Manual Técnico

Proyecto: Sistema Gestor de Barbería Online

MATERIA: Ingeniería se Software

Universidad de Guadalajara

Centro Universitario de Ciencias Exactas e Ingenierías

Departamento de Ciencias Computacionales

Sistema Gestor de citas en barbería

MANUAL TÉCNICO

CONTENIDO

1. Objetivos

El objetivo de este documento es proporcionar una guía técnica completa para la instalación, configuración, operación y mantenimiento del sistema gestor de barbería. Este manual está destinado a facilitar la implementación correcta del sistema en distintos entornos

1.1 Objetivos Específicos.

- *Documentar los requerimientos de hardware y software para ejecutar el sistema*
- *Describir paso a paso el proceso de instalación y configuración.*
- *Detallar la arquitectura física del sistema y sus componentes.*
- *Especificar los distintos tipos de usuarios y sus privilegios.*
- *Establecer procedimientos de contingencia ante errores comunes*

2. Alcance

Este manual está dirigido a administradores de sistemas, desarrolladores, personal de soporte técnico y usuarios con conocimientos básicos en bases de datos, servidores y configuración de aplicaciones Python / Flask.

Conocimientos previos recomendados:

- *Manejo de sistema operativo Windows o Linux*
- *Uso de terminal (CMD o Git Bash)*
- *Manejo básico de bases de datos (MariaDB).*
- *Comprensión de entornos virtuales en Python*
- *Afición de archivos de configuración*

Éste documento está dirigido a: Administradores de sistemas

Conocimientos básicos en: Bases de datos, servidores y configuración de aplicaciones Python / Flask

3. Requerimientos Técnicos

A continuación se describen y enlistan los requerimientos tanto en hardware como de software necesarios para el correcto funcionamiento del Sistema Gestor de Barbería. Estos requisitos aseguran que la aplicación web y la base de datos se ejecuten de forma estable y eficiente en entornos de desarrollo o producción local.

3.1 Requerimientos Mínimos de Hardware.

Procesador: Intel Core i3 o equivalente (2 núcleos, 2.0 GHz mínimo)

Memoria RAM (Mínimo): 4 GB

Disco Duro: 500MB de espacio libre

3.2 Requerimientos Mínimos de Software.

Privilegios de Administrador:
Si para instalar dependencias y servicios como MariaDB o configurar entornos virtuales

Sistema Operativo:
Windows 10 o superior / Ubuntu 20.04 o superior

Python:
Versión 3.10 o superior

Servidor de Base de Datos:
MariaDB versión 10.4 o superior

Entorno de desarrollo recomendado:
Visual Studio Code

Dependencias principales:
Flask, Flask-MySQL, PyMySQL, dotenv

Navegador web:
Google Chrome o Mozilla Firefox

4. Herramientas Utilizadas para el Desarrollo.

- **Herramienta 1**
  Visual Studio Code
  Editor de código multiplataforma con soporte para extensiones, control de versiones y resaltado de sintaxis.
  Motivo por el cual fue utilizado: Fue utilizado como entorno principal de desarrollo para escribir el código backend en Python, editar archivos HTML/CSS y administrar el proyecto mediante integración con GitHub

- **Herramienta 2**
  Python 3.10
  Descripción: Lenguaje de programación interpretado, ampliamente utilizado para el desarrollo web, scripting y automatización
  Motivo por el cual fue utilizado: Es el lenguaje base sobre el que está construido el sistema. Permite desarrollar el backend utilizando el microframework Flask

- **Herramienta 3**
  Descripción: Microframework web ligero par aPython.
  Motivo por el cual fue usado: Se utilizó para estructurar el backend del sistema, administrar rutas, formularios, sesiones de usuario y comunicación con la base de datos.

- **Herramienta 4**
  DBeaver
  Descripción: Cliente visual para la gestión de bases de datos
  Motivo por el cual fue utilizado: Sirvió para visualizar, modificar y exportar la estructura y contenido de la base de datos MariaDB utilizada por el sistema.

- **Herramienta 5**
  Git y GitHub

Descripción: Sistema de control de versiones distribuido y plataforma de alojamiento de repositorios.
Motivo por el cual fue utilizado: Para llevar control del código fuente, realizar copias de seguridad, y facilitar la colaboración con otros integrantes del equipo.

