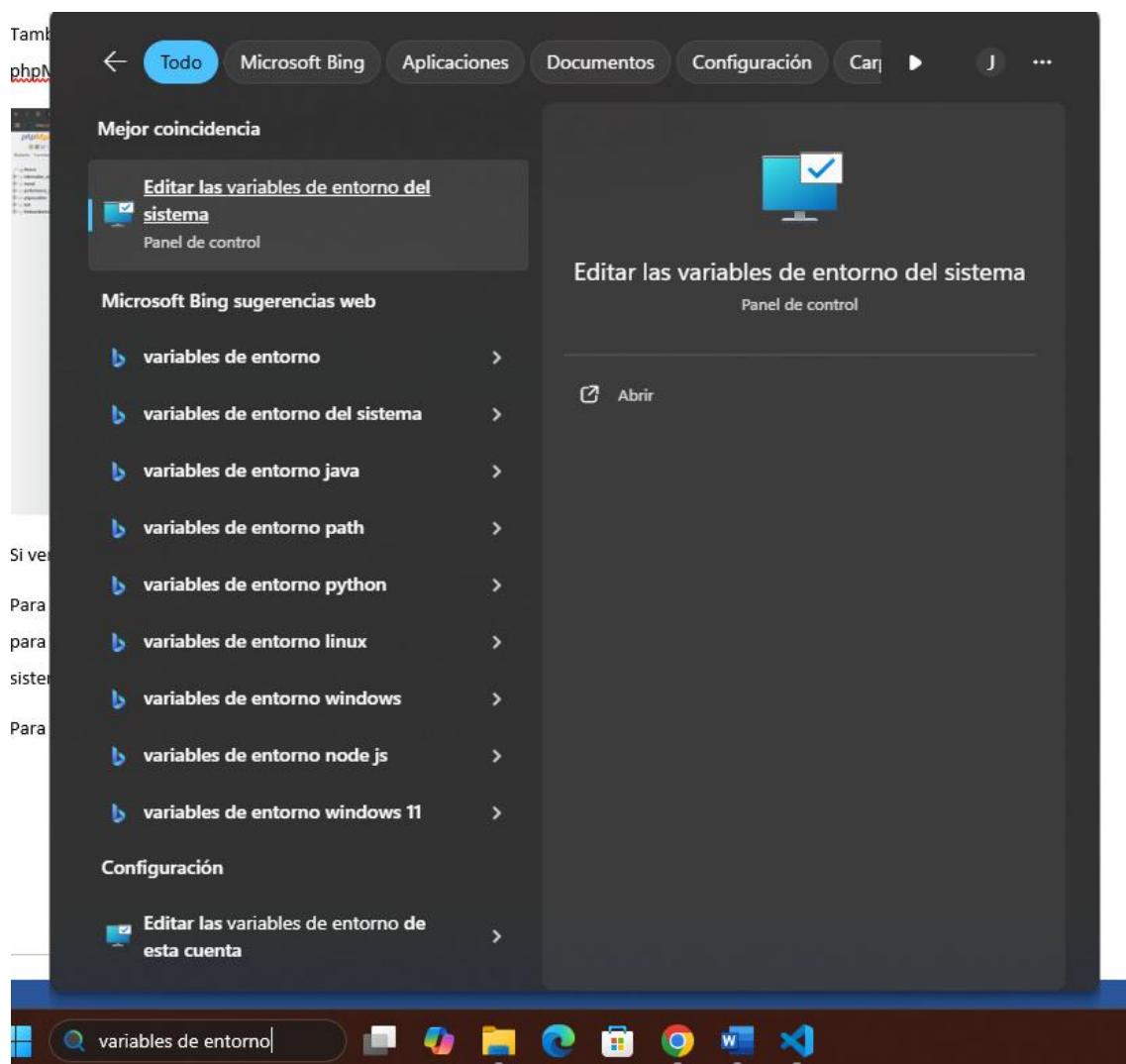
También podremos acceder ya a nuestro gestor de base de datos Mysql a través de phpMyAdmin. Para ello vamos a buscar en el navegador “localhost/phpmyadmin”:



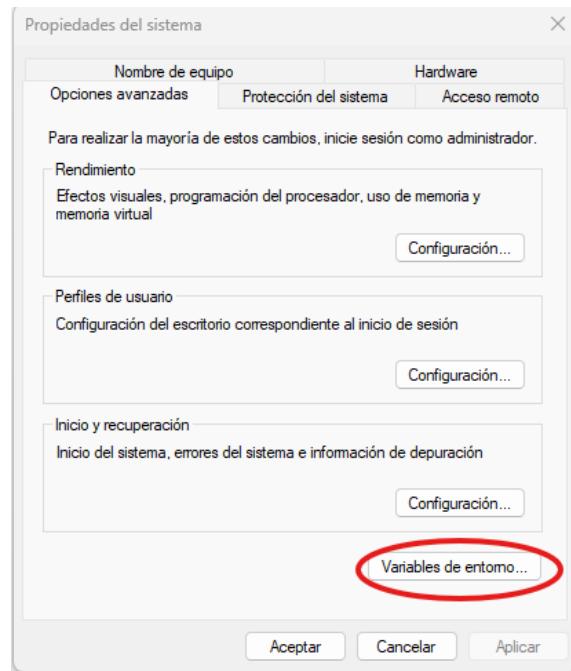
Si vemos algo como esto significa que todo ha ido bien.

Para finalizar la configuración de XAMPP debemos configurar la variable de entorno de PHP para poder ejecutar comandos PHP desde cualquier ventana o terminal, y para que nuestro sistema reconozca que PHP está instalado y funcione correctamente.

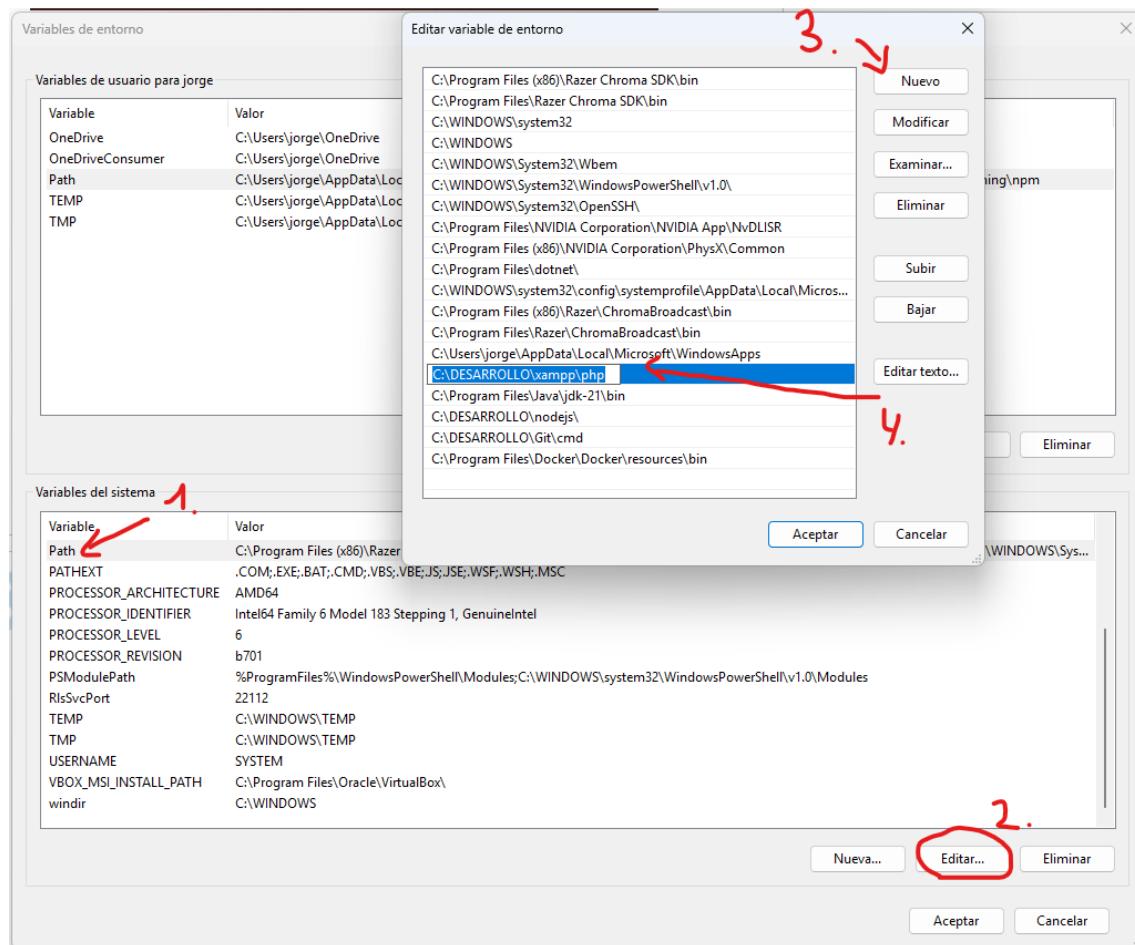
Para ello vamos a abrir el buscador de Windows y escribimos “Variables de entorno”:



Abrimos la primera opción que nos sale, y una vez dentro pulsamos “Variables de entorno...”:



Ahora debemos buscar “Path” en variables del sistema, hacer clic en “Editar”, pulsar “Nuevo” y añadir la ruta al directorio “php” dentro de la instalación de XAMPP:



Tras esto habremos finalizado la instalación de XAMPP y su configuración para nuestro Proyecto.

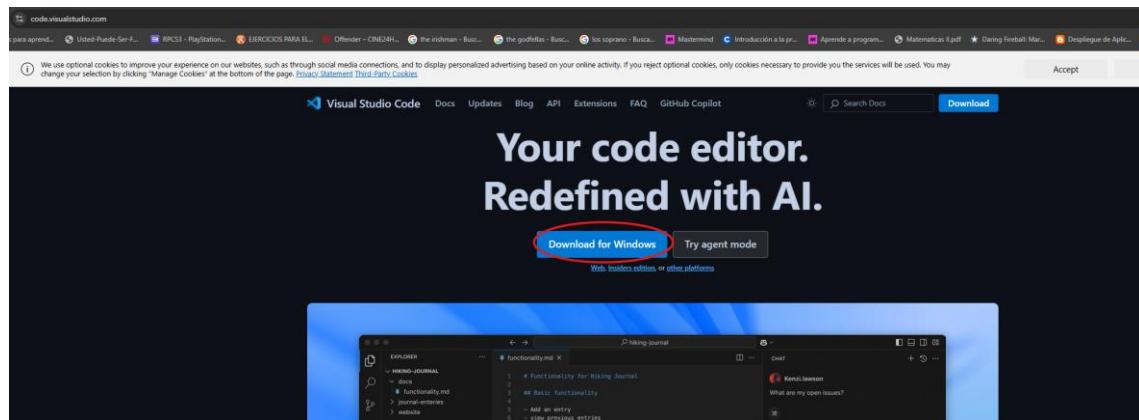
Es importante destacar que no es necesario configurar las variables de entorno de PHP y Mysql para desplegar el proyecto, de hecho en este tutorial ni siquiera se va a configurar la de Mysql.

