**Proyecto P. Draw**

**Informe 3**

**Asignatura: Taller de Diseño y desarrollo de Soluciones**

**Sección:** D-IEI-N4-P1-C1

**Nombre del docente:** Geraldo Valenzuela Polanco

**Nombre de los integrantes del grupo:**

Vicente Sepúlveda

Brian Fierro

Carlos Salvadores

**Fecha de entrega**

**25-09-2024**

**Contenido**

[I. Introducción 3](#_Toc182265177)

[II. Definición del patrón de arquitectura 4](#_Toc182265178)

[III. Implementación del patrón de arquitectura 5](#_Toc182265179)

[IV. Estándares de programación segura 7](#_Toc182265180)

[V. Perfiles de Usuario 8](#_Toc182265181)

[VI. Distintos Servicios de almacenamiento cloud 9](#_Toc182265182)

[VII. Elección de servicio de almacenamiento cloud 11](#_Toc182265183)

[VIII. Identificación de criterios de aceptación 12](#_Toc182265184)

[IX. Identificar funcionalidades con solución a las historias de usuario 15](#_Toc182265185)

[X. Desarrollo del plan de pruebas 17](#_Toc182265186)

[XI. Hardware necesario para soportar la aplicación. 19](#_Toc182265187)

[XII. Software necesario para soportar la aplicación. 20](#_Toc182265188)

[XIII. Referencias bibliográficas 22](#_Toc182265189)

# Introducción

El presente informe aborda aspectos críticos del desarrollo de software, con un enfoque particular en la implementación de patrones de arquitectura y estándares de programación segura.

Destacamos los **criterios de aceptación del usuario** como pilar fundamental de nuestro análisis, centrándonos en cuatro áreas clave:

* **Rendimiento**: Optimización de la velocidad y eficiencia del sistema.
* **Funcionalidad**: Cumplimiento preciso de los requisitos especificados.
* **Usabilidad**: Interfaz intuitiva y experiencia de usuario fluida.
* **Seguridad**: Protección robusta de datos y prevención de vulnerabilidades.

Además, se analizan entre 6 y 10 **historias de usuario** principales, proporcionando una visión integral de las necesidades y expectativas del cliente. Este enfoque garantiza un desarrollo alineado con los objetivos del proyecto y las demandas del mercado actual.

El informe también explora servicios de almacenamiento en la nube, perfiles de usuario y metodologías de prueba, ofreciendo una perspectiva completa del ciclo de desarrollo de software moderno.

# Definición del patrón de arquitectura

Para el proyecto de la página de portafolio de dibujos haremos uso del modelo MTV (MODEL – TEMPLATE – VIEW el cual es el predeterminado al utilizar Django.

## División clara de funciones del sistema

El patrón MTV de Django separa claramente las responsabilidades: Modelo gestiona los datos (dibujos, precios, categorías), Témplate maneja el diseño y la presentación, y Vista controla la lógica de negocio y la interacción con los datos.

## Facilidad de mantenimiento y escalabilidad

La arquitectura MTV hace que sea más fácil mantener y hacer crecer un proyecto, ya que divide el código en partes separadas (Modelos, Vistas y Templates) donde pueden añadirse nuevas funciones, como cambiar estilos o actualizar el portafolio, esta estructura permite hacer cambios sin complicar el resto del código. Además, se pueden agregar o modificar funciones, como manejar categorías o cambiar precios, sin tener que reorganizar todo.

## Automatización del sistema de administración

Django genera automáticamente un panel de administración a partir de los modelos definidos, lo cual sería útil para que el administrador del sitio pueda agregar o eliminar dibujos, gestionar pedidos o cambiar estados de forma sencilla

## Facilidad en la creación de formularios

Django facilita la creación y gestión de formularios, lo que es importante en el proyecto, los formularios de cotización y contacto se pueden hacer y validar rápidamente gracias a los formularios integrados de Django, y además se conectan fácilmente con la vista y la plantilla para enviar correos.

# Implementación del patrón de arquitectura

Para implementar este patrón de arquitectura, será necesario incluir los siguientes componentes para asegurar el correcto desarrollo y construcción del proyecto.

