* PROYECTO FIN DE GRADO - Titulo Provisional

Guillermo Muñoz Serrano

Jose Daniel Tello

## ÍNDICE

# Introducción

En la actualidad, las empresas están sufriendo una transformación en todos sus ámbitos. Las nuevas tecnologías suponen la entrada a un paradigma digital donde todos los servicios deben estar interconectados y funcionales. Las empresas deben ser capaces de operar y relacionarse con sus clientes de manera que puedan adaptarse a los nuevos cambios y retos acordes con los estándares tecnológicos actuales. La digitalización y la implementación de soluciones de IT avanzadas no solo mejoran la eficiencia operativa, sino que también ofrecen nuevas oportunidades para interactuar con los consumidores, construir una imagen de marca sólida y desarrollar estrategias de marketing efectivas que hagan que una empresa destaque frente al resto, o, por el contrario, que desaparezca del mercado. En este contexto, una empresa debe enfocar grandes esfuerzos en la creación de una página web bien diseñada y funcional. Una buena página web se ha convertido en un elemento esencial para cualquier empresa que aspire a destacar en un mercado cada vez más competitivo y tecnológicamente más desarrollado.

Las páginas web modernas son mucho más que simples escaparates estáticos; son plataformas dinámicas que permiten una interacción cliente-empresa mucho más efectiva. Las empresas ofrecen servicios de todo tipo a través de sus propias páginas web, desde la simple y tradicional presentación de información hasta la posibilidad de realizar transacciones, gestionar reservas, ofrecer soporte en tiempo real y mucho más.

Un servicio de gran utilidad es la posibilidad de integrar en una página web empresarial un sistema de fidelización por puntos. Este tipo de sistemas es extremadamente eficaz para fomentar la lealtad del cliente, aumentar la retención y, en última instancia, incrementar los ingresos. En el sector de la restauración, un sistema de puntos como él que hemos creado para este proyecto puede ser especialmente llamativo, ya que permite a los restaurantes crear una relación más cercana y personalizada con sus clientes.

## ¿Qué es lo que pretendemos hacer?

Nuestro objetivo en este proyecto es la creación de un sistema de puntos de fidelización enfocado al sector de la restauración. Montaremos una página web conectada a una base de datos simulando el sistema que podría tener un restaurante. Esta página web será capaz de leer códigos QR generados por un programa parecido al que pudiera haber en un establecimiento hostelero. Los clientes podrán acumular puntos mediante el empleo del lector de QR de la web; dichos QR llevarán la información contenida de la transacción a la base de datos. Posteriormente la propia base de datos tendrá un procedimiento de suma de puntos por usuario. Nuestra intención es ofrecer el esqueleto de un servicio sencillo de implementar y que permita una gran adaptabilidad para las necesidades de cada restaurante.

A través de este sistema, los clientes pueden acumular puntos con cada compra, los cuales pueden canjear por descuentos, promociones especiales u otros beneficios. Cualquier restaurante podrá implementar un sistema basado en lo que proponemos y que sea completamente operativo y modular.

## ¿Por qué lo que quiero hacer es importante? ¿Qué necesidad hay en el mercado de mi producto?

La importancia de este proyecto radica en darle la facilidad y las herramientas a pequeños establecimientos para que tengan sus propios sistemas por puntos en sus locales sin una gran inversión económica ni de tiempo.

Hay una clara tendencia en el mercado enfocada a que los establecimientos tengan una relación más estrecha con sus consumidores a través de las tecnologías. Los consumidores valoran cada vez más las experiencias personalizadas, y un sistema de puntos que se integra con una cuenta de usuario en línea permite al restaurante ofrecer promociones y recompensas adaptadas a los hábitos y preferencias de cada cliente. Esta personalización no solo mejora la satisfacción del cliente, sino que también puede traducirse en un aumento de las ventas y la lealtad del cliente a largo plazo.

Al crear un proyecto sencillo como este permitimos a los locales más pequeños tener un sistema de puntos a través de una web como lo pueden tener grandes empresas de restauración que lideran el mercado actual. Además, mediante el empleo de QR su uso para los usuarios es sencillo de entender al ser una tecnología con lo que están familiarizados.