## 2.2 Instalación de Visual Studio Code

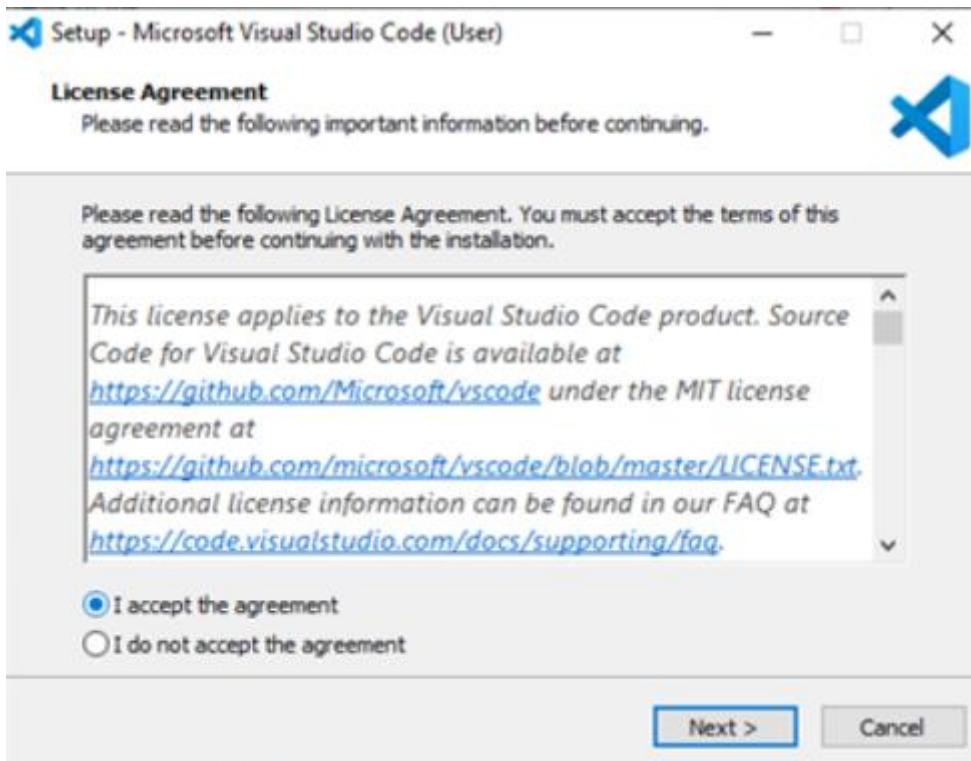
Respecto al IDE de desarrollo es importante aclarar que se puede utilizar cualquier entorno de desarrollo, pero se va a explicar la instalación del IDE utilizado por mí.

Lo primero es entrar a la web oficial de VSCode: <https://code.visualstudio.com/>

Una vez aquí pulsamos “Descargar para Windows”:

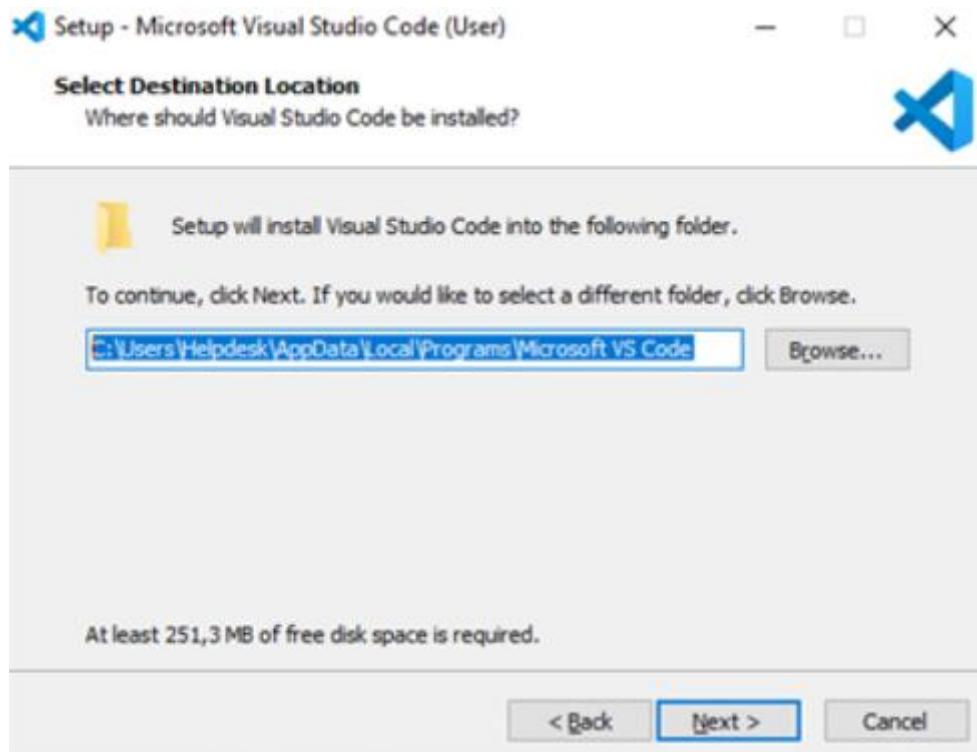


Esto nos descargará el instalador de VSCode, el cual debemos ejecutar. Lo primero que nos saldrá es la licencia de usuario, la cual debemos aceptar y pulsar en “Next”:

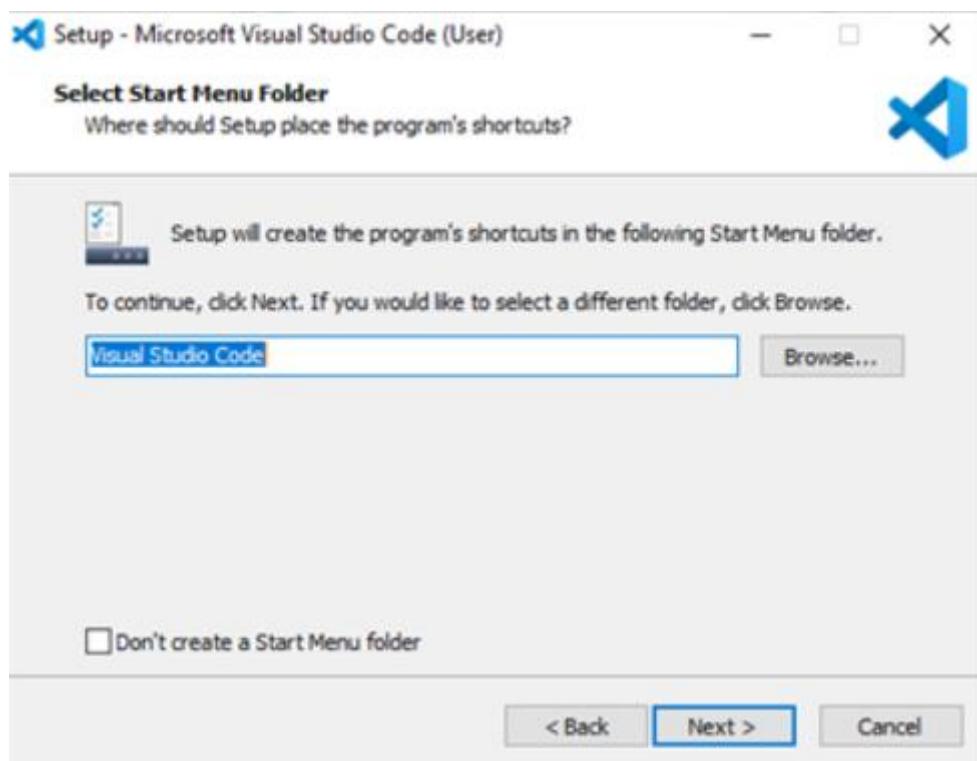


Lo siguiente será definir la ruta de instalación, recomiendo dejarla por defecto si no se necesita

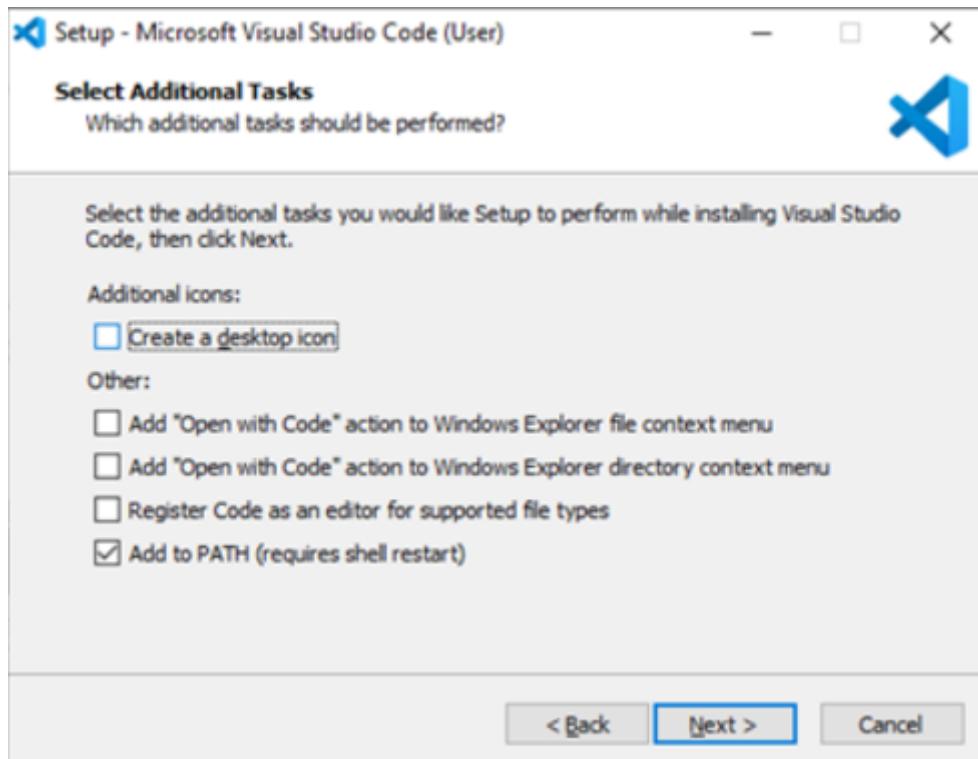
una ubicación concreta:



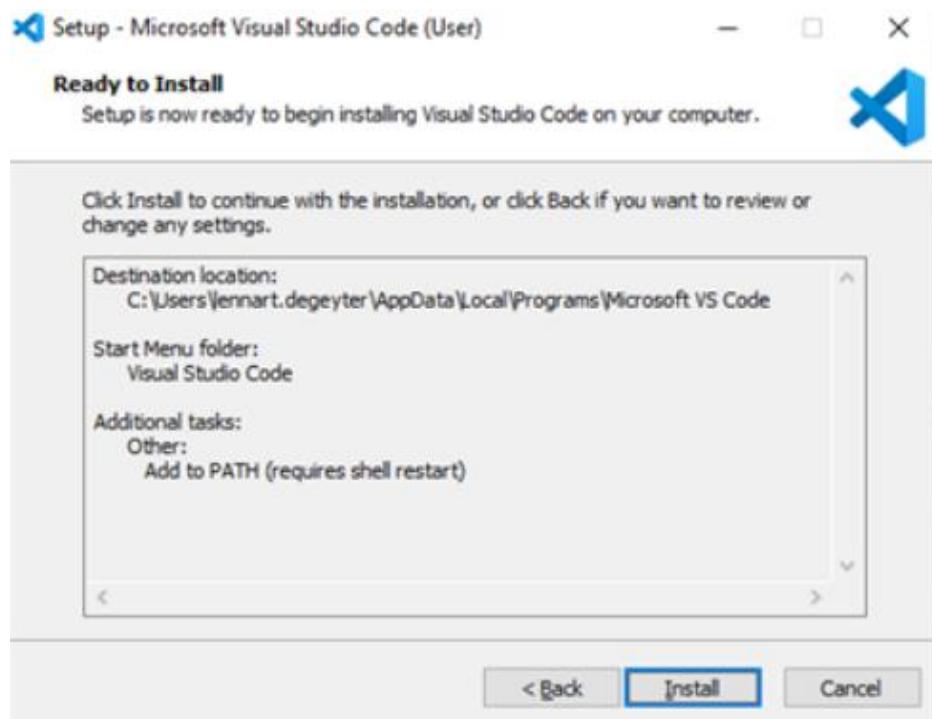
Ahora podremos poner un nombre a la carpeta del menú de inicio y seleccionar si no queremos crear una carpeta en el menú de inicio (dejar por defecto):



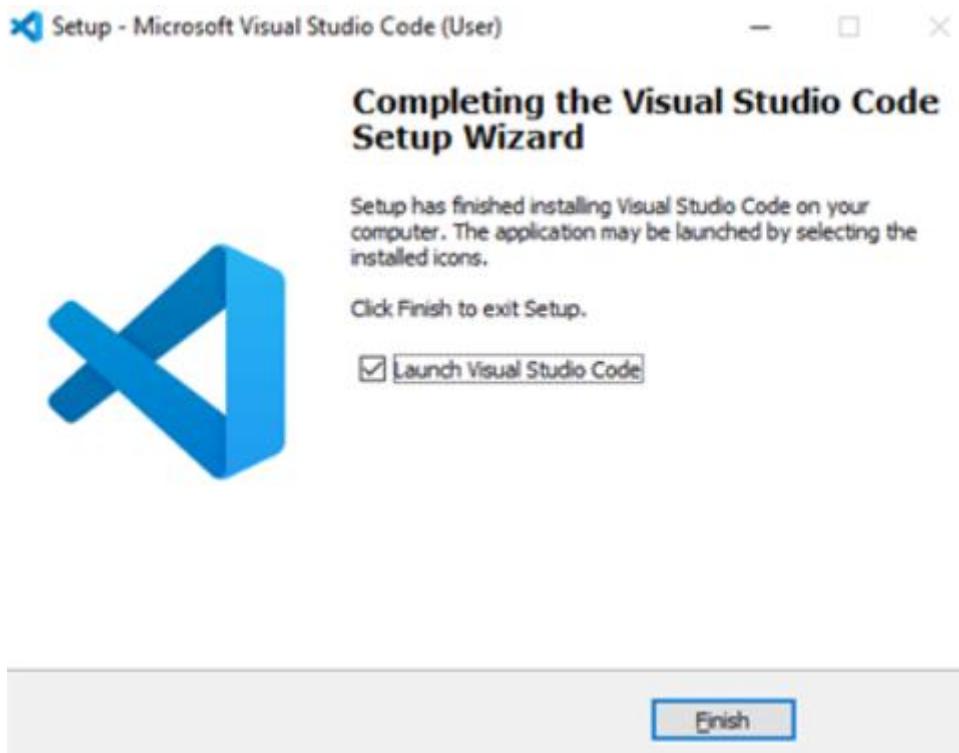
Tras esto podremos elegir si crear un acceso directo o no, y podremos añadir la ruta directamente al Path marcando la última opción:



Finalmente veremos un resumen de los pasos anteriores, pulsamos “Instalar” y comenzará la instalación:



Una vez finalizada nos saldrá una ventana indicando el éxito en la instalación, dejamos la opción marcada y pulsamos “Finish”:



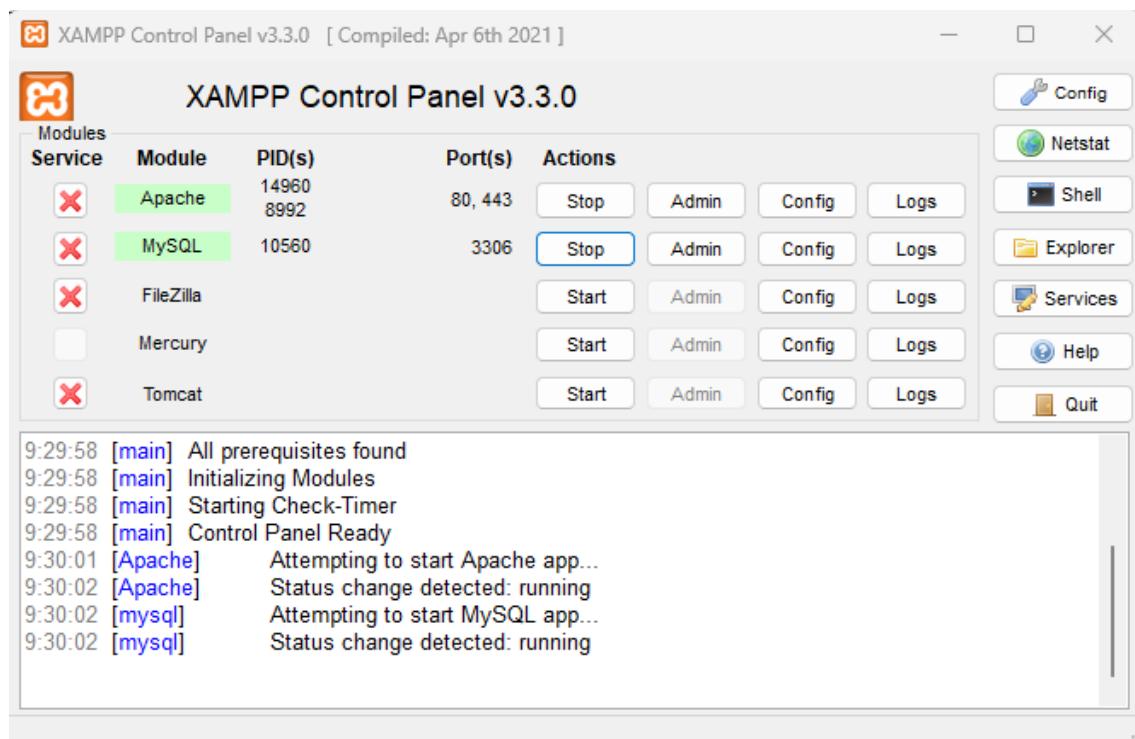
Si todo está bien se nos abrirá una ventana de Visual Studio Code, y la instalación habrá sido exitosa y finalizada.

### 3. Configuración de la base de datos

Dentro del proyecto encontraremos el directorio “BBDD” en la raíz del proyecto, donde está el archivo “database.sql”, el cual vamos a importar en phpmyadmin para tener lista la base de datos. Este script nos va a:

- Crear la base de datos “thebeastbarber”.
- Crear el usuario “admin” y otorgar los permisos suficientes para realizar el CRUD de la base de datos.
- Crear la tabla “usuarios” para poder guardar usuarios.
- Crear la tabla “citas\_reservadas” para poder gestionar las citas de los usuarios.

Lo primero que vamos a hacer es abrir el panel de control de XAMPP e iniciar Apache y Mysql:



Una vez realizado esto, vamos a ir a un navegador y buscamos “localhost/phpmyadmin”:



JORGE ALFONSO ALCALDE  
PROYECTO FIN DE GRADO: THE BEAST BARBER

The screenshot shows the phpMyAdmin interface for a MySQL database named 'thebeastbarber'. The 'Importar' (Import) tab is highlighted with a red circle. The left sidebar lists databases: Nueva, Information\_schema, mysql, performance\_schema, phpmyadmin, and test. The main area has sections for 'Configuraciones generales' (General configurations) and 'Configuraciones de apariencia' (Appearance configurations). On the right, there are panels for 'Servidor de base de datos' (Database server), 'Servidor web' (Web server), and 'phpMyAdmin' (phpMyAdmin version information).