## Template

En el template vamos a tener incorporar diferentes archivos HTML con sus respectivos archivos static, los archivos HTML que vamos a incluir son, página de presentación, página de portafolio, página de tipos y estilos de dibujos, página de precios, página de formulario de pago, página de formulario de pago sin iniciar sesión, página de login, página de registro, página de perfil de usuario, página de ver solicitudes, página de login (admin), página de opciones de dibujo, página de opciones de categoría, página de opciones de estilo, página de opciones de solicitudes. Además, va a tener dos archivos CSS en el static un CSS para la página vista por el usuario y otro CSS para la página vista por el administrador, con su respectivo archivo de imágenes y también un archivo de JS si es que es necesario.

## View

En las vistas vamos a tener que agregar estas dependiendo de las páginas que haya, con diferentes variaciones dependiendo de algunas validaciones o funciones que tengan incorporadas. En las vistas vamos tener que incorporar diferentes funciones dentro del archivo views.py las funciones que vamos a tener que incorporar son cargar dibujos, estilos y tipos en la página, calcular el precio del estilo y tipo, envió de formulario al administrador, envío de formulario hacía un correo, alerta de no haber creado cuenta, validación de datos suficientes en el formulario, validaciones de iniciar sesión(usuario y administrador), validación de datos faltantes en registro de usuario, validación de datos incongruentes en el registro, validación al ingresar correo en el perfil del usuario, validación de cambio de contraseña, cargar estado del dibujo, agregar imagen, eliminar imagen, agregar categoría, eliminar categoría, editar categoría, agregar estilo, eliminar estilo, editar estilo, cambiar estado de solicitud, ver solicitudes.

## Model

En los modelos vamos a definir cómo se estructuran los datos que se van a guardar en la base de datos y se usarán en las vistas y templates. Estos modelos estarán en el archivo models.py y manejarán las entidades clave como dibujos, estilos, tipos de dibujo, solicitudes y usuarios.

El modelo de Dibujo guardará cosas como el título, tipo (cabeza, torso, cuerpo completo), estilo, precio, imagen y descripción, y será crucial para mostrar los trabajos en las páginas de portafolio y estilos. Por su parte, los Estilos se gestionan a través de otro modelo, que organizará los diferentes estilos de arte, como realista o cartoon, tanto para la administración como para los usuarios que los vean.

Las solicitudes de cotización o compra se gestionan en un modelo que guardará detalles como el usuario que la hizo, el estado (en proceso, completado, rechazado), y otros datos necesarios para su administración. El modelo de Usuario se extenderá para manejar perfiles, historial de compras y solicitudes.

Por último, el sistema de login permitirá diferenciar el acceso entre usuarios comunes y administradores, asegurando que cada uno tenga los permisos correctos para sus funciones. Esto garantizará que cada usuario acceda solo a las opciones que le corresponden.

## Conexión entre los componentes

En este proyecto, los Modelos, Vistas y Templates trabajarán juntos para crear la aplicación web, donde los usuarios podrán ver los dibujos, estilos y hacer solicitudes, y los administradores podrán gestionar todo el contenido.

Modelos y Vistas: Los Modelos se encargan de definir cómo se guardan los datos, y las Vistas se conectan a estos para obtener la información necesaria y mandarla a las páginas. Por ejemplo, el modelo Dibujo se usará para obtener los dibujos que aparecerán en el portafolio, y el modelo de Solicitud para manejar las solicitudes que envían los usuarios. En las vistas, vamos a crear funciones en el archivo views.py que traen estos datos y los preparan, como cargar los dibujos o calcular precios según el tipo y estilo del dibujo.

Vistas y Templates: Cuando las Vistas ya tienen los datos de los Modelos, se los pasan a los Templates usando variables. Por ejemplo, si alguien entra al portafolio, la vista va a buscar los dibujos en el modelo y luego los mandará al Template de la galería para que los muestre en la página. Además, las vistas también se encargarán de procesar formularios (como cuando un usuario envía una cotización) y, dependiendo de si todo está bien o no, actualizará la página o mandarán una respuesta.

Templates y Modelos: Aunque los Templates son solo para mostrar información, dependen de los Modelos de forma indirecta a través de las vistas. Los templates usan los datos que les mandan las vistas para mostrar cosas dinámicas. Por ejemplo, un template de galería va a usar los datos del modelo de Dibujo para mostrar las imágenes, los precios y títulos. Los formularios en los templates también mandan datos, como en el login o en una cotización, que luego las vistas procesarán y almacenarán en los modelos correctos.

