

INSTITUTO TÉCNICO RICALDONE
DEPARTAMENTO DE SISTEMAS INFORMATICOS.



MANUAL TECNICO PARA EL SISTEMA “ASGARD”

Tercer año bachillerato Desarrollo de Software.

Grupo No.1

Sección A

Integrantes:

N°	Nombres:	Carnet:
•	-Villegas Gómez Diego Fernando	20220382
•	-Melgar Echegoyen Katherine Ariana	20220125
•	-Cornejo Godínez Gabriel Alessandro	20220005
•	- Escamilla Ruiz Roberto Alfonso	20220356
•	- Herrera López Genesis Tatiana	20220108

Docente: Emerson Alexander González Rodríguez.

Introducción:	3
Tecnologías y herramientas empleadas.....	4
Estructura de la base de datos.	5
Diccionario de datos.....	6
Arquitectura de software.....	9
Estructura del proyecto.....	12
Diseño de la aplicación.....	13
Buenas prácticas de desarrollo.....	15
Requerimientos de hardware y software.	16
Instalación y configuración.	18

Introducción:

En este manual, se ofrece una guía técnica sobre la configuración, uso y especificaciones del sistema “ASGARD”. Se cubrirán aspectos como la estructura de la base de datos, diseños, requerimientos y los componentes esenciales de la arquitectura. También se detallarán las tecnologías utilizadas y las mejores prácticas para su implementación y funcionamiento. Este manual está dirigido tanto a usuarios finales como al personal técnico responsable de su administración y soporte.

Tecnologías y herramientas empleadas.

El sistema “ASGARD” ha sido desarrollado utilizando una variedad de tecnologías y herramientas que garantizan su rendimiento, escalabilidad y seguridad. A continuación, se describen las principales tecnologías y herramientas utilizadas en la implementación del sistema:

HTML5 y CSS3: Se utilizan para la creación de las vistas del sistema, asegurando una estructura semántica y un diseño responsivo que se adapta a diferentes dispositivos y resoluciones de pantalla.

Bootstrap: Este framework de diseño front-end permite una rápida creación de interfaces atractivas y funcionales, facilitando la implementación de elementos visuales modernos y responsivos.

JavaScript (Vanilla JS): se utiliza para la programación asíncrona y permite la manipulación del DOM y la interacción dinámica con las vistas sin recargar la página. Esto hace que la experiencia de usuario sea fluida y eficiente.

PHP: PHP se utiliza en el backend del sistema y permite la creación de API que manejan la lógica comercial y la comunicación con la base de datos, utilizando prácticas de programación orientada a objetos para mejorar la organización y reutilización del código.

MariaDB y phpMyAdmin se utilizan como sistemas de gestión de bases de datos relacionales y herramientas de gestión. MariaDB permite el almacenamiento y gestión estructurados de la información del sistema.

Mientras que phpMyAdmin facilita la gestión y consulta visuales de la base de datos.

Control de versiones (Git y GitHub): Se utilizan para el manejo y control del proyecto, permitiendo la colaboración entre todos los miembros del equipo, teniendo un claro control y manejo de las versiones y actualizaciones del proyecto.

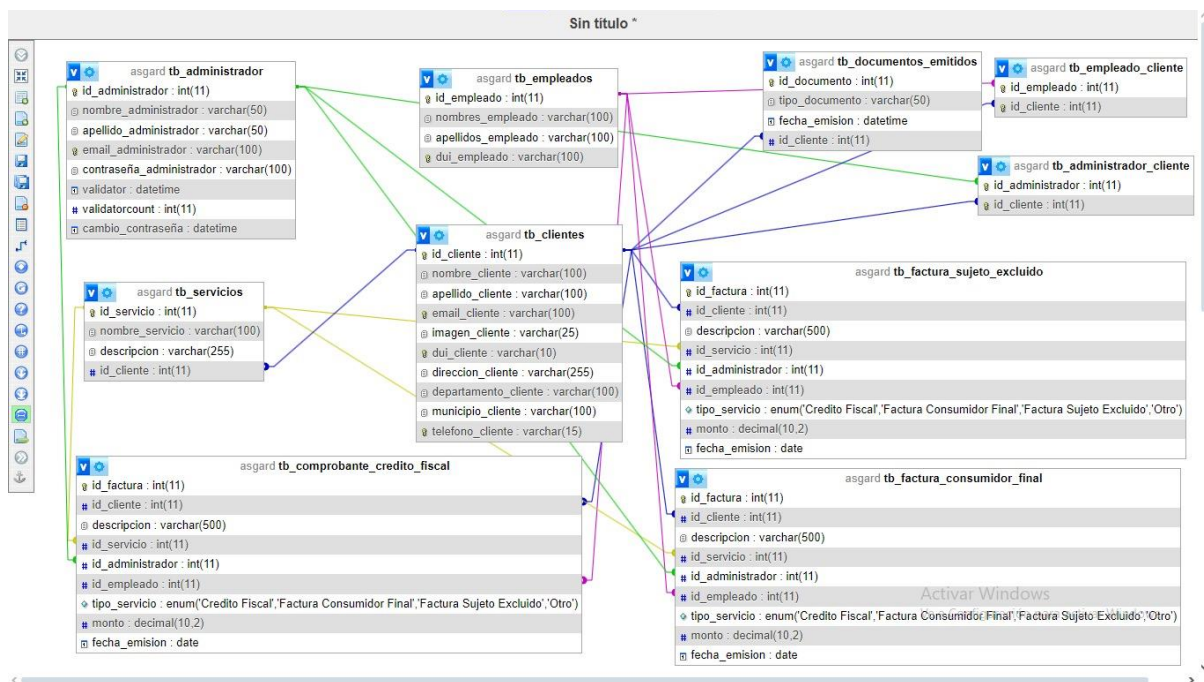
Zapier: Se utiliza como método de automatización de tareas, facilitando la integración entre diferentes aplicaciones y mejorando la eficiencia operativa del sistema, así ahorrando el tiempo en tareas que se volvían repetitivas.

Estructura de la base de datos.