Una vez aquí pulsamos “Importar” (señalado en la imagen anterior) y pulsamos en “Seleccionar archivo”:

Importando al servidor actual

The 'Importar' (Import) dialog box contains several sections: 'Archivo a importar:' (File to import), 'Importación parcial:' (Partial import), 'Otras opciones' (Other options), 'Formato' (Format), and 'Opciones específicas al formato:' (Format-specific options). The 'Archivo a importar:' section has a 'Seleccionar archivo' (Select file) button, which is circled in red. The 'Importación parcial:' section has a checked checkbox for 'Permitir la interrupción de una importación en caso que el script detecte que se ha acercado al límite de tiempo PHP' (Allow interruption of an import if the script detects it's approaching the PHP time limit). The 'Formato' section shows 'SQL' selected. The 'Opciones específicas al formato:' section includes 'Modalidad SQL compatible:' set to 'NONE' and a checked checkbox for 'No utilizar AUTO\_INCREMENT con el valor 0' (Do not use AUTO\_INCREMENT with value 0). At the bottom is a large 'Importar' (Import) button.

Una vez seleccionado nuestro script “database.sql” pulsamos “Importar”, si todo ha ido bien debemos ver algo así:



JORGE ALFONSO ALCALDE  
PROYECTO FIN DE GRADO: THE BEAST BARBER

The screenshot shows the MySQL Workbench interface with the following details:

- Database:** thebeastbarber
- Tables:**
  - usuarios:** Contains columns: Id (INT NOT NULL AUTO\_INCREMENT), nombre (VARCHAR(20) NOT NULL), apellido (VARCHAR(30) NOT NULL), telefono (BIGINT), correo (VARCHAR(50) NOT NULL), contraseña (VARCHAR(255) NOT NULL), and PRIMARY KEY (Id). It also has a UNIQUE KEY unique\_correo (correo) and a DEFAULT CHARACTERSET utf8mb4.
  - citas\_reservadas:** Contains columns: id (INT NOT NULL AUTO\_INCREMENT), usuario\_id (INT NOT NULL), servicio (VARCHAR(50) NOT NULL), fecha (DATE NOT NULL), hora (TIME NOT NULL), and PRIMARY KEY (id). It has a UNIQUE KEY uq\_reserva (fecha, hora) and a FOREIGN KEY (usuario\_id) REFERENCES usuarios(id).

Logs at the bottom show various MySQL errors related to database creation and user creation.

Se ha creado la base de datos, el usuario local con permisos suficientes para manipular la base de datos, la tabla “usuarios” para registrar a nuestros usuarios y la tabla “citas\_reservadas” para la gestión de citas reservadas. Las citas disponibles se generan mediante javascript, por lo que no es necesario tener una tabla con filas por cada cita disponible, lo que ha reducido enormemente la carga en base de datos.

El script como tal no inserta datos en las tablas, por lo que usted debe registrar un nuevo usuario a través de la web y generar citas con ese usuario para poder probar la web de manera completa.

#### 4. Despliegue del proyecto en XAMPP

Si has seguido esta guía en orden y has completado exitosamente los pasos anteriores tendrás ya instalado XAMPP (que incluye PHP y Mysql), VSCode y configurado phpMyAdmin con la base de datos ya creada y todo preparado, por lo que vamos a poner el proyecto en el directorio “htdocs” de XAMPP y explicar como visualizarlo en el navegador.

#### 4.1 Descarga del proyecto

Lo primero de todo es tener el proyecto, que lo puedes obtener de dos maneras:

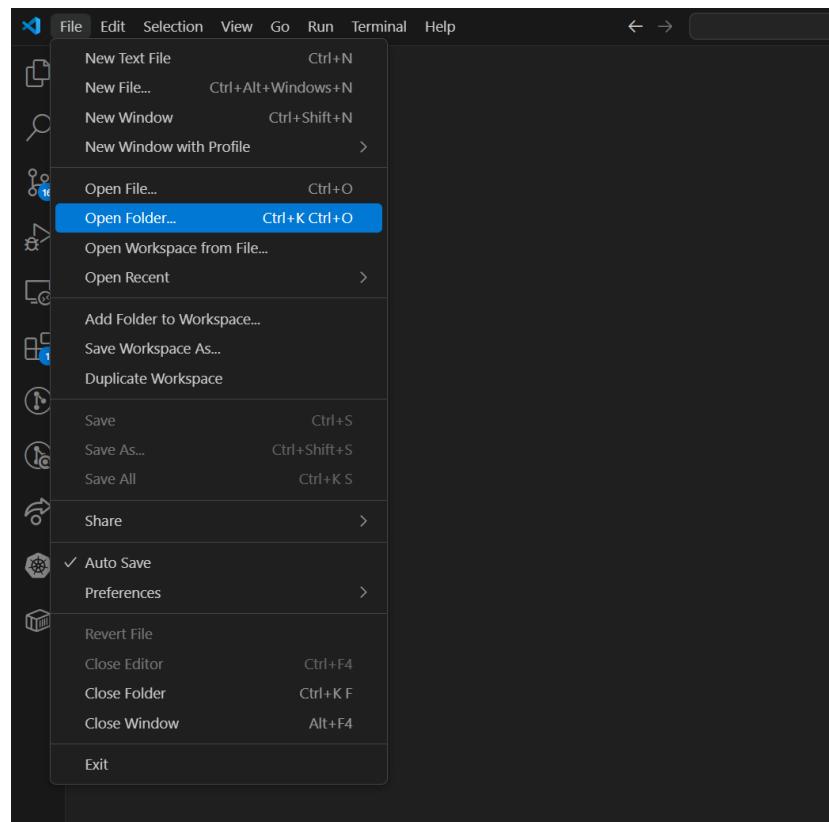
- Adquiriendo una copia el día de la presentación de este proyecto.
- Clonando el proyecto del repositorio de github: <https://github.com/jalfalc/TFG-TheBeast>.

En este caso no se va a explicar cómo instalar y configurar paso a paso git y github por motivo de tiempo, pero te dejo una guía por aquí:

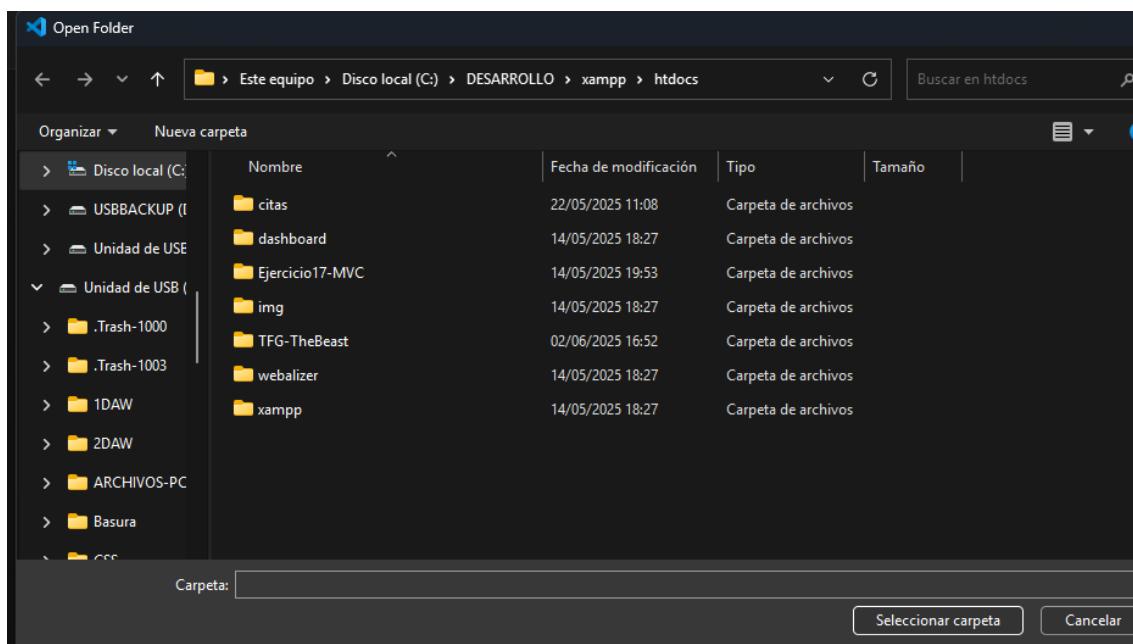
<https://www.youtube.com/watch?v=wHh3IgJvXcE>

Una vez hayas instalado y configurado git podremos descargar el proyecto del directorio de github, vamos a explicar como hacerlo mediante VSCode, pero se puede hacer directamente con un terminal (cmd, powershell, git bash, etc).

1. Abrimos VSCode y pulsamos “File” > “Open folder...”:



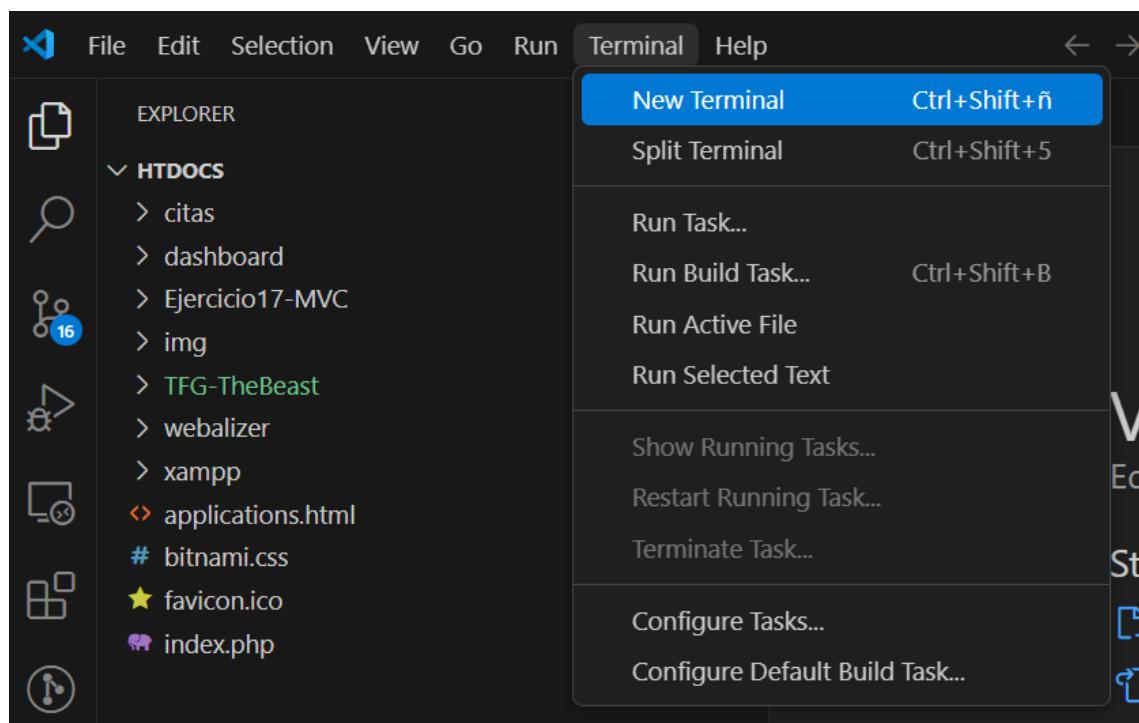
2. Buscamos la ubicación donde está instalado XAMPP, y accedemos a “htdocs”:



En mi caso está en "C:\DESARROLLO\xampp\htdocs", pero no tienes por qué tenerlo en la misma ruta. Dentro de "htdocs" tendremos los archivos por defecto de bienvenida de apache.