Este es el ciclo de cómo todo se conecta para que los datos se muestran, se procesen y se gestionen en el sistema.

# Estándares de programación segura

## Validación de entradas

Todas las entradas de los usuarios deben ser validadas y sanitizadas

Usar mecanismos como los formularios de Django, que ya incluyen validaciones, y asegurarse de que los datos sean del tipo y formato esperado.

## Manejo Seguro de Sesiones

Implementar medidas de seguridad para las sesiones del usuario, como usar HTTPS para proteger la información sensible y configurar un tiempo de caducidad para las sesiones inactivas.

## Control de Acceso y Autorización

Definir y aplicar un modelo de permisos basado en roles, asegurándose de que cada usuario solo tenga acceso a las funcionalidades necesarias para su rol.

## Manejo de Errores y Excepciones

Registrar los errores en un lugar seguro y mostrar mensajes genéricos al usuario.

## Cifrado y Protección de Datos Sensibles

Todos los datos sensibles deben estar cifrados

## Parcheo y Actualización del Software

Mantener todas las dependencias del proyecto, incluyendo las bibliotecas y el framework (Django), actualizadas para evitar vulnerabilidades.

## Revisión de Código (Code Review)

Realizar revisiones periódicas del código para detectar posibles vulnerabilidades o prácticas inseguras.

# Perfiles de Usuario

## Roles

### Desarrollador

Se encargará de la creación, mantenimiento en el proyecto además se encargará de realizar correctas prácticas de codificación para evitar vulnerabilidades.

### Tester

Probar la aplicación de múltiples maneras para asegurarse de que no tenga vulnerabilidades y realizará pruebas para descubrir fallos en la aplicación

### Usuario final

Uso de las aplicaciones y sistemas para cumplir con las tareas diarias

### Administrador

Permisos de lectura y escritura total dentro de la aplicación. (agregar, editar, eliminar elementos)

### Usuario sin inicio de sesión

Permisos mínimos dentro la aplicación, solo lectura.

### Usuario con inicio de sesión

Permisos limitados a ciertas funciones dentro de la aplicación, sus permisos solo son válidos al estar con la sesión iniciada.

Solo debe tener los permisos necesarios para realizar sus funciones básicas, las contraseñas deben cumplir la complejidad y caducidad

# Distintos Servicios de almacenamiento cloud

## Amazon Web Services (AWS)

Es una plataforma de pago con una opción gratuita limitada. Ofrece una amplia gama de servicios, como computación, almacenamiento, bases de datos, redes y herramientas de inteligencia artificial. Algunos de sus servicios más destacados son EC2, que proporciona servidores virtuales; S3, para almacenamiento de objetos; y Lambda, para computación sin servidor. AWS está disponible en numerosas regiones geográficas, lo que reduce la latencia. Ofrecen un plan gratuito durante 12 meses con recursos limitados, ideal para desarrolladores y pequeñas empresas que quieren probar sus servicios.

## Microsoft Azure

Es un servicio de pago con un nivel gratuito limitado. Es conocido por su integración profunda con productos de Microsoft, como Office 365 y Dynamics, lo que lo hace especialmente útil para empresas que ya utilizan el ecosistema Microsoft. Azure proporciona servicios de computación, almacenamiento, redes y bases de datos. Algunas de sus herramientas más conocidas incluyen Azure Functions, para la computación sin servidor, y Azure Virtual Machines, que permite desplegar servidores virtuales. Al igual que AWS, Azure ofrece un plan gratuito que incluye créditos y servicios limitados por 12 meses.

## Google Cloud Platform (GCP)

Es otra opción de pago con niveles gratuitos limitados. Ofrece servicios como Compute Engine para servidores virtuales, Google Kubernetes Engine (GKE) para la gestión de contenedores y Big Query para análisis de datos a gran escala. GCP también cuenta con App Engine, una plataforma sin servidor para desarrollar y desplegar aplicaciones. Google Cloud ofrece 12 meses de prueba gratuita con un crédito inicial de $300, ideal para proyectos de desarrollo y pruebas de concepto.