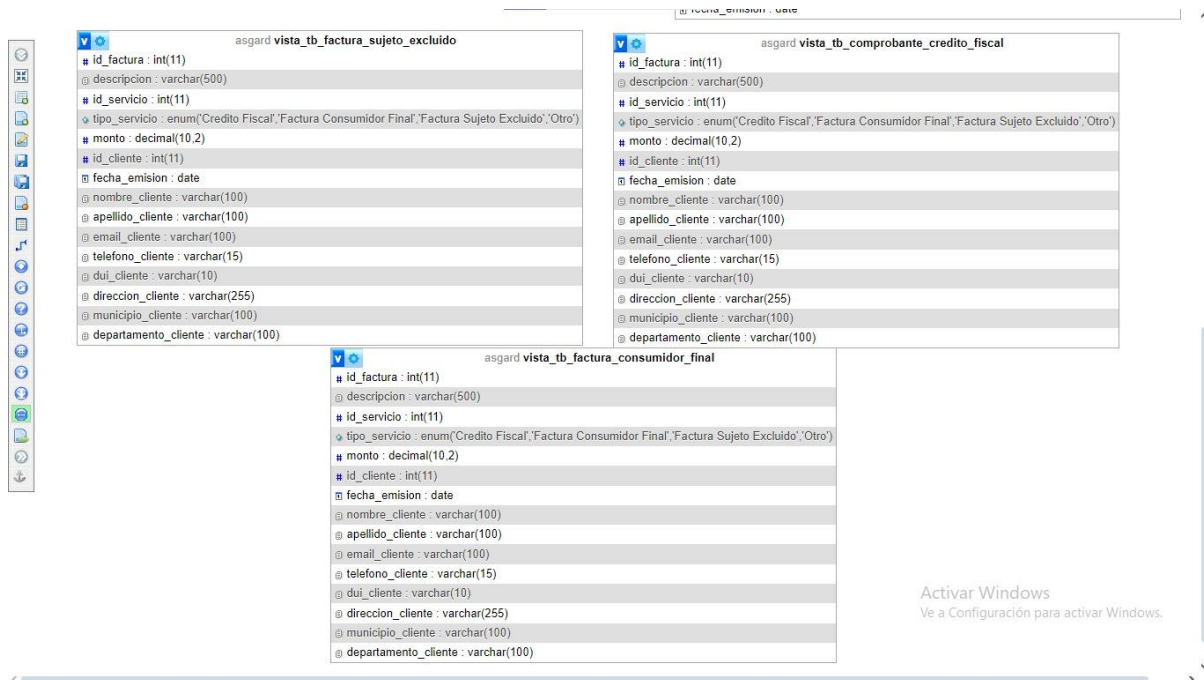
Modelo relacional

Diagrama de dominio de la base de datos del proyecto PTC.

Primera parte del diagrama:



Segunda parte del diagrama:



Script de la base de datos

En el siguiente enlace se podrá encontrar el script de la base de datos con triggers, procedimientos almacenados, funciones y estructuras de tablas.

<https://drive.google.com/file/d/1QnH86AXeYQxveqiFnQrLF1rvLmIEZyKH/view?usp=sharing>

Diccionario de datos.

tb_administrador

Columna	Tipo	Nulo	Predeterminado	Enlaces a	Comentarios	Tipo de medio
id_administrador (Primaria)	int(11)	No				
nombre_administrador	varchar(50)	No				
apellido_administrador	varchar(50)	No				
email_administrador	varchar(100)	Sí	NULL			
contraseña_administrador	varchar(100)	No				
validator	datetime	Sí	NULL			
validatorcount	int(11)	Sí	0			
cambio_contraseña	datetime	Sí	(current_timestamp() + interval 90 day)			

Índices

Nombre de la clave	Tipo	Único	Empaquetado	Columna	Cardinalidad	Cotejamiento	Nulo	Comentario
PRIMARY	BTREE	Sí	No	id_administrador	0	A	No	
email_administrador	BTREE	Sí	No	email_administrador	0	A	Sí	

tb_administrador_cliente

Columna	Tipo	Nulo	Predeterminado	Enlaces a	Comentarios	Tipo de medio
id_administrador (Primaria)	int(11)	No		tb_administrador -> id_administrador		
id_cliente (Primaria)	int(11)	No		tb_clientes -> id_cliente		

Índices

Nombre de la clave	Tipo	Único	Empaquetado	Columna	Cardinalidad	Cotejamiento	Nulo	Comentario
PRIMARY	BTREE	Sí	No	id_administrador	0	A	No	
				id_cliente	0	A	No	
id_cliente	BTREE	No	No	id_cliente	0	A	No	

tb_clientes

Columna	Tipo	Nulo	Predeterminado	Enlaces a	Comentarios	Tipo de medio
id_cliente (Primaria)	int(11)	No				
nombre_cliente	varchar(100)	No				
apellido_cliente	varchar(100)	No				
email_cliente	varchar(100)	Sí	NULL			
imagen_cliente	varchar(25)	No				
dui_cliente	varchar(10)	Sí	NULL			
direccion_cliente	varchar(255)	Sí	NULL			
departamento_cliente	varchar(100)	No				
municipio_cliente	varchar(100)	Sí	NULL			
telefono_cliente	varchar(15)	No				

Índices

Nombre de la clave	Tipo	Único	Empaquetado	Columna	Cardinalidad	Cotejamiento	Nulo	Comentario
PRIMARY	BTREE	Sí	No	id_cliente	0	A	No	
telefono_cliente	BTREE	Sí	No	telefono_cliente	0	A	No	
email_cliente	BTREE	Sí	No	email_cliente	0	A	Sí	
dui_cliente	BTREE	Sí	No	dui_cliente	0	A	Sí	

tb_comprobante_credito_fiscal

Columna	Tipo	Nulo	Predeterminado	Enlaces a	Comentarios	Tipo de medio
id_factura (Primaria)	int(11)	No				
id_cliente	int(11)	Si	NULL	tb_clientes -> id_cliente		
descripcion	varchar(500)	No				
id_servicio	int(11)	Si	NULL	tb_servicios -> id_servicio		
id_administrador	int(11)	Si	NULL	tb_administrador -> id_administrador		
id_empleado	int(11)	Si	NULL	tb_empleados -> id_empleado		
tipo_servicio	enum('Credito Fiscal', 'Factura Consumidor Final', 'Factura Sujeto Excluido', 'Otro')	No				
monto	decimal(10,2)	Si	NULL			
fecha_emision	date	Si	NULL			

Índices

Nombre de la clave	Tipo	Único	Empaquetado	Columna	Cardinalidad	Cotejamiento	Nulo	Comentario
PRIMARY	BTREE	Si	No	id_factura	0	A	No	
id_cliente	BTREE	No	No	id_cliente	0	A	Si	
id_servicio	BTREE	No	No	id_servicio	0	A	Si	
id_empleado	BTREE	No	No	id_empleado	0	A	Si	
id_administrador	BTREE	No	No	id_administrador	0	A	Si	

tb_factura_consumidor_final

Columna	Tipo	Nulo	Predeterminado	Enlaces a	Comentarios	Tipo de medio
id_factura (Primaria)	int(11)	No				
id_cliente	int(11)	Si	NULL	tb_clientes -> id_cliente		
descripcion	varchar(500)	No				
id_servicio	int(11)	Si	NULL	tb_servicios -> id_servicio		
id_administrador	int(11)	Si	NULL	tb_administrador -> id_administrador		
id_empleado	int(11)	Si	NULL	tb_empleados -> id_empleado		
tipo_servicio	enum('Credito Fiscal', 'Factura Consumidor Final', 'Factura Sujeto Excluido', 'Otro')	No				
monto	decimal(10,2)	Si	NULL			
fecha_emision	date	Si	NULL			