## Objetivos

**1.** **Desarrollar y Configurar la Base de Datos:**

• Crear una base de datos robusta que almacene información de usuarios, pedidos, facturas y transacciones.

1. **Diseñar y Desarrollar una Página Web Interactiva con Django:**

• Crear una página web funcional y de fácil uso utilizando el framework Django.

• Incluir funcionalidades de autenticación de usuarios, permitiendo a los clientes registrarse, iniciar sesión y gestionar sus cuentas.

**3.** **Integrar la Página Web con la Base de Datos:**

• Implementar una API en la página web que permita la interacción con el sistema de puntos leyendo el QR.

• La página web debe poder enviar la información del QR a la base de datos

**4. Desarrollar un Programa para la Terminal del Restaurante:**

• Creación y gestión de las cuentas de clientes. El programa debe ser capaz de generar los tickets con las facturas.

• El programa permitirá a los camareros tomar pedidos, generar cuentas y crear códigos QR.

• El programa debe conectarse con la base de datos para leer el menú disponible.

**5.** **Implementar el Sistema de Puntos de Fidelización:**

• Desarrollar un sistema de puntos que incentive a los clientes a regresar al restaurante, acumulando puntos con cada compra.

• Integrar el sistema de puntos con la API de la página web para que los clientes puedan ver y gestionar sus puntos acumulados.

**6.** **Implementar el Sistema en Docker:**

• Desarrollar los contenedores para que tenga nuestra base de datos, el framework Django para poder hacer pruebas en local y un servidor web Nginx para lanzar el proyecto

# Planificación

La planificación de nuestro proyecto se llevó a cabo en colaboración y con buena fe entre las dos partes. Las dos primeras semanas las dedicamos exclusivamente a la conceptualización de la idea y desarrollar cuales son los puntos clave que necesitaríamos, así como pensar por encima que tecnologías deberíamos implementar. En este proceso discutimos cuáles serían los objetivos del proyecto y trazamos un plan general de funcionamiento y desarrollo para una mayor fluidez a la hora de trabajar en equipo.

A partir de la segunda semana comenzamos a dividirnos las tareas siguiendo unos primeros objetivos que resumimos y dejamos por escrito. La división de trabajo se realizó de manera estratégica que cada uno podía elegir por qué parte del proyecto empezar, de esta forma cada uno escogió tareas que aprovechaban sus fortalezas y habilidades.

Tras una semana realizando las tareas que teníamos programadas y estando en comunicación para informar de cualquier necesidad o problema que nos encontramos, nos reuníamos al finalizar la misma para una reunión de control y ver avances. Cada uno mostraba en qué punto se encontraba con respecto a sus tareas asignadas y discutíamos cualquier obstáculo que pudiera haber surgido durante estas reuniones.

Estas reuniones semanales también nos permitían discernir cuáles serían las tareas de la siguiente semana incluyendo nuevas tareas que podían surgir durante el desarrollo del proyecto, asegurando que ambos estuviéramos alineados con los objetivos del proyecto y que el trabajo se realizara de manera coherente. Esta metodología de planificación y revisión semanal nos permitió resolver problemas rápidamente, así como mantener un ritmo de trabajo aceptable.

Las últimas dos semanas las dedicamos a juntar nuestras partes del proyecto e integrarlas completamente para asegurarnos que todo funcionaba correctamente. Estas dos semanas finales también las utilizamos para la escritura y revisión del proyecto.

* 1. Tabla de planificación

En la siguiente tabla exponemos brevemente y de forma esquemática como fue la planificación de nuestro proyecto por semanas.

|  |  |  |
| --- | --- | --- |
| SEMANA | JOSE DANIEL | GUILLERMO |
| 1 | Desarrollo idea proyecto | Desarrollo idea proyecto |
| 2 | Planificación del desarrollo,  Investigación funcionalidad framework Django | Planificación del desarrollo,  Investigación de generador de QR |
| 3 | Diseñar y Desarrollar una Página Web Interactiva con Django | Diseño y creación de la base de datos |
| 4 | Investigación e implementación de API para lector de QR | Desarrollar un Programa para la Terminal del Restaurante con generador de QR |
| 5 | Desarrollo de las Páginas Web incluyendo autenticación de usuarios y conectividad con la base de datos | Investigación de implementación de contenedores Docker |
| 6 | Puesta a punto del proyecto de Django | Dockerizar programas en contenedores: MySQL, DJANGO, NGINX |
| 7 | Documentación y resolución de errores | Documentación y resolución de errores |
| 8 | Documentación y resolución de errores | Documentación y resolución de errores |