## IBM Cloud

Es una plataforma de pago que también ofrece servicios gratuitos limitados. Uno de sus grandes atractivos es la integración de su inteligencia artificial Watson, que proporciona potentes capacidades de análisis de datos. IBM Cloud ofrece servicios en áreas como blockchain, Internet de las cosas (IoT) y computación cuántica. También soporta Kubernetes y contenedores. Su plan gratuito incluye servicios básicos y un crédito limitado, lo que facilita el acceso inicial a sus potentes herramientas.

## Oracle Cloud

Es otro servicio de pago que ofrece opciones gratuitas limitadas. Es conocido por su Oracle Autonomous Database, que proporciona una base de datos automatizada de alto rendimiento. Oracle Cloud Infrastructure también ofrece servicios de red y computación robustos. Su plan gratuito incluye servidores virtuales de baja capacidad y bases de datos, lo que lo convierte en una excelente opción para entornos de desarrollo y pruebas.

## Heroku

Es una plataforma de nube que ofrece servicios gratuitos y de pago. Es popular entre desarrolladores por su facilidad de uso para desplegar aplicaciones. Heroku permite la integración con GitHub para facilitar el despliegue continuo de aplicaciones. El plan gratuito tiene limitaciones, como el "modo de sueño" cuando la aplicación está inactiva, mientras que los planes de pago proporcionan más recursos y capacidades adicionales.

# Elección de servicio de almacenamiento cloud

El servicio de almacenamiento cloud que más se adecua a las necesidades del cliente es Google Cloud Storage (GSP), ya que ofrece planes más accesibles al inicio, comenzando con una capa gratuita que ofrece 5gb de almacenamiento gratuito indefinido, después de superar los 5 Gb se tiene que ir pagando el resto de almacenamiento que se vaya a utilizar, para ser un portafolio de dibujos es suficiente para ir empezando. A continuación, se van a mostrar las siguientes ventajas que tiene GSP en nuestro proyecto.

## Integración con Django

GCP tiene herramientas que se integran bien con Django, lo que te permite almacenar imágenes, archivos y otros datos de manera segura. Con la librería Django-storages.

## Espacio suficiente en el plan gratuito

El plan gratuito de GCP te da 5 GB de almacenamiento indefinido.

## Escalabilidad

Si es que el proyecto crece, no se necesita cambiar de servicio. GCP deja ampliar el almacenamiento y la capacidad a medida que se necesite, y solo se paga por lo que se usa cuando se supere el plan gratuito. Es bastante flexible en ese sentido.

## Seguridad y permisos avanzados

GCP permite tener un control súper detallado sobre quién puede ver o manejar los archivos. Es genial para el proyecto porque se pueden definir diferentes niveles de acceso para los administradores y los usuarios, y además se pueden configurar reglas de seguridad para que los datos siempre estén protegidos.

## Acceso global y fiabilidad

Si es que el proyecto empieza a escalar a más personas del mundo. Tiene centros de datos en todo el mundo, lo que asegura que los usuarios podrán acceder al contenido rápido, sin importar de dónde sean.

# Identificación de criterios de aceptación

**HU 01 - Como administrador, quiero tener control del contenido que subo y poder eliminarlo.**

**Criterios de aceptación:**

El administrador puede subir imágenes/dibujos al sistema.

El administrador puede eliminar un dibujo existente.

Después de eliminar un dibujo, este desaparece del portafolio del artista.

**Restricción**: El formato de los archivos que el administrador puede subir está limitado a imágenes JPEG, PNG, GIF.

**Restricción**: Los dibujos eliminados no pueden ser recuperados a menos que haya un sistema de respaldo.

**HU 02 - Como administrador, quiero visualizar los pedidos de la gente para poder ver cuántos tengo en cola.**

**Criterios de aceptación:**

El administrador puede ver una lista de pedidos.

Los pedidos se muestran con su estado (en cola, en proceso, completado, rechazado).

El administrador puede acceder al detalle de cada pedido.

**Restricción**: El cambio de estado solo se puede hacer si los datos del pedido están completos (nombre del cliente, tipo de dibujo, fecha de solicitud).