5. Instalación

A continuación se describen los pasos necesarios para la instalación del Sistema Gestor de Barbería en un entorno local:

Paso 1: Clonar el repositorio del proyecto

Abrir la terminal (GitBash o CMD)

Navegar a la carpeta donde se desea guardar el proyecto

Ejecutar el siguiente comando:

```
git clone https://github.com/usuario/repositorio-barberia.git
```

Paso 2: Acceder al directorio del proyecto

```
cd repositorio-barberia
```

Paso 3: Crear un entorno virtual en Python

```
python -m venv venv
```

Paso 4: Activar el entorno virtual

En Git Bash

```
source venv/Scripts/activate
```

En CMD Windows:

```
venv\Scripts\activate
```

Paso 5: Instalar las dependencias necesarias

```
pip install -r requirements.txt
```

Paso 6: Configurar el archivo .env (si aplica)

Crear un archivo .env en la raíz del proyecto y colocar las variables de conexión, por ejemplo:

```
DB_HOST=localhost

DB_PORT=3307

DB_USER=root

DB_PASSWORD=guacamole

DB_NAME=barberManager
```

Paso 7: Verificar que el servidor de base datos (MariaDB) esté en ejecución
En Windows Verificar desde XAMPP o desde Servicios (services.msc).
Asegurarse de que el puerto y usuarios coincidan con los del archivo . env o conexion.py

Paso 8: Ejecutar el script SQL para crear la base de datos
Abrir Dbeaver y otro cliente SQL
Conectarse al servidor y ejecutar el archivo tables.sql incluido en el repositorio, o importar el respaldo del equipo.

Paso 9: Iniciar la aplicación
Con el entorno virtual aún activo:
python main.py

El sistema estará disponible en:
http://127.0.0.1:5050/

6. Configuración

El sistema permite realizar configuraciones esenciales antes de su ejecución para asegurar una correcta conexión con la base de datos, la gestión del entorno de desarrollo y la personalización de parámetros del entorno local.

Configuración de conexión a la base de datos

En el archivo helper/conexión.py , se encuentra la función que establece la conexión a la base de datos. Deben configurarse los siguientes parámetros según el entorno:

```
import pymysql

def Connection():
    return pymysql.connect(
    host= “localhost”,
    port= ”3306” Verificar que este el correcto puerto asignado en la instalación del Gestor de la Base datos
    user= ”root”,
    password= “     “ Aqui va tambien el password asociado al Gestor de Base de Datos
```

db= “barberManager”
)

host: Dirección del servidor de base de datos.
port: Puerto de conexión de MariaDB (3306 por defecto)
user / password: Usuario con permisos sobre la base
db: Nombre exacto de la base de datos que utilizará el sistema

Configuración del entorno virtual
Para aislar las dependencias del sistema y evitar conflictos, es necesario activar el entorno virtual cada vez que se trabaje con el proyecto:

```
# Activar entorno en Git Bash
source venv/Scripts/activate

# Activar entorno en CMD
venv\Scripts\activate
```

Instalación de dependencias
En caso de no contar con las librerías necesarias, se deben instalar desde el archivo requirements.txt:

```
pip install -r requirements.txt
```

Variables de entorno (.env)
Opcionalmente, el sistema puede adaptarse para usar variables de entorno mediante una librería como python-dotenv. Esto permite ocultar datos sensible:

```
DB_USER=root
DB_PASSWORD=guacamole
DB_NAME=barberManager
```

Estas variables se cargan automáticamente al inicio si se utiliza la extensión adecuada.

Personalización del puerto de ejecución

En el archivo main.py , se puede definir el puerto manualmente si el predeterminado (5050) está ocupado:

```
if __name__ == '__main__':

app.run(debug=True, port=5050)
```

7. Diseño de la Arquitectura Física.

El sistema gestor de barbería fue desarrollado para ejecutarse en un entorno local o de red interna. A continuación, se describen las características físicas y de red del entorno de pruebas y desarrollo utilizado.

- Nombre de los equipos:
  DESKTOP-GERA(Equipo del desarrollador
  LAPTOP-PART(Equipo del colaborador)

- Ubicación física dentro del centro de cómputo, y en el sitio alterno de procesamiento: Desarrollo y pruebas realizadas en instalaciones personales de los integrantes del equipo, en un entorno local simulado.

- Direcciones IP asignadas: 127.0.0.1 (localhost) para desarrollo individual.

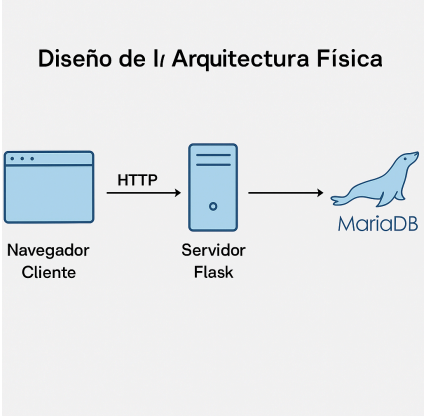
  En entorno de red: 192.168.1.101 y 192.168.0.102 (asignadas por DHCP en red local para pruebas entre equipos).