Índices

Nombre de la clave	Tipo	Único	Empaquetado	Columna	Cardinalidad	Cotejamiento	Nulo	Comentario
PRIMARY	BTREE	Si	No	id_factura	0	A	No	
id_cliente	BTREE	No	No	id_cliente	0	A	Si	
id_servicio	BTREE	No	No	id_servicio	0	A	Si	
id_empleado	BTREE	No	No	id_empleado	0	A	Si	
id_administrador	BTREE	No	No	id_administrador	0	A	Si	

tb_factura_sujeto_excluido

Columna	Tipo	Nulo	Predeterminado	Enlaces a	Comentarios	Tipo de medio
id_factura (Primaria)	int(11)	No				
id_cliente	int(11)	Si	NULL	tb_clientes -> id_cliente		
descripcion	varchar(500)	No				
id_servicio	int(11)	Si	NULL	tb_servicios -> id_servicio		
id_administrador	int(11)	Si	NULL	tb_administrador -> id_administrador		
id_empleado	int(11)	Si	NULL	tb_empleados -> id_empleado		
tipo_servicio	enum('Credito Fiscal', 'Factura Consumidor Final', 'Factura Sujeto Excluido', 'Otro')	No				
monto	decimal(10,2)	No				
fecha_emision	date	Si	NULL			

Índices

Nombre de la clave	Tipo	Único	Empaquetado	Columna	Cardinalidad	Cotejamiento	Nulo	Comentario
PRIMARY	BTREE	Si	No	id_factura	0	A	No	
id_cliente	BTREE	No	No	id_cliente	0	A	Si	
id_servicio	BTREE	No	No	id_servicio	0	A	Si	
id_empleado	BTREE	No	No	id_empleado	0	A	Si	
id_administrador	BTREE	No	No	id_administrador	0	A	Si	

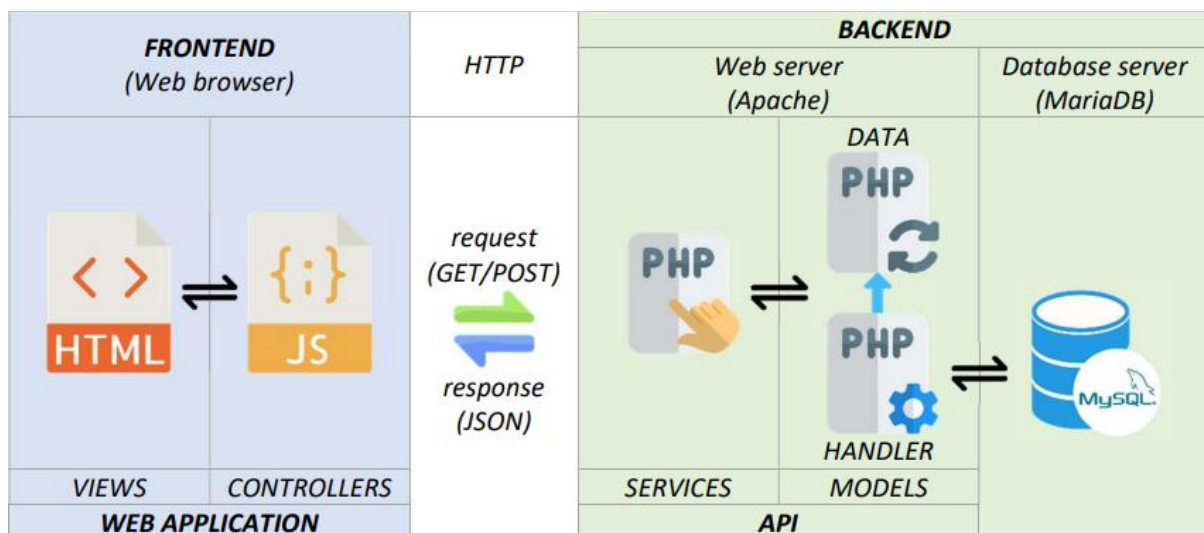
tb_servicios

Columna	Tipo	Nulo	Predeterminado	Enlaces a	Comentarios	Tipo de medio
id_servicio (Primaria)	int(11)	No				
nombre_servicio	varchar(100)	No				
descripcion	varchar(255)	Si	NULL			
id_cliente	int(11)	Si	NULL	tb_clientes -> id_cliente		

Índices

Nombre de la clave	Tipo	Único	Empaquetado	Columna	Cardinalidad	Cotejamiento	Nulo	Comentario
PRIMARY	BTREE	Si	No	id_servicio	0	A	No	
id_cliente	BTREE	No	No	id_cliente	0	A	Si	

Arquitectura de software.



Las vistas (views) son prácticamente páginas web elaboradas con HTML5 empleando Bootstrap, las cuales interactúan directamente con los controladores (controllers) programados en JavaScript mediante Vanilla JS. En

términos generales, la programación asíncrona con JavaScript permite realizar peticiones y recibir respuestas en

tiempo real para manipular el DOM (Document Object Model) según las necesidades de la aplicación, es decir,

es lo que hace funcionar el contenido dinámico en las páginas web.

La programación asíncrona facilita el manejo de la interacción del usuario con las vistas sin recargar la página

web, o sea, sin refrescar el navegador. Por medio de los controladores se hacen peticiones (request) y se reciben

respuestas (response) en formato JSON (JavaScript Object Notation). Cuando una petición implica enviar datos

como por ejemplo actualizar un registro, se remiten por medio del método POST, en cambio, si lo que se requiere

únicamente es solicitar datos como por ejemplo listar registros, se piden a través del método GET.

Las vistas y los controladores son los únicos elementos que funcionan del lado del cliente (frontend)

interpretados por el navegador web, donde la implementación de programación asíncrona puede realizarse de

diversas formas, no solo con Vanilla JS, por ejemplo, se podría optar por utilizar librerías como jQuery (no

recomendable), React, entre otras; o dar un salto más allá mediante la utilización de un framework basado en

JavaScript como Vue, Angular u otro similar.

Para cada página web existe un archivo JavaScript que actúa como controlador principal, capaz de gestionar

todas las peticiones y respuestas, ya sea de forma automática o por intervención del usuario. Dicha gestión se

realiza a través de diversos métodos, eventos y funciones; que dotan a la página web de toda la funcionalidad

necesaria para realizar una o varias tareas específicas. Por conveniencia, se recomienda que el nombre del

controlador principal sea el de la página web.

Por otro lado, tenemos los elementos del lado del servidor (backend) que llamaremos API (Application

Programming Interface – Interfaz de Programación de Aplicaciones –), la cual interactúa directamente con los

controladores por medio de la capa de servicios (SERVICES), recibiendo peticiones GET o POST y retornando

respuestas en formato JSON. La API reacciona según las acciones requeridas por los controladores, donde cada

acción está definida en la URL del archivo que la contiene, lo que recibe el nombre de endpoint.

La API está programada con PHP, instanciando en SERVICES una o más clases de los modelos (MODELS), es decir,

se implementa OOP (Object Oriented Programming – Programación Orientada a Objetos –). Además, en los

modelos se validan todos los datos de entrada del lado del servidor al momento de establecer (SET) los valores

a los atributos, específicamente en la capa DATA, como parte del encapsulamiento por si se escapa algún dato

erróneo o malicioso desde el frontend.

La capa SERVICES está formada por varios archivos o scripts, cada archivo contiene una o varias acciones

(endpoints). Algunas acciones pueden requerir autenticación del usuario, por lo que se debe validar para evitar

exponer datos sensibles o realizar operaciones sin control, previniendo de esta forma posibles ataques o robos

de información. También, aquí se obtienen las excepciones de la base de datos y se realiza el manejo de errores

en los valores de entrada.