Los podemos eliminar, pero en este caso se va a explicar cómo hacerlo sin eliminarlos.

Una vez situado dentro de "htdocs" en VSCode pulsamos "Seleccionar carpeta" y nos situará dentro del directorio "htdocs". Una vez aquí pulsamos "ctrl + shift+ ñ" para abrir un terminal, o si no vamos a "terminal > new terminal":

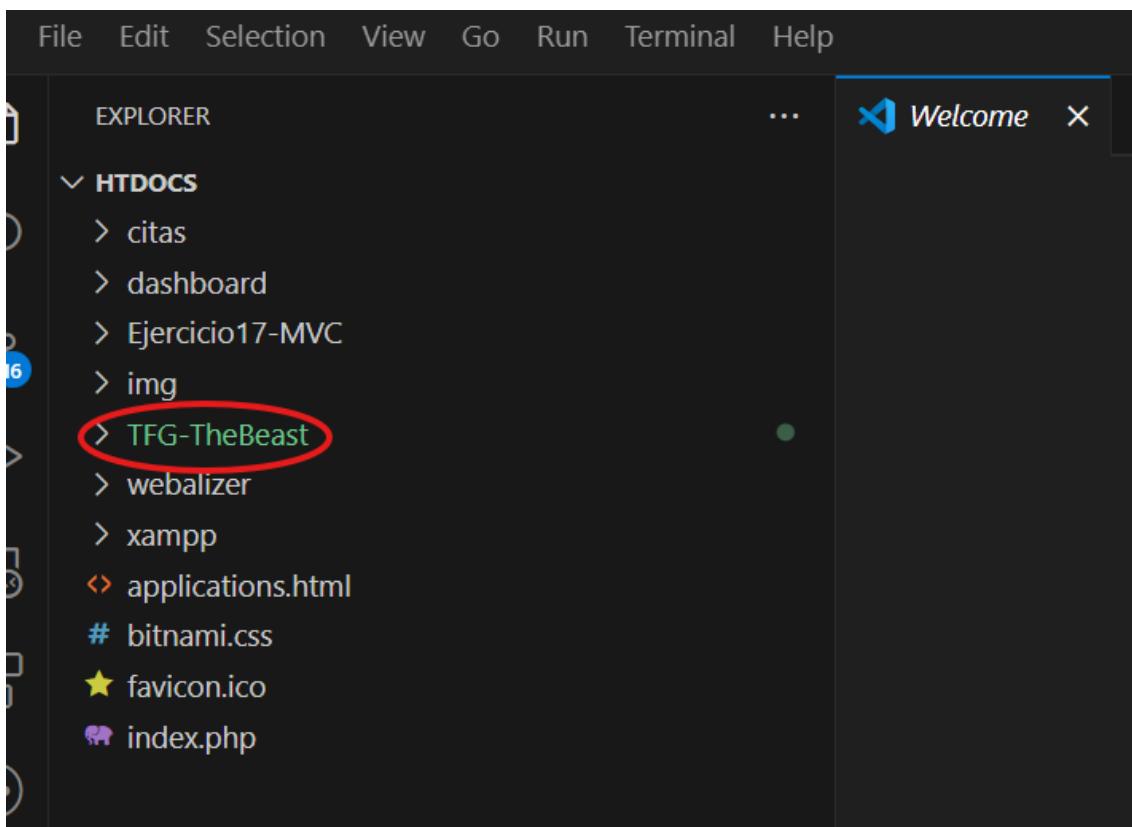


Ahora, en el terminal que se ha abierto ejecutamos "git clone

<https://github.com/jalfalc/TFG-TheBeast>:

The screenshot shows the VSCode terminal tab active. The command 'git clone https://github.com/jalfalc/TFG-TheBeast' is being typed into the terminal. The output shows the command being run, with the URL highlighted in blue. The terminal tab is labeled 'TERMINAL' and has other tabs like 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'PORTS', and 'GITLENS' visible at the top.

Esto nos descargará el proyecto dentro de "htdocs", lo cual se puede comprobar así:



En mi caso no lo he ejecutado porque ya lo tenía, pero te tiene que aparecer dentro de “htdocs”.

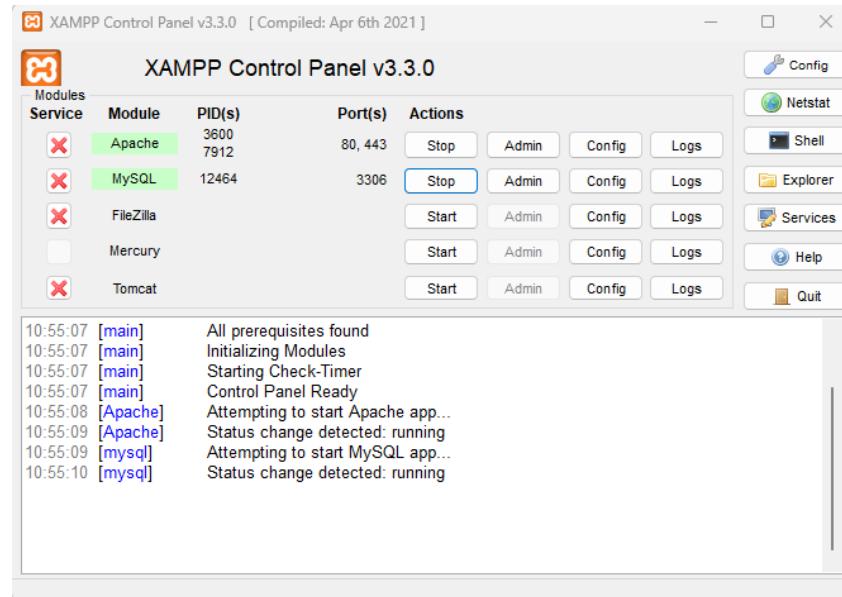
**¡¡¡ES IMPORTANTE RECALCAR QUE TODOS ESTOS PASOS SOLO SON NECESARIOS PARA DESCARGAR EL PROYECTO DE GITHUB!!!!**

Si usted ya tiene el proyecto no necesita realizar esta serie de pasos para descargarlo.

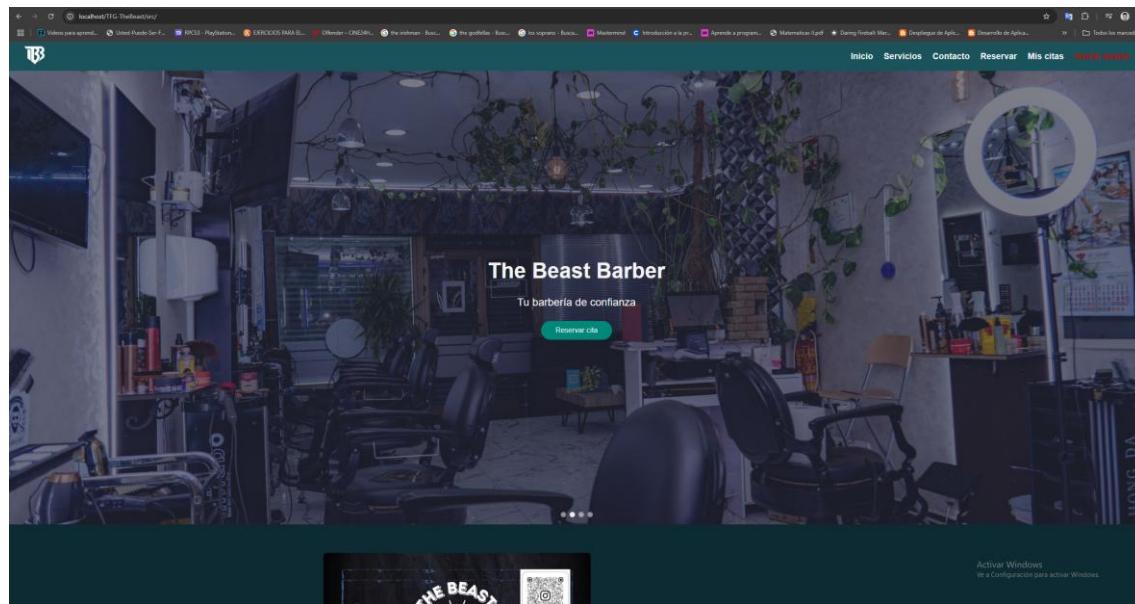
En caso de que lo tenga sólo necesita situar el proyecto dentro del directorio “htdocs” de XAMPP, y estará en la misma situación que si hubiese seguido los pasos descritos para descargarlo con VSCode.

#### 4.2 Despliegue del proyecto

Antes de abrir el navegador es muy importante tener iniciado Mysql y Apache en el panel de control de XAMPP:



Ahora, con la base de datos lista y configurada, y el proyecto dentro del directorio “htdocs” de XAMPP podemos ir al navegador y buscar “<http://localhost/TFG-TheBeast/src/>” . Si todo ha ido bien deberíamos ver la página de inicio:



En caso contrario debe revisar la ruta del proyecto, puesto que la url se descompone así:

- localhost/: hace referencia a “htdocs” de XAMPP (que es el directorio de apache).

- TFG-TheBeast/: es el nombre del directorio del proyecto.
- src/ : es necesario porque dentro está el archivo “index.php”, que es el que se ejecuta por defecto (por eso no es necesario poner “http://localhost/TFG-TheBeast/src/index.php”).

Si no te funciona comprueba tu ruta y tu ubicación del proyecto y ajústala correctamente.