**Restricción**: El estado no puede revertirse a "En proceso" si ya ha sido marcado como "Completado" o "Rechazado".

**HU 03 - Como administrador, quiero controlar el estado de los pedidos para gestionar su proceso.**

**Criterios de aceptación:**

El administrador puede cambiar el estado de un pedido (en proceso, completado, rechazado).

Los cambios en el estado se reflejan instantáneamente en la interfaz de usuario.

El cliente recibe una notificación cuando cambia el estado de su pedido.

**Restricción**: Los estilos deben tener un nombre único (no se pueden repetir nombres).

**Restricción**: El administrador solo puede eliminar estilos que no estén asignados a ningún dibujo.

**HU 04 - Como administrador, quiero gestionar mis estilos para poder asignarlos a mis dibujos.**

**Criterios de aceptación:**

El administrador puede agregar nuevos estilos de dibujo.

El administrador puede editar y eliminar estilos existentes.

Los estilos pueden ser asignados a los dibujos en el sistema.

**Restricción**: Los usuarios deben estar registrados e iniciar sesión para ver el portafolio completo.

**Restricción**: El portafolio no mostrará dibujos en proceso, solo aquellos que ya han sido completados y aprobados por el administrador.

**HU05 - Como administrador, quiero gestionar mis tipos de dibujo para asignarlos correctamente.**

**Criterios de aceptación:**

El administrador puede agregar, editar y eliminar tipos de dibujo.

Los tipos de dibujo se muestran correctamente en las opciones de los clientes al solicitar un trabajo.

**HU06 - Como usuario, quiero ver los dibujos del artista para poder ver sus trabajos.**

**Criterios de aceptación:**

El usuario puede ver todos los dibujos en una galería.

Cada dibujo se muestra con su estilo y tipo.

El usuario puede hacer clic en un dibujo para ver más detalles.

**Restricción**: Los usuarios deben estar registrados e iniciar sesión para ver el portafolio completo.

**Restricción**: El portafolio no mostrará dibujos en proceso, solo aquellos que ya han sido completados y aprobados por el administrador.

**HU 07 - Como usuario, quiero ver los estilos del artista para conocer todas las opciones disponibles.**

**Criterios de aceptación:**

El usuario puede ver una lista o galería con todos los estilos de dibujo disponibles.

Los estilos incluyen una breve descripción y ejemplos de cada uno.

**HU 08 - Como usuario, quiero solicitar un dibujo al artista para comprar un trabajo.**

**Criterios de aceptación:**

El usuario puede seleccionar un tipo de dibujo y estilo.

El formulario de solicitud se envía correctamente con la información del usuario.

El usuario recibe una confirmación de la solicitud enviada.

**Restricción**: El formulario de solicitud no se puede enviar si faltan datos obligatorios como el tipo de dibujo y el estilo.

**Restricción**: Los usuarios solo pueden hacer una solicitud a la vez, no pueden enviar múltiples solicitudes en simultáneo.

**HU 09 - Como usuario, quiero ver las solicitudes que haya hecho para hacer un seguimiento de mis pedidos.**

**Criterios de aceptación:**

El usuario puede ver una lista de sus solicitudes.

Cada solicitud muestra el estado actual (en proceso, completado, rechazado).

El usuario puede ver los detalles de cada solicitud.

**Restricción**: Solo se pueden ver las solicitudes de los últimos 6 meses.

**Restricción**: Los usuarios no pueden modificar una solicitud una vez que el pedido ha sido marcado como "En proceso".

# Identificar funcionalidades con solución a las historias de usuario

**HU01: Control del contenido subido y eliminado**

Funcionalidad 1: Subida de imágenes/dibujos por el administrador.

Funcionalidad 2: Eliminación de dibujos existentes por el administrador.

Funcionalidad 3: Actualización del portafolio del artista tras la eliminación de un dibujo.

**HU02: Visualización de pedidos en cola**

Funcionalidad 4: Vista de la lista de pedidos con su estado (en cola, en proceso, completado, rechazado).

Funcionalidad 5: Acceso al detalle de cada pedido por el administrador.

**HU03: Gestión del estado de pedidos**

Funcionalidad 6: Actualización del estado del pedido (en proceso, completado, rechazado).

Funcionalidad 7: Reflejo instantáneo de los cambios de estado en la interfaz.

Funcionalidad 8: Envío de notificación al cliente cuando cambia el estado de su pedido.