Los modelos son una representación fidedigna de la base de datos, modelando generalmente cada tabla por

medio de dos clases: DATA y HANDLER. Cada clase debe elaborarse en un archivo, o por lo menos eso sugieren

las PSR (PHP Standards Recommendations – Recomendaciones de Estándares PHP –). En dichos modelos se

realizan las operaciones con la base de datos por medio de la capa HANDLER, donde se implementa la clase PDO

(PHP Data Object – Objetos de Datos PHP –).

La extensión PDO es una clase nativa en PHP a partir de la versión 5.1, la cual define una interfaz ligera para

trabajar con diferentes bases de datos a través de controladores específicos que proveen características propias

de cada base de datos, siendo MariaDB una de las opciones soportadas. PDO proporciona una capa de

abstracción de acceso a datos, lo que significa que, puede utilizarse con diversos gestores de bases de datos de

forma automática o realizando pequeñas configuraciones.

seguir




Seguimos esta lógica para llevar a cabo el desarrollo:

1. Construir la base datos.
2. Elaborar los mockups definiendo todos los detalles de diseño y tomando en cuenta la base de datos.
3. Crear las vistas (views) con un framework de acuerdo con los mockups.
4. Programar las clases de los modelos (models) con PHP según la base de datos, así:
 - a. Clase Handler: definir las propiedades a partir de las columnas de la tabla y luego un método por cada sentencia SQL.
 - b. Clase Data: crear un método set por cada propiedad e incluir la validación respectiva antes de asignar el valor.
5. Implementar los modelos para las acciones de los servicios (services) y probar con Postman o una herramienta similar.
6. Programar los controladores (controllers) con JavaScript.

Estructura del proyecto.



Proyecto web:

Index of /ASGARD/views

Name	Last modified	Size	Description
 Parent Directory		-	
 admin/	2024-06-27 15:02	-	
 public/	2024-06-27 13:27	-	

Apache/2.4.56 (Win64) OpenSSL/1.1.1t PHP/8.2.4 Server at localhost Port 80

Index of /ASGARD

Name	Last modified	Size	Description
 Parent Directory		-	
 api/	2024-06-27 13:27	-	
 asgard.sql	2024-06-27 13:27	4.5K	
 controllers/	2024-06-27 13:27	-	
 resources/	2024-06-27 13:27	-	
 views/	2024-06-27 13:27	-	

Apache/2.4.56 (Win64) OpenSSL/1.1.1t PHP/8.2.4 Server at localhost Port 80

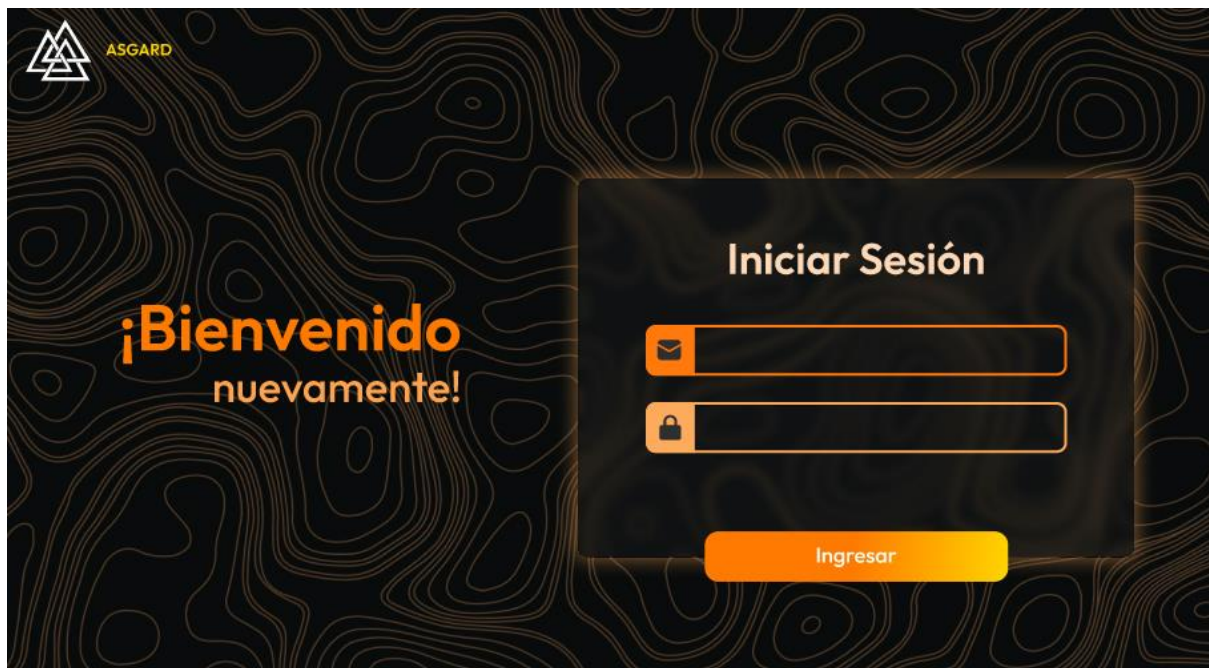
- proyecto-web/
 - |— src/
 - | |— componentes/
 - | |— páginas/
 - | |— estilos/
 - | |— utils/
 - |— public/
 - | |— index.html
 - |— package.json
 - |— README.md
-

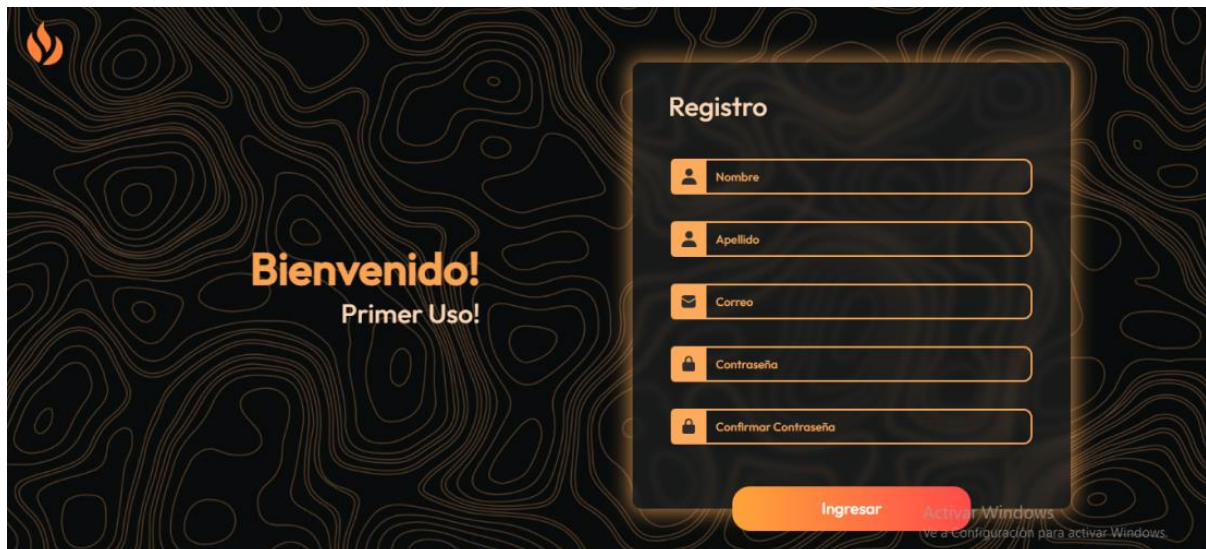
Diseño de la aplicación.