Si has logrado llegar hasta aquí enhorabuena, está todo listo para comprobar el funcionamiento de la web. En este momento ya puede ir a “Iniciar sesión”:



Y aquí hacer clic en “Crear cuenta”:

Iniciar Sesión

Correo electrónico:

Contraseña:

Enviar

¿No tienes una cuenta? [Crear cuenta](#)

Y ya podrá crearse una cuenta y probar toda la funcionalidad del proyecto.

## 5. Estructura del proyecto

Vamos a explicar la estructura del proyecto: cómo está formado, que partes lo componen, que archivos componen cada sección y cómo funciona el proyecto como tal.

Como ya se ha comentado en la introducción, el proyecto está estructurado siguiendo el patrón modelo-vista-controlador, donde el modelo tiene la lógica de negocio, el controlador solicita datos al modelo y les da forma, y la vista solicita dichos datos ya tratados al controlador, y los muestra al usuario.

La estructura del proyecto se ve así:

```
TFG-TheBeast/  
|  
└── .env  
└── README.md  
|  
└── BBDD/  
    └── database.sql
```

```
|  
|   └── folder documentacion/  
|       |   └── folder diagramas/  
|       |       └── folder diagrama-gantt/  
|       |           └── file DiagramaGantt.md  
|       |           └── file DiagramaGantt.png  
|       |           └── file diagrama_gantt.png  
|       |           └── file diagrama_gantt.xlsx  
|  
|  
|   └── folder manuales/  
|       └── file ManualAdministracion_TBB.docx  
|       └── file ManualUsuario_TBB.docx  
|  
|  
└── folder img/  
    |   └── folder login/  
    |       └── file login-background.webp  
|  
|  
|   └── folder servicios/  
|       └── file barba.png  
|       └── file cejas.jpg  
|       └── file corte&barba.png  
|       └── file jubilado.png
```

| | └── 📄 limpiezafacial.avif  
| | └── 📄 opinion1.jpg  
| | └── 📄 opinion2.webp  
| | └── 📄 opinion3.jpg  
| | └── 📄 opinion4.jpg  
| | └── 📄 opinion5.png  
| | └── 📄 opinion6.jpg  
| | └── 📄 servicio1.png

| |

| └── 📂 principal/  
| └── 📄 carousel1.jpg  
| └── 📄 carousel2.webp  
| └── 📄 carousel3.jpg  
| └── 📄 carousel4.jpg  
| └── 📄 image.png  
| └── 📄 mapa.png  
| └── 📄 Servicios.jpg  
| └── 📄 whatsapp.png  
| └── 📄 logo.png

|



```
└── src/
    ├── Controller/
    │   ├── Login_Controller.php
    │   ├── Logout_Controller.php
    │   ├── MisReservas_Controller.php
    │   ├── Principal_Controller.php
    │   ├── Registro_Controller.php
    │   ├── Reservas_Controller.php
    │   ├── Servicios_Controller.php
    │   └── Verify_Controller.php
    ├── Model/
    │   └── DAO/
    │       ├── Conexion.php
    │       ├── MySqlDAO.php
    │       └── Usuario.php
    └── View/
        ├── Footer/
        │   └── Footer_View.php
        ├── Header/
        │   └── Header_View.php
        └── EditarReserva_View.php
```



```
|   └── Login_View.php
|   └── MisReservas_View.php
|   └── Principal_View.php
|   └── Registro_View.php
|   └── Reservas_View.php
└── Servicios_View.php
```

```
└── css/
    ├── colors.css
    ├── footer/
        └── footer.css
    ├── header/
        └── header.css
    └── login/
        └── login.css
    └── principal/
        └── principal.css
    └── reservas/
        └── mis_reservas.css
```



```
|   |   |   └── reservas.css
|   |   └── servicios/
|   |       └── servicios.css
|
|   └── js/
|       └── header.js
|
|   └── registro/
|       └── validarRegistro.js
|
|   └── login/
|       └── validarLogin.js
|
|   └── principal/
|       └── principal.js
|
|   └── reservas/
|       └── cargarHoras.js
|
|   └── editarReserva.js
|
|   └── eliminarReserva.js
|
|   └── servicios/
|       └── servicios.js
|
└── index.php
```

Como se puede apreciar se trata de un proyecto de un tamaño ya considerable, por lo que vamos a desglosar cada sección de la web explicando qué archivos la componen, cual es la finalidad de cada uno y cómo funcionan todos los archivos conjuntamente. Por problemas de tiempo no se va a explicar con detalle cada archivo, pero puedes ver comentarios en todo el código que explican la funcionalidad del código al que acompañan.

Para comenzar es imprescindible explicar que función tiene “index.php”:

```
index.php  x

src > index.php > ...
      jalfalc, 4 weeks ago | 2 authors (JorgeAlfonso and one other)
1  <?php
2  ini_set('display_errors', 1);
3  ini_set('display_startup_errors', 1);
4  error_reporting(E_ALL);
5
6  session_start();
7
8  $controlador = "Principal";
9
10 if(!empty($_GET['controlador'])){
11     $controlador = $_GET['controlador'];
12 }
13
14 if (file_exists("Controller/" . $controlador . "_Controller.php")){
15     require_once ("Controller/" . $controlador . "_Controller.php");
16 }else{
17     echo "Error, el controlador requerido no existe o no se encuentra";
18 }
19
20 ?> JorgeAlfonso, last month • creada estructura inicial del proyecto y
```

Es el archivo que controla todo el flujo del proyecto. Podemos considerarlo el router del proyecto ya que todo pasa a través de él. Este archivo se encarga de llamar a cada controlador según el parámetro ‘controlador’ que se le pase en la URL por método GET. Por defecto tiene el valor ‘Principal’, por lo que si no pasamos un controlador en la URL este llamará a “Principal\_Controller.php”, y este nos mostrará la vista Principal (la de inicio), por lo que por defecto si no hay controlador siempre iremos a inicio.

A partir de ahora **para explicar los archivos que componen cada sección vamos a utilizar rutas relativas**, que va a ser lo óptimo para entender a qué archivo nos referimos ya que la ruta relativa siempre va a ser la misma independientemente de la ubicación del proyecto.

## 5.1 Header y Footer

El Header y Footer son comunes en todas las secciones, ya que en todas las vistas se utiliza “include” para añadir ambos y evitar repetir código. Están compuestos por los siguientes archivos:

- Header:

- **src\View\Header\Header\_View.php**: es el código HTML para crear el header, contiene un botón hamburguesa que despliega un modal con las opciones de navegación para la versión Tablet/móvil. Podemos diferenciar unas opciones de navegación diferentes para usuarios logeados y no logeados.
- **src\css\header\header.css**: da estilo al menú de navegación, y tiene media queries para hacerlo responsive según el tamaño del dispositivo.
- **src\js\header.js**: da la funcionalidad necesaria para que en versión Tablet/móvil se pueda abrir el modal con las opciones de navegación.

- Footer:

- **src\View\Footer\Footer\_View.php**: es el código HTML para mostrar la información del footer.
- **src\css\footer\footer.css**: da estilo al código HTML del Footer\_View.php.

Para que todas las vistas utilicen el header y footer se ha añadido a todas las vistas esto:

```
<?php include 'Header/Header_View.php';?>
```

Y esto:

```
<?php include 'Footer/Footer_View.php';?>
```

Y se le ha asignado el <link> apuntando al estilo de ambos en el <head> de cada vista:



```
<link rel="stylesheet" href="css/header/header.css">  
<link rel="stylesheet" href="css/footer/footer.css">
```

Aparte de esto, para que funcione el menú de navegación en versión móvil se ha añadido la etiqueta <script> apuntando a “principal.js” al final de cada vista:

```
<script src="js/principal/principal.js"></script>
```

## 5.2 Sección “Inicio”

Sección accesible en todo momento (con y sin sesión iniciada). A esta sección se accede al abrir la web en el navegador por primera vez (index.php la muestra por defecto) o al hacer clic en “Inicio” en el menú de navegación:



En esta sección están involucrados los siguientes archivos:

- **src\Controller\Principal\_Controller.php:** al ser llamado por “index.php” muestra la vista “Principal\_View.php”.
- **src\View\Principal\_View.php:** es la vista de “Inicio”, donde encontramos un carrusel con un botón que redirige a la sección “Reservar”, un cartel de servicios con un botón que redirige a la sección “Servicios”, la sección de “Contacto”, y el horario de la peluquería.
- **src\css\principal\principal.css:** se encarga de dar estilo a “Principal\_View.php”, cabe destacar que la sección es completamente responsive tanto para Tablet como para móvil.
- **src\js\principal\principal.js:** es el encargado de controlar el funcionamiento del carrousel. En este script se crean los botones para cambiar de imagen en el carrusel, y el intervalo de tiempo para cambiar de imagen automáticamente. En caso de que se cambie de imagen manualmente pulsando uno de los botones del carrusel el script ajusta automáticamente de nuevo el intervalo de tiempo para mostrar la siguiente imagen.

## 5.3 Sección “Servicios”

Sección accesible en todo momento (con y sin sesión iniciada). A esta sección se accede al hacer clic en “Servicios” en el menú de navegación:



En esta sección están involucrados los siguientes archivos:

- **src\Controller\Servicios\_Controller.php:** su función es únicamente llamar a la vista “Servicios\_View.php” al ser llamado.
- **src\View\Servicios\_View.php:** Es el código HTML de la sección. En esta sección gran parte del HTML es generada por “servicios.js”.
- **src\css\servicios\servicios.css:** estila la vista “Servicios\_View.php”, y hace que la sección sea responsive. El teléfono los servicios se muestran en una sola fila, mientras que en ordenador se muestran de 3 en 3.
- **src\js\servicios\servicios.js:** en esta sección tiene múltiples funciones:
  - o Genera las tarjetas de los servicios.
  - o Genera la funcionalidad para abrir un modal con detalles del servicio al pulsar “Más información”.
  - o Genera las tarjetas de los testimonios.