# Descripción Informática

En este apartado vamos a explicar de forma detallada en que consiste, como es el diseño del proyecto y que requerimientos hacen falta para poder llevarlo a cabo. Para mostrar de forma más clara las necesidades de nuestro proyecto, hemos dividido el total en dos partes separadas: el servidor y el restaurante.

Estas dos partes diferenciadas se deben a que se van a implementar tecnologías y herramientas diferentes en el restaurante y el servidor, los cuáles se encuentran en espacios físicos separados.

* 1. Análisis de requisitos
* Restaurante:

Requisitos hardware:

* Una terminal con acceso a internet. Puede ser un ordenador personal convencional, algún dispositivo tipo Tablet o cualquier dispositivo capaz de correr un programa.
* Una impresora o un dispositivo para imprimir las cuentas en el restaurante. Esto será necesario para poder enseñar de forma física al cliente el QR necesario para que el cliente pueda escanearlo. También es posible mostrar el QR desde un dispositivo móvil o una Tablet para no tener que imprimir en papel.

En nuestro caso estamos utilizando nuestro ordenador personal para simular la terminal del restaurante.

Requisitos Software:

* Es necesario descargar las dependencias adecuadas para poder utilizar Python en la terminal.
* Instalación de conector Python con MYSQL.

En nuestro caso estamos utilizando una máquina virtual con Ubuntu 22.04.3 LTS. La versión de Python corresponde con Python 3.11.7. Nuestro conector Python-MySQL corresponde con mysql-connector-python versión 8.3.0.

* Servidor

Requisitos Hardware:

* Este apartado depende de las necesidades del negocio, en caso de un local grande que necesite una gestión importante sería necesario tener un servidor donde alojar los programas que dictaremos más adelante. En caso de no tener un volumen de negocio muy grande es posible crear un pequeño servidor por tu cuenta o incluso utilizar un ordenador. Existe la opción de no necesitar un servidor físico y utilizar los servicios de Cloud para hospedar nuestros programas.

En nuestro caso, al ser un entorno de pruebas estamos utilizando nuestros ordenadores personales.

Requisitos Software:

* Instalación de las dependencias de Docker y Docker-Compose
* Sistema operativo capaz de ejecutar los contenedores Docker. (Para una mejor optimización podemos utilizar un sistema operativo de tipo Server).

En nuestro caso estamos utilizando un sistema operativo Ubuntu Desktop al estar en un entorno de pruebas.

* 1. Requisitos funcionales específicos

Requisitos de Hardware

* Procesador: CPU de 64 bits con soporte para virtualización.
* Memoria RAM: 2 GB mínimo (4 GB o más recomendado).
* Almacenamiento: Al menos 10 GB de espacio libre en disco.
* Conexión a Internet: Para descargar las imágenes Docker y dependencias necesarias.

Requisitos de Software

* Sistema Operativo: Distribución Linux basada en Debian (por ejemplo, Ubuntu).
* Docker: Instalado y configurado.
* Docker Compose: Instalado y configurado.

1. A diagram of a computer

   Description automatically generatedDiseño

Ilustración 1. Esquema del diseño del proyecto.

El funcionamiento de nuestro proyecto es simple, siguiendo el esquema:

1. Un cliente entra al restaurante y realiza un pedido.
2. El empleado del restaurante apunta el pedido a través de la terminal del restaurante utilizando el Programa de Cuentas de nuestro proyecto, donde asignará un numero de mesa al cliente y apuntará su pedido.
3. Cuando el cliente termina su comida pide la cuenta al servicio. El empleado generará el ticket de la cuenta y mediante el Programa de Cuentas imprimirá el QR con los datos de la transacción asociados.
4. El cliente introducirá sus credenciales en la pagina web del restaurante para autenticarse mediante su navegador contra la base de datos
5. Una vez accedido a su cuenta pulsará el botón de escanear QR.
6. El cliente escanea el QR con la aplicación de nuestra página. La información que lleva contenida el QR se lanza contra la base de datos donde se realiza un insert con la información del pedido. Dentro de esta transacción ya irán incluidos el número de cuenta, el momento exacto de la generación de la cuenta, el usuario asociado y el número de puntos que corresponden por este pedido al usuario.
7. La base de datos lanza un procedimiento para sumar a los usuarios los puntos correspondientes a los últimos inserts de la tabla de pedidos.
8. El usuario puede ver actualizado a través de su cuenta en la página web los puntos que tiene acumulados.

**FALTA**

**5.2.1. Funcionalidad**

**5.2.2. Interacción**

**No tienen que ser las dos, pero en este punto hacemos una demostración de como funciona todo. Hasta que este dejar espacio en blanco.**

1. Implementación y desarrollo

En este apartado vamos a mostrar las configuraciones y el desarrollo de cada una de las partes del proyecto.

* 1. Base de datos MySQL

Nuestra base de datos está diseñada para operar tanto con el programa que generará nuestros códigos QR como con nuestra página web, que está gestionada a través de un proyecto de Django.

A continuación, vamos a enumerar todas las tablas que hemos empleado en nuestro proyecto.

AQUÍ FALTARIA PONER LAS TABLAS DE GUILLE

Tabla

Descripción generada automáticamente con confianza baja

De todas estas tablas, la mayoría son tablas creadas por el propio Django, ya que las utiliza para usar funciones del propio proyecto, en nuestro caso las tablas mas importantes serian:

* **cuentarestaurante:** esta tabla se encarga de recibir los insert que se harán de los QR generados.

Esta es su estructura:

Texto

Descripción generada automáticamente

* **usuarios**: esta tabla se encarga de gestionar tanto el inicio de sesión de los usuarios como la creación de una nueva cuenta.

Esta es su estructura:

Texto

Descripción generada automáticamente con confianza media

La tabla tiene campos como “is\_staff” o “is\_superuser” porque la funcionalidad del inicio de sesión esta gestionada con funciones del propio Django sin embargo en nuestro no usamos estos campos, pero son necesarios para que funcione correctamente.

* 1. Programa de cuentas

El programa está diseñado para ser utilizado en un restaurante para gestionar los pedidos de las mesas, generar cuentas y crear códigos QR con los detalles de la cuenta. Utiliza una base de datos MySQL para almacenar la información del menú y maneja los pedidos y pagos mediante interacciones en la terminal. Los códigos QR generados permiten a los clientes escanear y obtener puntos que pueden ser canjeados en futuras visitas.

Este programa se trata de un código escrito en Python. Vamos a ir apartado por apartado del código explicando su función:

**A screenshot of a computer

Description automatically generatedImportaciones**

Estas son las importaciones de las librerías de Python necesarias para nuestro programa:

* **mysql.connector**: Para conectarse y operar con la base de datos MySQL.
* **math**: Proporciona funciones matemáticas, en este caso se usa para redondear números.
* **json**: Para trabajar con datos en formato JSON.
* **qrcode**: Para generar códigos QR.
* **datetime**: Para trabajar con fechas y horas.

A screen shot of a computer program

Description automatically generated**Funcion conectar\_base\_datos()**

Esta función intenta conectarse a una base de datos MySQL con las credenciales y parámetros especificados. Si la conexión es exitosa, devuelve la conexión; de lo contrario, imprime un mensaje de error y devuelve None.

**Función obtener\_platos\_y\_precios(conexion)**

A computer screen with text on it

Description automatically generated

Esta función obtiene una lista de platos y sus precios desde la base de datos. Ejecuta una consulta SQL para seleccionar id\_plato, nombre y precio de la tabla menu, y almacena los resultados en una lista de tuplas.

**Función crear\_cuenta\_pedido(platos\_precios, num\_mesa)**

A computer screen with text

Description automatically generated

Esta función permite crear una cuenta para un pedido. Los usuarios pueden ingresar el número del plato que desean agregar al pedido. La función suma los precios de los platos seleccionados y devuelve una lista que contiene el número de mesa y el total de la cuenta.