Paleta de colores



Para garantizar una experiencia de usuario agradable, la paleta de colores se basa en tonos naranja y se aplican de manera agradable en diferentes elementos del sistema. A continuación, imágenes de cómo se utiliza la paleta de colores.





Tamaños de pantallas soportados

En el proyecto Asgard, se ha utilizado Bootstrap 5.3, que proporciona soporte para diferentes tamaños de pantalla mediante un sistema de puntos de quiebre (breakpoints). Esto permite que el sitio sea completamente adaptable (responsive) a varios dispositivos, asegurando una correcta visualización y funcionalidad. Los tamaños de pantalla soportados son los siguientes:

- Extra small (xs): Menos de 576px (para dispositivos móviles pequeños).
- Small (sm): 576px o más (para smartphones más grandes).
- Medium (md): 768px o más (para tablets).
- Large (lg): 992px o más (para laptops y pantallas de escritorio pequeñas).
- Extra large (xl): 1200px o más (para pantallas de escritorio grandes).
- Extra extra large (xxl): 1400px o más (para pantallas muy grandes).

Estos tamaños aseguran que el sistema Asgard se adapte adecuadamente a cualquier resolución, proporcionando una experiencia de usuario óptima en dispositivos móviles, tablets, laptops y pantallas grandes.

Buenas prácticas de desarrollo.

Estándares de programación en javascript Los estándares de programación en JavaScript son un conjunto de pautas y convenciones que los desarrolladores siguen con el fin de escribir código de forma consistente, legible y mantenible. A continuación, se describen los estándares aplicados en el desarrollo del proyecto Asgard.

1. Nomenclatura de Variables y Funciones:

Nombres Descriptivos: Se utilizan nombres descriptivos en mayúsculas con guiones bajos para las constantes y variables, mientras que las funciones se nombran utilizando la convención camelCase.

Nombres Significativos: Los nombres de variables y funciones son claros e indican su propósito específico.

2. Indentación:

La indentación se refiere al uso de espacios al inicio de una línea de código para organizar visualmente el flujo y la jerarquía del código.

Consistente: Se emplea una indentación de 4 espacios por nivel, garantizando la legibilidad del código.

3. Punto y Coma:

Inclusión de Puntos y Comas: Se finalizan las declaraciones con punto y coma para prevenir posibles errores.

4. Declaración de Variables:

Uso de const y let: Se usa const para variables cuyo valor es inmutable, y let para aquellas que pueden cambiar.

5. Evitar el Uso de Variables Globales:

Encapsulación: Se minimiza el uso de variables globales para prevenir conflictos de nombres y mejorar la encapsulación.

6. Evitar la Escritura en Línea:

Legibilidad: Se evita el uso de múltiples declaraciones en una sola línea para mejorar la claridad del código.

7. Manejo de Errores:

Uso de try-catch: Se utilizan bloques try-catch para manejar errores de forma efectiva y proporcionar mensajes de error significativos.

Requerimientos de hardware y software.

Hardware:

	Mínimo	Recomendado
Núcleos de CPU (*)	2 x 1,8 GHz 32 bit (x86)	4 x 2,4 GHz 64 bits (x64)
RAM	8 GB de memoria del sistema disponible	12 GB
Espacio en disco	3,5 GB para instalaciones nuevas, 5 GB para actualizaciones (incluidos los archivos temporales necesarios durante la instalación)	N/D

Software:

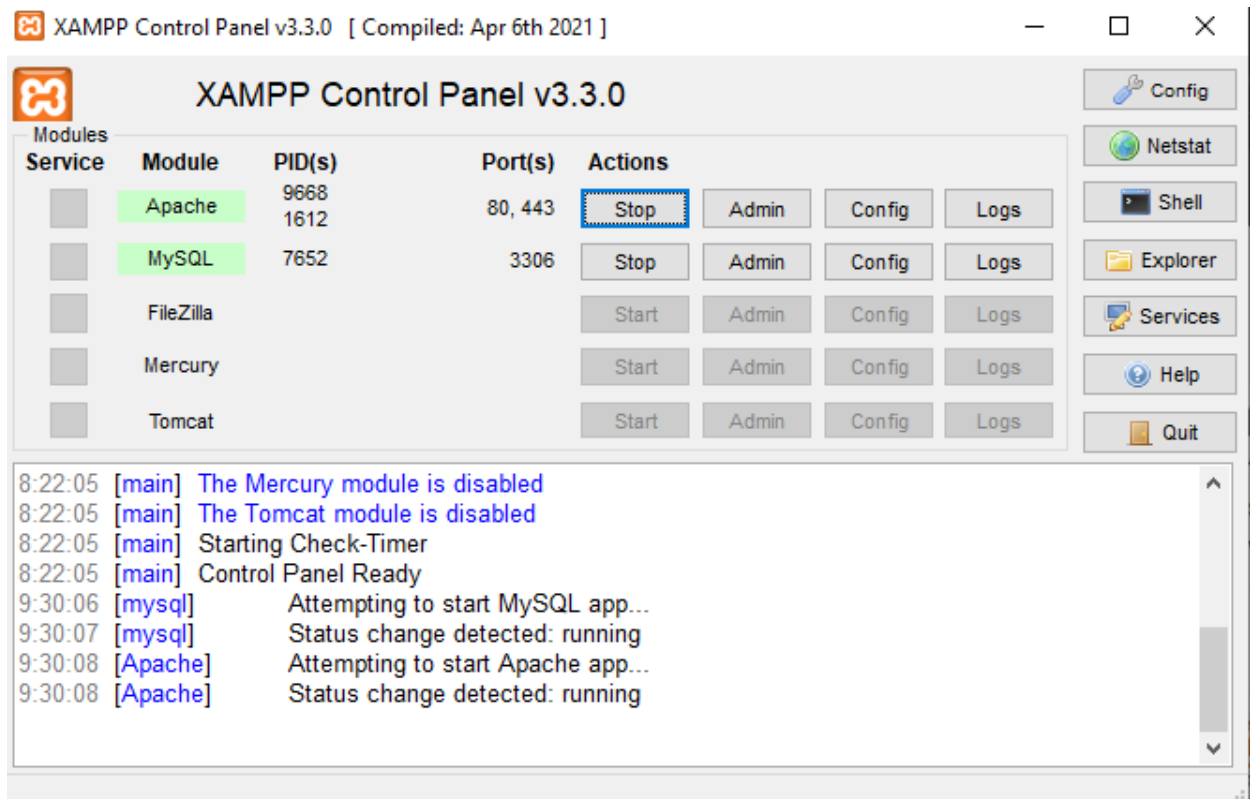
	Versiones compatibles	Particularidades
Sistema operativo	Windows 8.1 Windows 8.1 N	Microsoft Visual C++ Redistributable para Visual Studio 2015, 2017, 2019 y 2022: <ul style="list-style-type: none"> Versión x86 para un sistema operativo de 32 bits (necesaria para el control de origen SVN y GIT), Versión x64 para un sistema operativo de 64 bits (necesaria para el control de origen GIT).
	Windows 10 Windows 10 N	Microsoft Visual C++ Redistributable para Visual Studio 2015, 2017, 2019 y 2022: <ul style="list-style-type: none"> Versión x86 para un sistema operativo de 32 bits (necesaria para el control de origen SVN y GIT), Versión x64 para un sistema operativo de 64 bits (necesaria para el control de origen GIT).
	Windows 11	Microsoft Visual C++ Redistributable para Visual Studio 2015, 2017, 2019 y 2022, versión x64 (necesario para el control de origen GIT)
	Microsoft Azure Windows 10 Enterprise Multisesión	También funciona de forma conjunta con Azure Virtual Desktop (AVD). ¹ Admite máquinas Windows 365.
	Windows Server 2012 R2	Microsoft Visual C++ Redistributable para Visual Studio 2015, 2017, 2019 y 2022, versión x64 (necesario para el control de origen GIT) ²
	Windows Server 2016	Microsoft Visual C++ Redistributable para Visual Studio 2015, 2017, 2019 y 2022, versión x64 (necesario para el control de origen GIT) ²
	Windows Server 2019	Microsoft Visual C++ Redistributable para Visual Studio 2015, 2017, 2019 y 2022, versión x64 (necesario para el control de origen GIT) ²
	Windows Server 2022	Microsoft Visual C++ Redistributable para Visual Studio 2015, 2017, 2019 y 2022, versión x64 (necesario para el control de origen GIT) ²
Entornos de Citrix	XenApp v6.5 o posterior	
	XenDesktop v7.0 o posterior	
.NET Framework	Versión 4.6.1 o posterior	<p>.NET Framework es necesario para ejecutar proyectos de Windows: heredado y debe instalarse en máquinas.</p> <p>.NET (anteriormente .NET Core) es un marco más reciente necesario para ejecutar proyectos de Windows y multiplataforma. La versión .NET adecuada se añade automáticamente en la carpeta de instalación al instalar o actualizar Studio.</p> <p>Si tu máquina funciona con el sistema operativo Windows en un idioma distinto del inglés, instala el paquete de idioma correspondiente para la versión de .Net framework que estés utilizando.</p> <p>Es necesario para ejecutar Studio en cualquiera de los idiomas disponibles. Las versiones de .Net Framework y del paquete de idioma relacionado deben coincidir, y cualquier versión de .Net Framework en conflicto instalada en la máquina debe eliminarse.</p>