## 5.4 Sección “Contacto”

Sección accesible en todo momento (con y sin sesión iniciada). A esta sección se accede al hacer clic en “Contacto” en el menú de navegación:



No es una sección como tal, lo que hace es redirigir a “Inicio”, y dentro de “Inicio” a la sección “Contacta con nosotros”. Esto se consigue gracias a este elemento <a> del “src\View\Header\Header\_View.php”:

```
<li><a href="index.php?controlador=Principal#contacto">Contacto</a></li>
```

Por tanto, los archivos implicados en esta “sección” son:

- **src\View\Principal\_View.php.**
- **src\css\principal\principal.css**

## 5.5 Sección “Iniciar sesión”

Sección accesible únicamente con sesión no iniciada. En caso de querer acceder con sesión iniciada se redirigirá a la sección “Inicio”. A esta sección se accede al hacer clic en “Iniciar sesión” en el menú de navegación:



Los archivos involucrados en esta sección son:

- **src\Controller>Login\_Controller.php:** encargado de mostrar la vista “Login\_View.php” o “Registro\_View”, según el valor del parámetro “action” de la url, que se lee mediante el método GET. Si action recibe un valor aleatorio muestra por defecto la vista de login.
- **src\View>Login\_View.php:** si “Login\_Controller.php” recibe el parámetro “action” con valor “Login” se muestra esta vista, la cual muestra el formulario de inicio de sesión.
- **src\View\Registro\_View.php:** si “Login\_Controller.php” recibe el parámetro “action” con valor “Registro” se muestra esta vista, la cual muestra el formulario de registro.
- **src\css\login\login.css:** da estilo tanto a la vista “Login\_View.php” como a “Registro\_View.php”. Cabe destacar que centra el formulario de login y registro, que pone una imagen de fondo en ambas secciones y que ambas son responsive.

Todos los archivos mencionados anteriormente son los involucrados de mostrar las vistas de login y registro. Ahora vamos a explicar qué archivos se encargan de la funcionalidad de ambas secciones.

Para la sección de Login:

- **src\js\login\validarLogin.js:** es el encargado de validar que se hayan introducido ambos campos en el formulario de inicio de sesión. En caso de faltar algún campo (o ambos) muestra un mensaje de error descriptivo en color rojo.
- **src\Model\Usuario.php:** utilizado por “Verify\_Controller” para crear un usuario con las credenciales introducidas en el formulario de inicio de sesión.
- **src\Model\DAO\MySqlDAO.php:** el controlador “Verify\_Controller.php” utiliza el método “validarUsuario(Usuario \$usuario)” para comprobar si el usuario existe en la tabla “Usuarios”.
- **src\Controller\Verify\_Controller.php:** si nuestro script “validarUsuario.js” no detecta errores al enviar el formulario se llama a este controlador, el cual se encarga de nuevo de verificar que todos los campos han sido rellenados, y si se cumple esto se crea un objeto “Usuario” y se comprueba en la tabla “Usuarios” de la base de datos si hay algún usuario que coincida con las credenciales. Cabe destacar que las credenciales en base de datos se guardan hasheadas, y que la comparación se realiza hasheando la contraseña introducida por el usuario y comparándola directamente con el hash de las que están guardadas en base de datos. Si una coincide se da por válido ese usuario y se inicia sesión, de lo contrario se redirige a “Login\_View.php” mostrando un mensaje de error.

Para la sección de Registro:

- **src\js\registro\validarRegistro.js:** este script se utiliza para validar los campos introducidos en el formulario de registro. En caso de que falte o falle alguna muestra un mensaje en el campo específico informando del fallo. Las restricciones que se deben cumplir son las siguientes:
  - Nombre: debe tener menos de 20 caracteres.
  - Apellidos: debe tener menos de 30 caracteres.
  - Telefono: debe tener 9 números, sin prefijo.
  - Email: debe tener un formato de email válido (email real).
  - Contraseña: debe tener al menos 8 caracteres.
  - Repetir contraseña: debe coincidir con el campo “Contraseña”.
- Además de esto, **todos los campos son obligatorios.**
- **src\Model\Usuario.php:** utilizado por “Registro\_Controller” para crear un usuario con las credenciales introducidas en el formulario de registro.

- **src\Model\DAO\MySqlDAO.php:** el controlador “Registro\_Controller.php” utiliza el método “registrarUsuario(Usuario \$usuario)” para insertar un registro en la tabla “Usuarios” con los datos del nuevo usuario. Este método comprueba previamente que no se esté usando ya el correo introducido en el registro, y si no se está utilizando se inserta el nuevo Usuario.
- **src\Controller\Registro\_Controller.php:** en este controlador se vuelve a validar que los campos lleguen y no están vacíos, se crea un usuario con los datos introducidos por el usuario y se invoca a “registrarUsuario(Usuario \$usuario)” de Mysql.php. Si todo ha ido bien se redirige a la sección “Mis servicios” y se mantiene la sesión iniciada, si algo falla o el correo ya está en uso se informa del error.

## 5.6 Sección “Mis citas”

Sección accesible únicamente con sesión iniciada. En caso de querer acceder sin sesión iniciada se redirigirá a la sección “Iniciar sesión”. A esta sección se accede:

- Al hacer clic en “Iniciar sesión” en el menú de navegación (al tener sesión iniciada su menú de navegación mostrará esta sección):



- Al haber iniciado sesión o registrarse: si el inicio de sesión o el registro son exitosos se redirigirá automáticamente a esta sección.

Los archivos implicados en esta sección son los siguientes:

- **src\Controller\MisReservas\_Controller.php:** este controlador valida la sesión, carga y formatea las reservas para mostrarlas, gestiona la modificación de reservas (carga datos en el formulario de “EditarReserva\_View.php”, valida los datos y actualiza la reserva), y también permite cancelar una reserva ya existente (combinado con eliminarReserva.js, que es el que controla el envío del formulario para eliminar la cita), todo usando mensajes flash en \$\_SESSION para notificar al usuario tanto de errores como de éxito en las operaciones.

- **src\js\reservas\eliminarReserva.js:** En el DOM, busca todos los formularios con clase .form-eliminar (cada reserva lleva uno para “Eliminar”). Intercepta el evento submit de cada form, muestra un cuadro de confirmación nativo (“¿Seguro que deseas cancelar esta cita?”) y, si el usuario acepta, permite que el formulario se envíe para invocar la acción de eliminación en el controlador; si cancela, previene el envío. así se separa la lógica de confirmación del HTML y se centraliza en un solo archivo JavaScript.
- **src\js\reservas\cargarHoras.js:** script utilizado para cargar las horas disponibles del día seleccionado en el calendario de “Reservar” y “EditarReserva\_View.php”. Tiene en cuenta el horario de la peluquería, por eso los lunes no hay horas disponibles, y el los fines de semana hay un número reducido de horas disponibles. Primero se pide a “MisReservas\_Controller.php” que compruebe las citas reservadas en la fecha seleccionada, y luego pinta todas las horas disponibles sin incluir las que ya han sido reservadas.
- **src\View\MisReservas\_View.php:** es la vista de la sección “Mis reservas”. Muestra todas las citas del usuario en una tabla con columnas de Fecha, Servicio y Acciones. Si hay mensajes flash en sesión, los muestra antes de la tabla. Cada fila incluye un enlace “Modificar” que lleva al formulario de edición y un formulario “Eliminar” (interceptado por JavaScript para confirmar antes de enviar). Si no hay reservas, enseña un mensaje y un botón “Reservar cita” que redirige al formulario de reservas. También carga los estilos del header, footer y la vista de “Mis citas”, y al pie incluye los scripts header.js (para el menú de navegación) y eliminarReserva.js (para el cuadro de confirmación al eliminar).
- **src\Model\DAO\MySqlDAO.php:** MySqlDAO.php actúa como capa de acceso a datos para todo lo relacionado con “Mis citas”. En concreto, se utilizan los siguientes métodos:
  - o **obtenerReservasUsuario(\$userId):** retorna todas las reservas del usuario (fecha, hora, servicio, etc.) para que el controlador las formatee y pase a la vista de listado.
  - o **getReservaPorId(\$reservaId):** devuelve los datos de una reserva concreta (incluyendo servicio, fecha y hora) para llenar el formulario de edición.

- **actualizarCita(\$reservaId, \$servicio, \$fecha, \$hora)**: escribe en la base de datos los cambios que el usuario realice sobre una cita existente.
- **eliminarCita(\$reservaId)**: borra de la tabla la reserva seleccionada cuando el usuario la cancela

El caso de “**Modificar cita**” es algo más complejo que el resto, por lo que voy a explicarlo individualmente:

1. Cuando el usuario hace clic en “Modificar” en la tabla de “MisReservas\_View.php”, el enlace apunta a `index.php?controlador=MisReservas&action=Modificar&id={id}`. (MODIFICADO: el “id” se pasa por POST)
2. Eso dispara el método `Modificar()` en “MisReservas\_Controller.php”, que:
  - Recupera la reserva concreta llamando a “`getReservaPorId($id)`” del DAO.
  - Formatea la fecha y la hora para mostrarlas como “Fecha de la cita actual” y “Servicio actual”.
  - Carga la vista “`EditarReserva_View.php`” con toda la información de esa reserva.
3. “`EditarReserva_View.php`” muestra el formulario con el calendario (Flatpickr), el select de servicios y las horas libres. Además de esto, gracias al script “`editarReserva.js`” se muestra la fecha de la cita actual y el servicio actual, para que el usuario tenga la referencia de los datos de la cita que está actualizando.
4. Al pulsar “Guardar cambios” en ese formulario, se envía un POST a “`index.php?controlador=MisReservas&action=Actualizar`”.
5. En “`MisReservas_Controller.php`”, el método “`Actualizar()`” recibe esos datos (`id, servicio, fecha, hora`), valida que no estén vacíos, llama a “`$this->dao->actualizarCita(...)`” y, según el resultado, fija un mensaje flash y redirige de nuevo a “`Mostrar()`”.