**Función seleccionar\_mesa(mesas\_ocupadas)**

A screenshot of a computer program

Description automatically generated

Esta función permite seleccionar una mesa para el pedido, asegurándose de que el número de mesa sea válido y que la mesa no esté ya ocupada. Devuelve el número de mesa seleccionado.

**Función menu()**

**A screenshot of a computer code

Description automatically generated**

Simplemente saca por pantalla el despliegue de opciones que se pueden realizar y devuelve en su salida el número que el usuario ha seleccionado.

A screen shot of a computer code

Description automatically generated**Función generar\_json\_pedido(pedido)**

Esta función genera un archivo JSON y un código QR para un pedido. Calcula los puntos del pedido, crea un archivo JSON con los detalles del pedido y genera un código QR que contiene los mismos datos. El archivo JSON y la imagen del código QR se guardan con nombres que incluyen la fecha y hora actuales.

**Función main()**

A screen shot of a computer program

Description automatically generated

Esta es la función principal del programa. Se encarga de coordinar todas las funciones:

1. Conecta a la base de datos.
2. Obtiene la lista de platos y precios.
3. Mantiene un bucle de operación que permite al usuario seleccionar opciones del menú:
   * Ver la lista de platos y precios.
   * Crear una cuenta para un pedido.
   * Pagar la cuenta de una mesa.
4. Maneja las mesas ocupadas y los pedidos.
5. Genera y guarda el archivo JSON y el código QR cuando se paga una cuenta.
   1. Django

Nuestra aplicación web estará gestionada por el framework de Python Django. La aplicación gestionará las páginas webs y su interacción con la base de datos, tanto el inicio de sesión y creación de cuentas, como la lectura del QR y su posterior inserción en la tabla correspondiente de la base de datos.

Todo el proyecto esta escrito en Python menos las páginas web que están en HTML, vamos a explicar paso a paso como crear y configurar este proyecto.

* + 1. Instalación y creación del proyecto en Django
* En este caso consideramos que previamente ya se encuentra instalado Python en la máquina.

1. Escribir en la terminal el comando de instalación de Django.



1. Crear un nuevo proyecto. En nuestro caso pondremos “tfg\_proyect” que será el nombre del proyecto.



1. Crear la aplicación dentro de la carpeta del proyecto. En nuestro caso usaremos restaurante.



Una vez realizados todos los pasos anteriores deberá quedar una estructura parecida a la de las imágenes:

**CARPETA DEL PROYECTO:**

Interfaz de usuario gráfica

Descripción generada automáticamente

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja**CARPETA DE LA APLICACIÓN DEL PROYECTO:**

* + 1. Desarrollo de la aplicación

Primero se debe configurar los archivos de configuración del proyecto a nivel global:

1. **SETTINGS.PY**

Texto

Descripción generada automáticamente

1. Primero añadimos la aplicación en la directiva “INSTALED\_APPS”.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

1. Añadimos la librería **OS** para poder incluir la ruta a la carpeta donde estarán nuestros archivos HTML.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

1. Establecemos la conexión con la base de datos.
2. **URLS.PY**
3. Establecer la ruta del archivo de urls de la aplicación para que pueda encontrar todos los archivos correctamente.

Texto

Descripción generada automáticamente

1. Configurar los archivos a nivel de aplicación:

Texto

Descripción generada automáticamente

En nuestro caso:

1. Texto

   Descripción generada automáticamentePrimero definimos el modelo de “cuentarestuarante” para poder manejar las inserciones en nuestra tabla posteriormente.
2. Definimos el modelo “Usuario” y “UsuarioManager” para que de esta manera sean compatibles con el sistema de autentificación de Django, también es necesario importar las librerías vistas en la primera captura de este apartado para que todo esto funcione correctamente.
   1. Creación de contenedores para MySQL, Django y NginX

### Dockerización de MYSQL y Django

Esta es una parte fundamental del proyecto ya que nos permite crear un contenedor con todas las aplicaciones necesarias para su lanzamiento y uso. Por supuesto, este proyecto solo es un esqueleto, en un entorno real habría que configurar el contenedor según las necesidades del empresario.