Navegadores web (para la automatización del navegador)	Microsoft Edge 78 o posterior con modo Internet Explorer (IE)	Internet Explorer como aplicación de escritorio no es compatible a partir del 15 de junio de 2022, siendo Internet Explorer 11 la última versión principal. Seguimos proporcionando soporte para Microsoft Edge con el modo Internet Explorer (IE), que ofrece soporte integrado con navegadores heredados para sitios que requieren Internet Explorer (consulta más detalles aquí). Automatizado con la extensión para Microsoft Edge .
	Google Chrome versión 64 o posterior	Automatizado con la extensión para Chrome o el protocolo Webdriver .
	Mozilla Firefox versión 52.0 o posterior	Automatizado con la extensión para Firefox o el protocolo Webdriver .
	Microsoft Edge en la versión 1803 de Windows 10 o posterior	Automatizado con el Protocolo de Webdriver .
	Microsoft Edge en modo IE (Internet Explorer v8.0 o superior) en la versión 1803 de Windows 10 o superior	Automatizado con la extensión para Microsoft Edge .
Microsoft Office (para proyectos creados en el perfil de StudioX)	Office: 2013	
	Office: 2016	
	Office: 2019	
	Office 365	
SistemaDeArchivos	NTFS	
IP	IPv4, IPv6	Studio se puede instalar en sistemas operativos compatibles con IPv6.

Instalación y configuración.

1. Instalar un servidor local.

Para poder desarrollar aplicaciones web usando PHP en un entorno local se necesitará un servidor web que soporte PHP, en nuestro caso se utiliza XAMPP V3.3.0 para poder tener el servidor activo tendremos que iniciar dos apartados, que sería Apache para poder tener el servidor y MySQL para poder usar la base de datos.