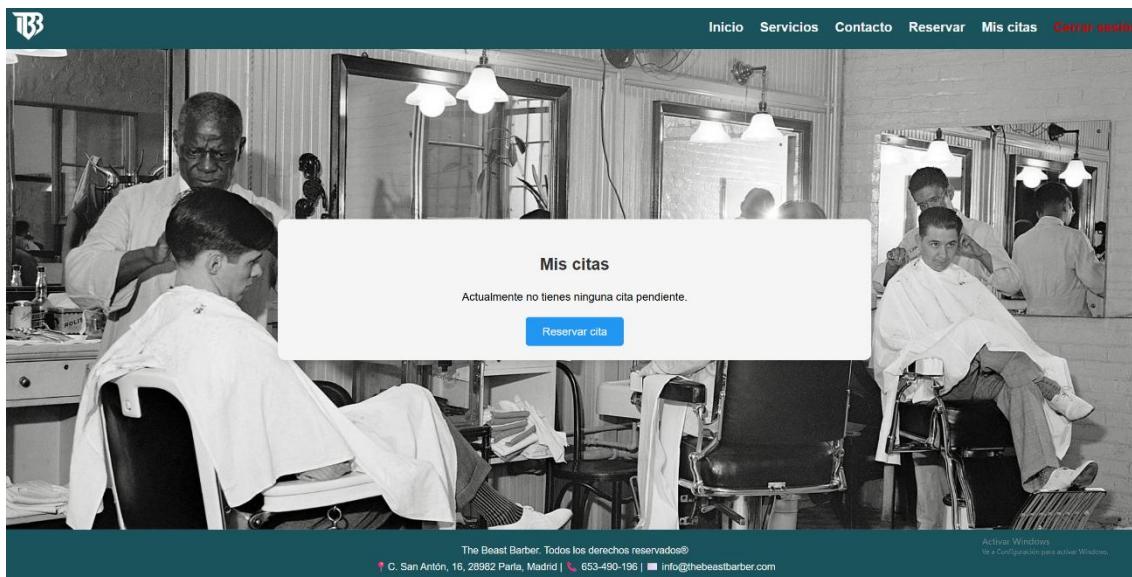
## 5.7 Sección “Reservar”

Sección accesible únicamente con sesión iniciada. En caso de querer acceder sin sesión iniciada se redirigirá a la sección “Iniciar sesión”. Aún con la sesión no iniciada esta sección está visible en el menú de navegación para redirigir al usuario al login, lo que considero que da a entender al usuario que es necesario iniciar sesión para acceder al contenido de la sección. A esta sección se puede acceder de varias maneras:

- Al hacer clic en “Reservar” en el menú de navegación con sesión iniciada:



- Al estar en “Mis citas” con sesión iniciada y sin citas reservadas, donde se muestra el botón “Reservar cita”, que redirige a esta sección:



Los archivos implicados en esta sección son los siguientes:

- **src\Controller\Reservas\_Controller.php**: Este controlador realiza las siguientes tareas:
  - **Verifica que el usuario esté logueado**; si no, redirige a “Iniciar sesión”.
  - **Método “Mostrar()”**:

1. Lee posibles mensajes de `$_SESSION['success_reserva']` y `$_SESSION['error_reserva']` (éxito/error de la última operación) y los limpia.
2. Carga la vista “Reservas\_View.php”.

- **Método Horas():**

1. Recibe por GET fecha y servicio.
2. Consulta a `MySqlDAO::getHorasReservadas($fecha)` para obtener qué franjas ya están ocupadas ese día.
3. Construye internamente las franjas horarias disponibles según el horario de la barbería (martes–viernes 09:00–15:00 y 17:00–21:00; sábados y domingos 09:00–14:00 y 15:00–18:00; lunes cerrado).
4. Filtra las horas ocupadas y devuelve un JSON { horas: [ "09:00", "09:30", ... ] }

- **Método “Confirmar()”:**

- Recoge `$_POST['servicio']`, `$_POST['fecha']` y `$_POST['hora']`
- Valida que todos los campos estén presentes; si falta alguno, guarda `$_SESSION['error_reserva'] = 'Falta elegir servicio, fecha u hora.'` y redirige a `Mostrar()`.
- Llama a `MySqlDAO::reservarCita($usuarioId, $servicio, $fecha, $hora)`
- Si devuelve true, formatea la fecha a “DD-MM-YYYY” y guarda `$_SESSION['success_reserva'] = "Cita para {$fechaFormateada} a las {$hora} confirmada."`
- Si devuelve false (ya ocupada), guarda `$_SESSION['error_reserva'] = 'Lo siento, esa hora ya no está disponible.'`
- Redirige de nuevo a `Mostrar()`

- `src\View\Reservas_View.php`: Vista principal de “Reservar cita”. Incluye:
  - El header y footer común a todas las secciones.
  - Muestra un título “Reservar Cita”

- Si existe <?= \$htmlspecialchars(\$mensaje) ?> (éxito) lo pinta con clase .reserva-mensaje.success; si existe <?= \$htmlspecialchars(\$error) ?> (error), lo pinta con .reserva-mensaje.error.
- Formulario <form id="form-reserva" action="index.php?controlador=Reservas&action=Confirmar" method="post" novalidate> que contiene:
  - < select > con los servicios disponibles (el primer option disabled, solo para mostrar un mensaje en el select).
  - Calendario Flatpickr para seleccionar la fecha de la cita.
  - Grid de horas disponibles según la fecha seleccionada en el calendario.
  - Campo oculto para enviar la hora.
  - Botón de confirmación que inicialmente aparece deshabilitado hasta que se ha seleccionado servicio, fecha y hora.
  - Scripts de Flatpickr para que el calendario aparezca en español y script “cargarHoras.js” para cargar las horas libres según la fecha seleccionada.
- src\js\reservas\cargarHoras.js: al cargar el DOM realiza lo siguiente:
  - Inicializa Flatpickr sobre #fecha en modo inline, con idioma español, mes/año desplegados, límite minDate: "today" y formato Y-m-d
  - Escucha el evento change en el selector #servicio y también el callback onChange de Flatpickr para llamar a recargarHoras().
  - La función recargarHoras():
    - Si no hay fecha o servicio, limpia #horas-disponibles y deja el botón “Confirmar” deshabilitado.
    - Hace un `fetch("index.php?controlador=Reservas&action=Horas&fecha=Y YYYY-MM-DD&servicio=...")`.
    - Recibe JSON { horas: [ "09:00", "09:30", ... ] }.
    - Genera botones <button> dentro de “#horas-disponibles” por cada hora libre. Al hacer clic en uno:
      - Desmarca cualquier botón previamente seleccionado.
      - Marca el nuevo con la clase seleccionada.

- Guarda la hora en #hora-seleccionada.value y habilita #btn-confirmar.
  - Después de pintar, resetea internamente horaSel y deja “Confirmar” deshabilitado hasta que el usuario pulse sobre una hora.
  - Configura setInterval(recargarHoras, 3000) para recargar las horas cada 3 segundos y mantenerlas sincronizadas con posibles cambios en otros clientes.
  - Ejecuta recargarHoras() una vez al final para inicializar el grid aunque aún no haya fecha/servicio.
- src\Model\DAO\MySqlDAO.php: respecto a las consultas a base de datos en esta sección se utilizan:
1. getHorasReservadas(string \$fecha): array :
    - Ejecuta “SELECT TIME\_FORMAT(hora, '%H:%i') AS hora FROM citas\_reservadas WHERE fecha = ?”
    - Devuelve un array de strings (“HH:MM”) con las horas ya ocupadas en ese día.
  2. reservarCita(int \$usuarioId, string \$servicio, string \$fecha, string \$hora): bool
    - Ejecuta INSERT INTO citas\_reservadas (usuario\_id, servicio, fecha, hora) VALUES (?, ?, ?, ?)
    - Si tiene éxito, retorna true; si se viola la clave única (fecha, hora), retorna false.

**En conjunto, Reservas\_Controller.php, Reservas\_View.php, cargarHoras.js y el DAO tienen el siguiente flujo:**

1. El usuario entra en “Reservar cita” → se llama Reservas\_Controller::Mostrar()  
→ carga “Reservas\_View.php”
2. Flatpickr se inicializa y muestra el calendario en español; el <select> de servicios queda listo.

3. Cuando el usuario elige un servicio y hace clic en algún día válido del calendario, el script cargarHoras.js hace AJAX a ?controlador=Reservas&action=Horas&fecha=...&servicio=....
4. El controlador Reservas\_Controller::Horas() filtra las franjas según horario, quita las horas ocupadas con getHorasReservadas(), devuelve JSON con horas libres
5. El script pinta esos botones y, al pulsar uno, guarda la hora en un campo oculto e habilita “Confirmar cita”
6. Al pulsar “Confirmar cita”, el formulario envía por método POST ?controlador=Reservas&action=Confirmar.
7. Reservas\_Controller::Confirmar() valida datos, llama a MySqlDAO::reservarCita() :
  - a. Si tiene éxito, formatea la fecha a “DD-MM-YYYY” y guarda \$\_SESSION['success\_reserva'] = "Cita para dd-mm-YYYY a las HH:MM confirmada.".
  - b. Si falla (hora ya ocupada), guarda \$\_SESSION['error\_reserva'] = "Lo siento, esa hora ya no está disponible."

Finalmente redirige a Mostrar().

8. Mostrar() vuelve a cargar “Reservas\_View.php”, donde se imprime el mensaje de éxito/error en la parte superior.

## 6. Aspectos importantes a tener en cuenta

Hay una serie de aspectos peculiares en el proyecto que son importantes de mencionar.

### 6.1 Conexión a la base de datos

La clase “MySqlDAO” utiliza “src\Model\DAO\Conexion.php” para obtener una conexión a la base de datos. Inicialmente la clase “Conexión” tenía las credenciales en el código, y tenía un método estático para devolver una conexión a la base de datos, pero no es una buena práctica y, a pesar de no desplegar el proyecto he preferido crear un archivo “.env” en la raíz del proyecto simulando a las variables de entorno que usaría para guardar las credenciales de la base de datos.

Ahora mi clase “Conexión” se encarga de leer las credenciales del “.env” y de devolver una conexión utilizando dichas credenciales. Por último, cabe destacar que los métodos de “Conexion.php” siguen siendo estáticos.

### 6.2 Directorio “img”

En la raíz del proyecto encontramos “img”, que es el directorio donde se guardan todas las imágenes utilizadas en el proyecto. Está organizado en subdirectorios con nombres intuitivos para saber en qué sección se utiliza cada imagen. Estoy mencionando esto aquí porque en todo el documento no he hablado de las imágenes y quería por lo menos mencionarlo aquí.

### 6.3 Directorio “documentación”

En este directorio se van a guardar todos los documentos relacionados con el proyecto: diagramas, manuales, esquemas, pruebas, etc. Estará dividido por subsecciones para que quede bien organizado todo.

## 7. Bibliografía

- Web oficial de XAMPP:  
<https://www.apachefriends.org/es/download.html>
- Web oficial de Visual Studio Code:  
<https://code.visualstudio.com/>
- Enlace al repositorio Github del proyecto ‘The Beast Barber’:  
<https://github.com/jalfalc/TFG-TheBeast>
- Guía de instalación y configuración de Git:  
<https://www.youtube.com/watch?v=wHh3IgJvXcE>