En nuestro caso hemos incluido en Docker tres imágenes: MYSQL, Django y NginX.

Pasos a seguir:

1. Necesitamos instalar Docker y Docker-compose en nuestra máquina, nosotros lo haremos en una máquina virtual de Ubuntu.
   1. Instalar dependencias necesarias:

sudo apt install apt-transport-https ca-certificates curl software-properties-common

* 1. Agregar la clave GPG oficial de Docker:

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add –

* 1. Agregar el repositorio de Docker a tus fuentes APT :

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable"

* 1. Actualizar el índice de paquetes de nuevo:

sudo apt update

* 1. Instalar Docker:

sudo apt install docker-ce

* 1. Instalar Docker-compose:

sudo curl -L "https://github.com/docker/compose/releases/download/<VERSION>/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

* 1. Dar permisos de ejecución al binario:

sudo chmod +x /usr/local/bin/docker-compose

1. Creamos un nuevo directorio donde queremos que se encuentre el contenedor. Copiamos dentro todos los recursos de nuestro proyecto de Django. Dejo un esquema básico y a modo de ejemplo de cómo es la estructura:

A screen shot of a computer

Description automatically generated

1. A screenshot of a computer program

   Description automatically generated Creamos un archivo Dockerfile en la raíz del proyecto (junto a manage.py) que definirá nuestro contenedor. Este Dockerfile concretamente es el que define nuestro contenedor de Django.
2. Creamos el archivo requirements.txt con los siguientes elementos:



Este archivo depende del contenedor y del proyecto de Django. Se utiliza para listar todas las dependencias de Python que el proyecto necesita para funcionar. Cada línea de este archivo especifica un paquete que debe ser instalado.

1. Crear el docker-compose.yml en la raiz del directorio:

A screenshot of a computer program

Description automatically generated

En este archivo es donde se definen las imágenes de los contenedores que tendremos conectados, en nuestro caso sería Django, MySQL y NginX. Es importante definir los volúmenes de cada imagen, estos volúmenes son útiles para que la información se quede guardada una vez los contenedores se paren.

En el caso del contenedor MySQL también definimos las credenciales de contraseña del usuario root y la base de datos que vamos a utilizar.

1. Modificar settings.py en la carpeta del proyecto de Django para usar las variables de entorno para la configuración de la base de datos:

A computer screen shot of a computer program

Description automatically generated

1. Crear un archivo nginx.conf en la raíz del directorio donde estará la configuración de nuestro servidor web.

A screenshot of a computer program

Description automatically generated

En este archivo se han definido las rutas virtuales a la carpeta con las imágenes, el puerto que debe usar el servidor y la conexión con Django entre otras configuraciones.

1. (OPCIONAL) Generar un archivo entrypoint.sh en la raíz para realizar varias acciones antes de iniciar los contenedores. Estas acciones en nuestro caso son:

• Esperar a que otros servicios estén listos, por ejemplo, esperar a que la base de datos MySQL esté lista antes de que Django intente conectarse a ella.

• Ejecutar comandos de inicialización

• Realizar migraciones de la base de datos. Esto se utiliza para mantener la base de datos de nuestro contenedor siempre actualizada.

• Recolectar archivos estáticos.

• Ejecutar el servidor de aplicaciones, iniciar Gunicorn (o cualquier otro servidor WSGI) para servir la aplicación Django.

A screenshot of a computer program

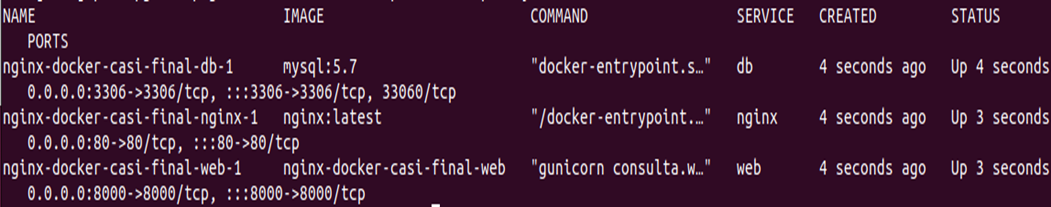
Description automatically generated

1. Construir y ejecutar los contenedores:

docker-compose up –build

docker-compose up -d

1. Escribir el comando: “docker-compose ps”. Con esto podemos observar el estado de nuestros contenedores.



* CONSIDERACIONES A TENER EN CUENTA:

Docker no puede levantar un contenedor si hay un servicio en la maquina ocupando alguno de los puertos necesarios para alguno de los contenedores. Por ejemplo, si la máquina que estamos empleando ya tiene un MySQL, es imperativo detener el servicio en ejecución o derivar el servicio a un puerto distinto. Otra opción es configurar el contenedor para que se lance por otro puerto.

Es importante revisar que las versiones que vas a utilizar de las imágenes, librerías o módulos sean compatibles entre sí.

Si la maquina tiene más contenedores creados anteriormente también se debe revisar si pueden entrar en conflicto con los contenedores de nuestro proyecto. Puede comprobarse si hay algún contenedor más corriendo de la siguiente forma:

• Comando: “docker ps”

En caso de que haya algún contenedor que sea necesario parar su ejecución podemos observar el PID del contenedor y ejecutar el siguiente comando:

• Comando: “docker stop <id\_contenedor>”

# Manual de usuario

En este apartado nos centraremos en como instalar y ejecutar los recursos de nuestro proyecto para su funcionamiento. Sera necesario que tengamos un sistema operativo Linux para la instalación.

## Manual de instalación

* Instalación de Docker y Docker Compose: revisar apartado 4.4.1 Dockerización de MYSQL y Django

**Paso 1**: Conectar el USB y Copiar la Carpeta

Copiar la carpeta que contiene la configuración Docker (por ejemplo, tfg) a una ubicación en tu sistema, como tu directorio home.

cp ./<path del proyecto en el usb> <path del directorio del proyecto>

**Paso 2**: Navegar al Directorio de la Carpeta Copiada

* Abrir una terminal y navegar al directorio donde la carpeta tfg fue copiada.

Cd <path del directorio del proyecto>

**Paso 3**: Construir y Ejecutar el Contenedor tfg:

* Construir la imagen Docker (si es necesario):

docker-compose build

* Ejecutar el contenedor tfg:

docker-compose up -d

**Paso 4**: Verificar el Contenedor en Ejecución:

docker ps

**Paso 5**: Acceder a los Servicios

* Abrir el navegador web y navegar a la URL correspondiente (por ejemplo, http://localhost) para acceder a la aplicación.

**Paso 6**: Gestionar el Contenedor tfg

* Detener el contenedor**:**

docker-compose down

* Ver los logs del contenedor tfg:

docker logs tfg

## 4.2 Manual de ejecución

**Paso 1**: Ejecutar los contenedores

* Una vez instalado Docker y Docker-Compose y con el contenedor tfg ya creado levantamos el servicio:

docker compose up -d

* Observar si la ejecución de los contenedores es correcta:

docker compose ps

* Entrar en [http://localhost](http://localhost/) o http://<ip\_del\_servidor> para entrar en el proyecto desde el servidor local.

**Paso 2**:

* Conectar contenedor mysql con Workbench:

En caso de que decidas tener un gestor de Base de Datos como workbench, esta es la forma adecuada de realizar la conexión con el contenedor:

1.- Entramos en la aplicación MySQL Workbench

2.- Pulsamos sobre la etiqueta “+” debajo de MySQL Connections para añadir una nueva conexión:

A screenshot of a computer

Description automatically generated

3.- Escribe la dirección ip de nuestro servidor. En nuestro caso pondremos la dirección de loopback porque tenemos el contenedor en la misma maquina que el Workbench:

A screenshot of a computer

Description automatically generated

4.- Conectar el usuario adecuado:

A screenshot of a computer

Description automatically generated

5.- Clickamos sobre Pasword: Store in Vault e introducimos la contraseña:

A screenshot of a computer

Description automatically generated

6.- Clickamos en ok:

A screenshot of a computer

Description automatically generated

7.- Ahora podemos observar cómo se ha creado la conexión:

A screenshot of a computer

Description automatically generated

# Conclusión

¾ cara

## Opinión Personal

½ cara