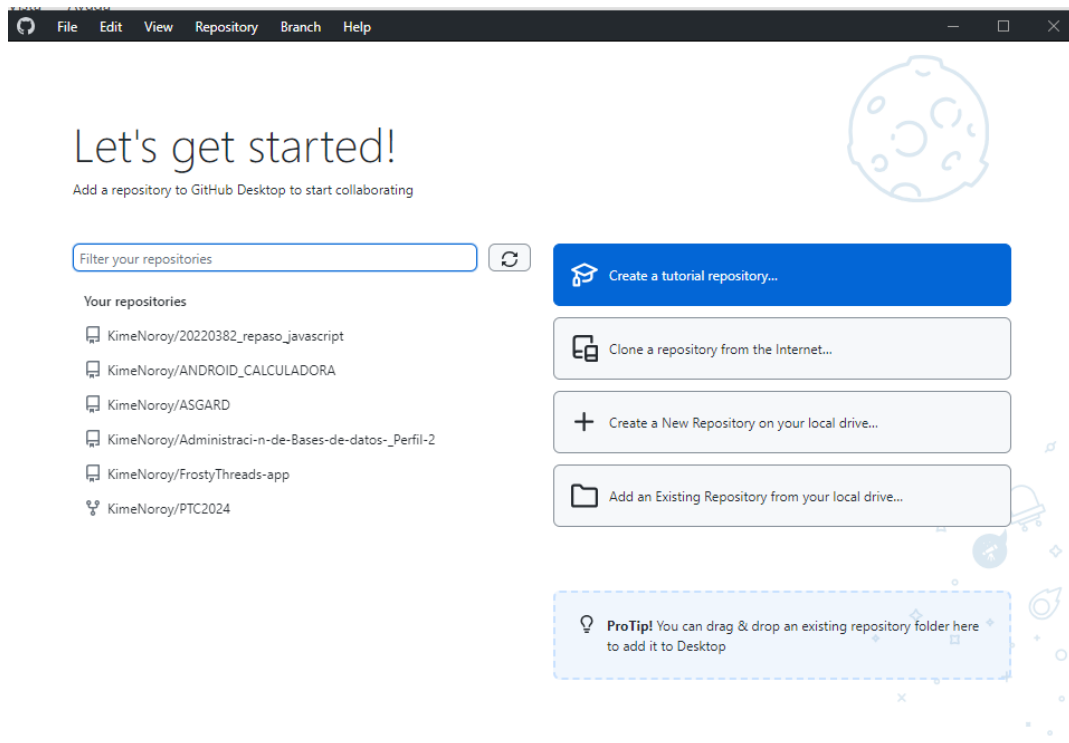


2. Configurar el entorno de trabajo.

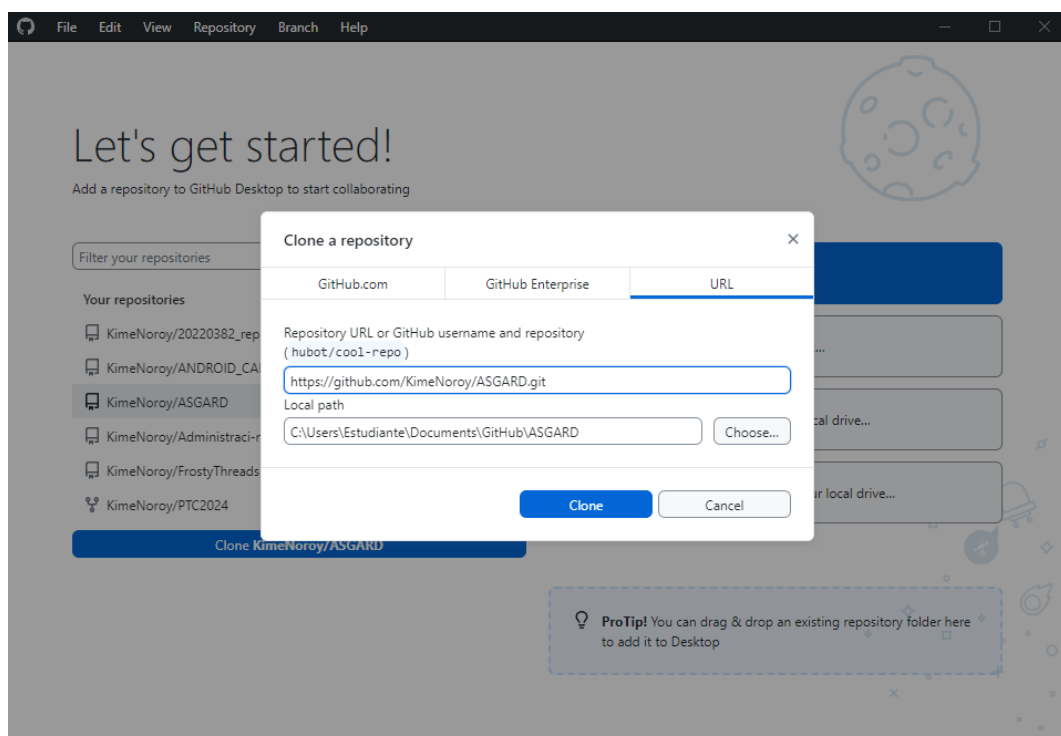
Iremos a la carpeta htdocs dentro del directorio donde instalamos XAMPP, Por ejemplo

C:\xampp\htdocs y dentro de htdocs crearemos, para poder poner una carpeta con nuestro proyecto tendremos que estar como colaborador en nuestro proyecto de GitHub luego clonaremos nuestro proyecto en la carpeta htdocs.

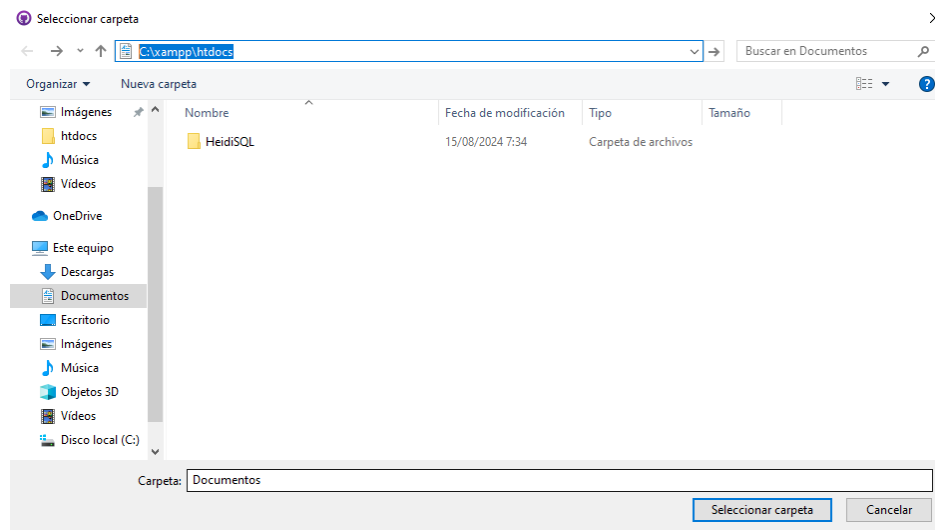
Abriremos “GitHub Desktop” (<https://desktop.github.com/download/>) si no lo tenemos lo instalamos y necesitamos iniciar sesión con nuestra cuenta de GitHub donde somos colaboradores con el proyecto.



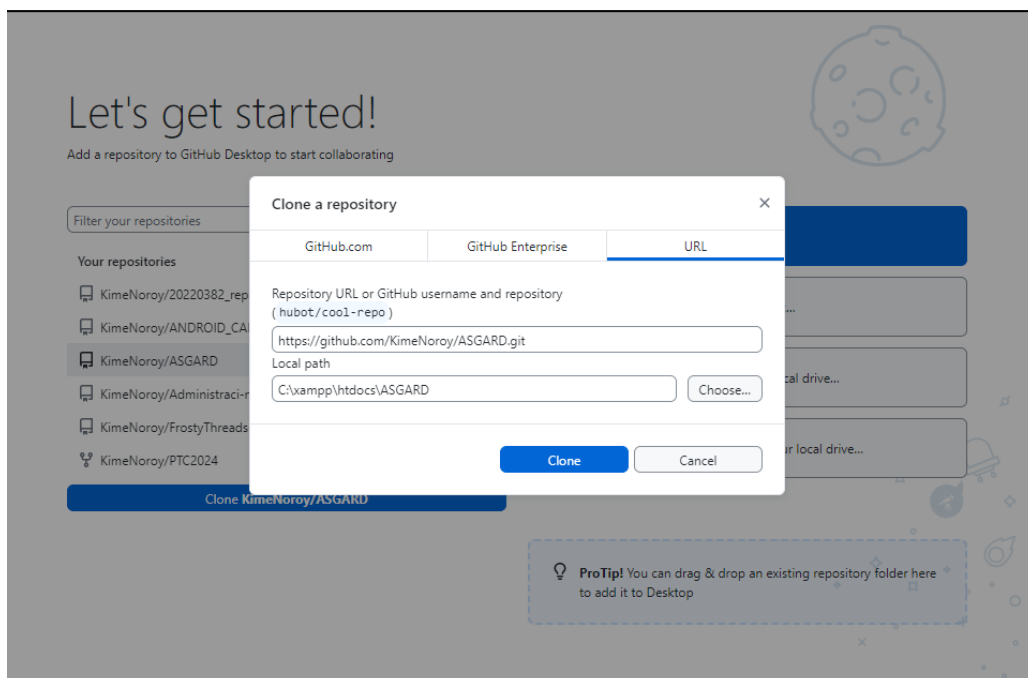
Luego elegiremos el nombre del proyecto que en este caso es “ASGARD” lo seleccionamos y se nos abrirá una nueva pestaña donde se nos mostrará el URL del proyecto y en que carpeta queremos guardarlo, le daremos al boton que se llama “Choose”



Se nos abrirá el explorador de archivos y en parte de arriba pondremos el directorio donde está la carpeta htdocs.