**HU04: Gestión de estilos de dibujo**

Funcionalidad 9: Creación, edición y eliminación de estilos de dibujo por el administrador.

Funcionalidad 10: Asignación de estilos a los dibujos en el sistema.

**HU05: Gestión de tipos de dibujo**

Funcionalidad 11: Creación, edición y eliminación de tipos de dibujo.

Funcionalidad 12: Visualización correcta de tipos de dibujo en las opciones de solicitud.

**HU06: Visualización de la galería de dibujos**

Funcionalidad 13: Visualización de la galería de dibujos por los usuarios.

Funcionalidad 14: Muestra de estilo y tipo de cada dibujo en la galería.

Funcionalidad 15: Opción para que el usuario vea los detalles de un dibujo.

**HU 07: Visualización de estilos de dibujo**

Funcionalidad 16: Lista o galería de estilos de dibujo disponibles para los usuarios.

Funcionalidad 17: Muestra de descripciones breves y ejemplos de cada estilo.

**HU 08: Solicitud de dibujos**

Funcionalidad 18: Selección de estilo y tipo de dibujo por parte del usuario.

Funcionalidad 19: Envío correcto del formulario de solicitud con los datos del usuario.

Funcionalidad 20: Confirmación de la solicitud enviada al usuario.

**HU 09: Seguimiento de solicitudes**

Funcionalidad 21: Lista de solicitudes del usuario.

Funcionalidad 22: Visualización del estado actual de cada solicitud (en proceso, completado, rechazado).

Funcionalidad 23: Acceso a los detalles de cada solicitud por parte del usuario.

# Desarrollo del plan de pruebas

## Pruebas funcionales

**Pruebas de unidad**

probar cada funcionalidad individualmente, como la subida, eliminación de dibujos, gestión de pedidos, etc. Ejemplo: verificar que un administrador puede eliminar un dibujo y que este desaparece del portafolio.

**Pruebas de integración**

Se prueba cómo interactúan los diferentes módulos del sistema. Ejemplo: comprobar que, al cambiar el estado de un pedido, se refleja correctamente en la interfaz del cliente.

**Pruebas de regresión**

Garantizan que nuevas funciones no rompan las funcionalidades ya existentes. Ejemplo: tras agregar la función de gestión de estilos, verificar que la visualización de pedidos sigue funcionando correctamente.

**Pruebas de aceptación de usuario**

Validar que el sistema cumple con los criterios de aceptación definidos en las historias de usuario.

## Pruebas no funcionales

**Pruebas de usabilidad**

Evaluar la facilidad de uso tanto para los administradores como para los usuarios. Ejemplo: verificar que los usuarios puedan navegar fácilmente por la galería y enviar solicitudes sin confusión.

## Pruebas de interfaz

**Pruebas de interfaz**

Verificar que los elementos visuales, como botones, formularios, y listas desplegables, funcionan correctamente Ejemplo: asegurarse de que los estilos de dibujo se muestran correctamente en la galería.

## Pruebas de Sistema

**Pruebas de extremo a extremo**

Verificar que todo el flujo funciona correctamente. Ejemplo: un usuario ve un dibujo, selecciona un estilo y tipo, envía una solicitud, y el administrador gestiona el pedido hasta su finalización.

**Pruebas de compatibilidad**

Aseguran que el sistema funciona correctamente en diferentes navegadores y dispositivos. Ejemplo: verificar la funcionalidad en navegadores como Chrome, Firefox, y en dispositivos móviles.

# Hardware necesario para soportar la aplicación.

## Computador (De escritorio, portátiles, netbooks y ultrabooks):

* **Manejo de navegador con HTML5**

## Almacenamiento (Disco duro HDD o SSD, pendrive, etcétera):

* + **Estándar: 8GB a más**

## RAM:

* 1GB a más

# Software necesario para soportar la aplicación.

## Sistema operativo

* + Windows 7 a Windows 11: Windows 11 es el nombre de la versión más reciente del sistema operativo Windows desarrollado por Microsoft, además de ser el sucesor de la anterior versión del sistema operativo, Windows 10, lanzado en 2015.

## Navegadores

* + Navegador que soporte HTML5 (Chrome, Firefox, Opera, Brave, etcétera.)