- Puertos TCP/UDP necesarios para comunicación:

  5050 -> Puerto asignado para el servidor Flask
  3306 0 3307 -> Puerto del servidor MariaDB
  Puertos HTTP estándar (80/443) no utilizados en entorno local.

- Dependencias con otros sistemas: Requiere que el servicio de base de datos MariaDB esté activo y accesible
  Opcionalmente, puede integrarse con herramientas como GitHub (Via HTTPS) para versiones

- Interrelaciones de conexión entre los equipos.
  En pruebas de red, el servidor Flask alojado en una máquina responde a peticiones HTTP desde otro equipo conectado mediante la IP local, permitiendo la simulación de un entorno cliente-servidor.



8. Usuarios

El sistema contempla distintos tipos de usuarios según el nivel de interacción con la aplicación, base de datos y sistema operativo. A continuación, se describen los perfiles necesarios y sus privilegios correspondientes.

8.1 Usuarios de base de datos

Nombre del usuario: root

Usuario administrador principal de MariaDB, utilizado en desarrollo y pruebas para acceder sin restricciones a todas las funciones de la base de datos del sistema

Grupos a los que pertenece: administradores

Privilegios generales a nivel de base de datos:
Crear y Eliminar bases de datos
Acceder a todos los esquemas
Asignar privilegios a otros usuarios

Privilegios sobre objetos de las mismas *(tablas, vistas, triggers, etc.):*
SELECT, INSERT, UPDATE, DELETE
CREATE, ALTER, DROP
TRIGGER Y VIEW completos sobre todas las tablas

8.2 Usuarios de sistema operativo

Nombre del usuario: *User(Usuario local en Windows 10)*

Usuario principal del entorno de desarrollo. Tiene a su cargo la instalación de dependencias, ejecución del servidor Flask, edición del código fuente y administración del servidor de base de datos local.

Grupos a los que pertenece: Administradores

Privilegios sobre carpetas:
Control total sobre la carpeta del proyecto BarberManager
Acceso de lectura y escritura sobre archivos de configuración (.env, .py)
Permiso para modificar rutas del entorno virtual (venv/) y archivos temporales

Acceso a la carpeta de instalación de MariaDB (puede reiniciar el servicio)

8.3 Usuarios de aplicaciones

Nombre del usuario: admin

Usuario propietario y principal administrador del sistema. Tiene acceso completo a todas las funciones de la plataforma incluyendo gestión de usuarios, barberos, citas, productos, ventas y comentarios.

Grupos a los que pertenece: Administradores de sistema

Privilegios dentro de la aplicación:
- Crear, editar y eliminar usuarios del sistema
- Registrar y gestionar barberos, servicios y productos
- Visualizar reportes de ventas y citas
- Modificar información de la base de datos desde la interfaz
- Acceso total al panel de administración

9. Contingencias y soluciones.

A continuación se enlistan algunas de las contingencias más frecuentes durante la instalación, configuración y uso del sistema, junto con sus respectivas soluciones:

Contingencia 1: Error al conectar con la base de datos
Descripción: Al ejecutar la explicación, se muestra un error `Access denied for user 'root'@'localhost'` *o* `Can't connect to MySQL server.`

Solución recomendada:
Verificar que el servicio de MariaDB esté en ejecución.
Revisar que los parámetros de conexión (usuario, contraseña, puerto y nombre de base de datos) en conexion.py o .env estén correctos
Asegurarse de que se esté usando el puerto correcto 3306 o 3307 según configuración

Contingencia 2: El entorno virtual no se activa correctamente

Descripción: al intentar activar el entorno con source venv/Scripts/activate, el sistema lanza error.

Solución recomendada:

Verificar que el entorno virtual haya sido creado con python -m venv venv.

Usar el comando correcto según el sistema(Git Bash vs CMD).

En algunos casos, se debe habilitar la ejecución de scripts con:

Set-ExecutionPolicy RemoteSigned -Scope CurrentUser

Contingencia 3: Al ejecutar el script SQL, se presentan errores de clave foránea
Descripción: El script muestra errores Foreign key is incorrectly formed.

Solución recomendada:

Asegurarse de ejecutar las tablas en el orden correcto (primero las referenciadas).

Verificar que los tipos de datos coincidan entre claves foráneas y primarias.

Usar InnoDB como motor de almacenamiento

Contingencia 4: La aplicación iniciar pero no carga datos en el navegador
Descripción: El sitio abre en el navegador pero las secciones como barberos, usuarios o productos aparecen vacías.
Solución recomendada:

Confirmar en Dbeaver que las tablas contienen datos.

Verificar que las consultas SQL en el backend (por ejemplo, en helpers/.py) no estén vacías o mal estructuradas.*

Asegurar que la conexión a la base de datos apunta al esquema correcto