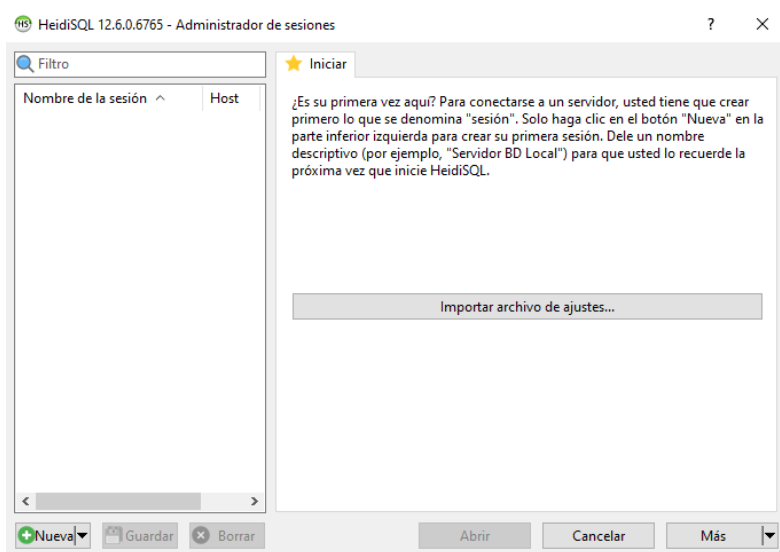


Luego de poner la URL de donde está la carpeta htdocs le damos al botón “seleccionar carpeta” y nos llevara de nuevo a GitHub Desktop y le damos al boton “Clone” y ya tendremos el proyecto en la carpeta htdocs para poder abrirlo con el servidor de Apache usando XAMPP

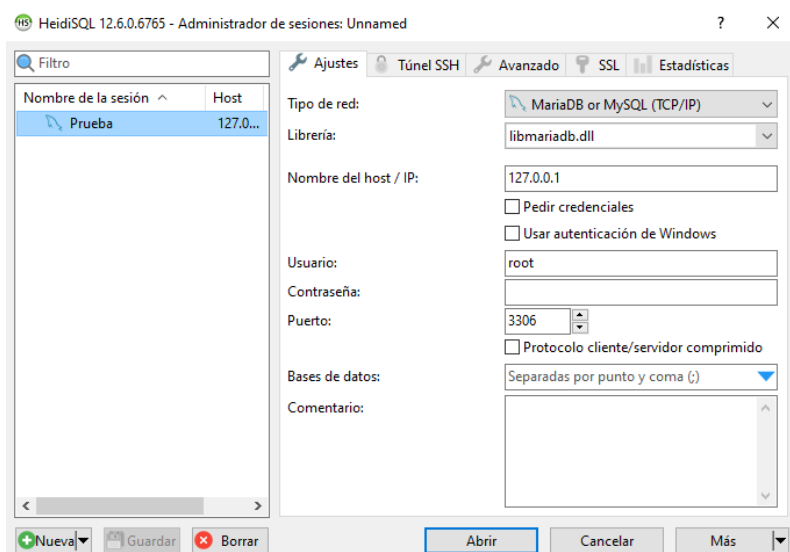


3. Tener alojada la base de datos.

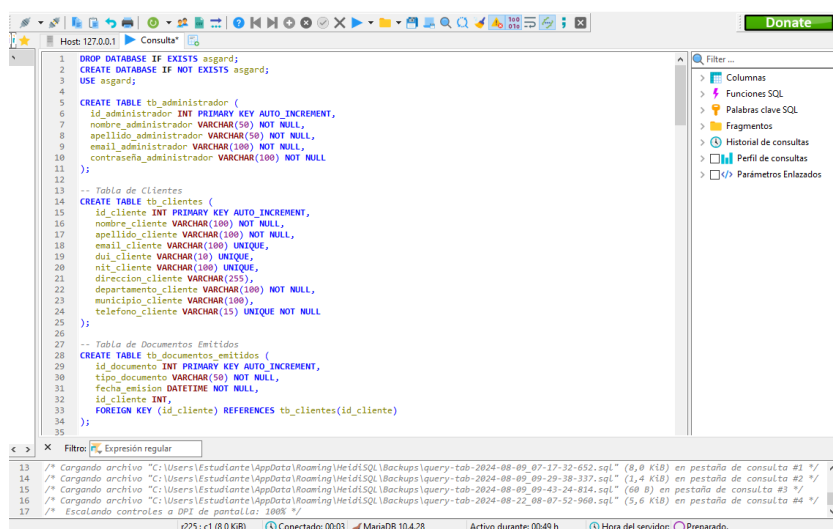
Para poder usar la base de datos deberemos tener el IDE HeidiSQL V12.8.0.6927 (<https://www.heidisql.com/download.php>) luego de instalarlo y abrirlo nos aparecerá esta ventana.



Le daremos en el botón que dice “nuevo” y nos aparecerá nuevas opciones para poder poner el nombre de la sesión donde estamos trabajando y diferentes opciones como el nombre de la ip, el usuario root, la contraseña del usuario, puerto etc, la dejaremos como viene por predeterminado y le daremos al botón que dice “Abrir”



Luego de darle al boton de “Abrir” se nos mostrara una pestaña que dice “Host 127.0.0.1” donde se nos mostrara todas las bases de datos que tengamos guardadas en nuestro equipo (Computadora) pero le daremos en la pestaña de al lado donde dice “Consulta” y en ese apartado pondremos nuestro script de la base de datos.



4. Acceder al proyecto:

Iremos a nuestro navegador favorito o donde tengas más confianza y con el XAMPP abierto y con el Apache y el MySQL encendidos iremos a la barra de búsqueda y pondremos el siguiente URL “localhost/ASGARD”




Index of /ASGARD

Name	Last modified	Size	Description
Parent Directory		-	
api/	2024-06-27 13:27	-	
asgard.sql	2024-06-27 13:27	4.5K	
controllers/	2024-06-27 13:27	-	
resources/	2024-06-27 13:27	-	
views/	2024-06-27 13:27	-	

Apache/2.4.56 (Win64) OpenSSL/1.1.1 It PHP/8.2.4 Server at localhost Port 80

Luego nos vamos a la carpeta que se llama “views” y seleccionaremos la carpeta que se llamada “admin”.

Index of /ASGARD/views

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 Parent Directory		-	
 admin/	2024-06-27 15:02	-	
 public/	2024-06-27 13:27	-	

Apache/2.4.56 (Win64) OpenSSL/1.1.1t PHP/8.2.4 Server at localhost Port 80

Y ya nos llevara a la página inicial “index” donde ya tendremos acceso a la aplicación web y si no hay ningun administrador registrado nos mandara al primer uso.