## Lenguaje de Programación y Frameworks

* + Python: Python es un lenguaje de programación de alto nivel que te permite trabajar con más eficiencia y efectividad.
  + Django: Django es un framework de Python de alto nivel que fomenta el desarrollo rápido y el diseño limpio y pragmático.

## Base de datos

* + MySQL: MySQL es una base de datos de código abierto, relacional y de alto rendimiento, compatible con muchos lenguajes de programación y aplicaciones.

## Otros programas

* + XAMPP: **XAMPP** es un paquete de software libre, que consiste principalmente en el sistema de gestión de bases de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script PHP y Perl.

## Instrucciones

* + Tener instalado:
    - Visual Studio Code: [Descargar Visual Studio Code](https://code.visualstudio.com/Download)
    - Python: [Descargar python](https://www.python.org/ftp/python/3.13.0/python-3.13.0-amd64.exe)
    - MySQL Workbench: [Descargar MySQL Workbench](https://dev.mysql.com/downloads/workbench/)
    - XAMPP: [Descargar XAMPP](https://sourceforge.net/projects/xampp/files/XAMPP%20Windows/8.2.12/xampp-windows-x64-8.2.12-0-VS16-installer.exe)
    - En XAMPP instalar el Apache y MySQL
  + Tener en cuenta que la base de datos se crea con MySQL Workbench
  + Ejecutar CMD de Windows y ejecutar el comando “cd:” dos veces y localizar la ubicación de la aplicación MySQL Workbench y con esto logramos abrir la ubicación del archivo, pero si nos encontramos con el acceso directo, debemos pulsar clic derecho y “Abrir la ubicación del archivo”, ahora copiar la ruta, ir a la terminal, copiar “cd <Ruta de MySQL Workbench>”
  + Escribimos “MySQL -u root” y ejecutamos después “create database portafolio\_db” con esto creamos la base de datos.
  + Ejecutar Apache con MySQL en la aplicación XAMPP
  + Ejecutar comandos de instalación en la terminal de Visual Studio Code: pip install Django==5.0, pip install pillow, pip install mysql-client.
  + Abrimos el Proyecto con Visual Studio Code, eliminamos la carpeta *pycache y 0001\_initial.py*
  + abrimos al terminal de visual studio code y debemos estar dentro de p\_draw y ejecutar el comando “python manage.py makemigrations” después ejecutar el comando “migrate”
  + para ejecutar escribir python manage.py runserver

## Futuras Consideraciones

* La pagina web debería de tener un enlace (Dominio) y alojada a una nube.
* Tener la base de datos alojada en una nube.

# Referencias bibliográficas

## Sitio web

* + *Desarrollolibre:* <https://www.desarrollolibre.net/blog/python/el-patron-de-diseno-mtv-en-django-y-su-relacion-con-el-mvc>
  + *Espifreelancer:* <https://espifreelancer.com/mtv-django.html>
  + *Oracle:* <https://www.oracle.com/cloud/free/>
  + *IBM:* <https://www.ibm.com/cloud/free>
  + *Heroku:* <https://www.heroku.com/pricing>
  + *Azure.microsoft:*<https://azure.microsoft.com/en-us/pricing/purchase-options/azure-account?icid=azurefreeaccount>
  + *Aws.amazon:* <https://aws.amazon.com/free/free-tier-faqs/>

[https://aws.amazon.com/free/?all-free-tier.sort-by=item.additionalFields.SortRank&all-free-tier.sort-order=asc&awsf.Free%20Tier%20Types=\*all&awsf.Free%20Tier%20Categories=\*all](https://aws.amazon.com/free/?all-free-tier.sort-by=item.additionalFields.SortRank&all-free-tier.sort-order=asc&awsf.Free%20Tier%20Types=*all&awsf.Free%20Tier%20Categories=*all)

* + *Cloud.google:* <https://cloud.google.com/docs?hl=es-419>

<https://cloud.google.com/free?hl=es_419>

* + *Loadview-testing:*[*https://www.loadview-testing.com/es/blog/tipos-de-pruebas-de-software-diferencias-y-ejemplos/*](https://www.loadview-testing.com/es/blog/tipos-de-pruebas-de-software-diferencias-y-ejemplos/)
  + *Owasp:*<https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/stable-es/02-checklist/>