Documentación Técnica: Sistema de Gestión de Restaurantes

---

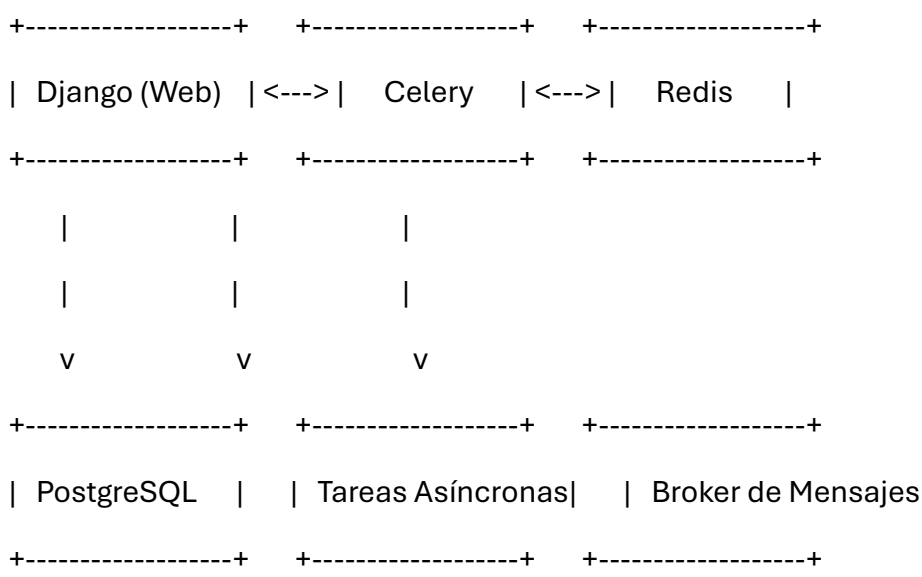
1. Introducción

El **Sistema de Gestión de Restaurantes** es una aplicación web desarrollada con Django, Django REST Framework, Celery y Redis. El sistema permite gestionar restaurantes, menús, usuarios y órdenes, además de generar reportes en formato CSV de manera asíncrona.

---

2. Arquitectura del Sistema

2.1 Diagrama de Arquitectura



2.2 Componentes Principales

1. **Django (Web)**:
    - Framework principal para la lógica de la aplicación.
    - Expone APIs RESTful para gestionar restaurantes, usuarios, menús y órdenes.
2. **Celery**:
    - Sistema de colas de tareas asíncronas.
    - Se encarga de procesar tareas en segundo plano, como la generación de reportes.

3. **Redis**:
    - Broker de mensajes para Celery.
    - Almacena la cola de tareas y los resultados de las tareas asíncronas.
4. **PostgreSQL**:
    - Base de datos relacional para almacenar la información del sistema.

---

3. Flujos de Trabajo

3.1 Gestión de Restaurantes

- **Crear Restaurante**: Los usuarios pueden crear nuevos restaurantes.
- **Listar Restaurantes**: Obtener una lista de todos los restaurantes.
- **Actualizar/Eliminar Restaurante**: Modificar o eliminar un restaurante existente.

3.2 Gestión de Usuarios

- **Crear Usuario**: Registrar nuevos usuarios (clientes o administradores).
- **Listar Usuarios**: Obtener una lista de todos los usuarios.
- **Actualizar/Eliminar Usuario**: Modificar o eliminar un usuario existente.

3.3 Gestión de Órdenes

- **Crear Orden**: Registrar una nueva orden asociada a un usuario y un restaurante.
- **Listar Órdenes**: Obtener una lista de todas las órdenes.
- **Actualizar/Eliminar Orden**: Modificar o eliminar una orden existente.

3.4 Generación de Reportes

- **Generar Reporte de Ventas**: Crear un reporte en formato CSV con las ventas de un restaurante en un mes específico.
- **Descargar Reporte**: Descargar el reporte generado.

---

4. Detalles de Implementación

4.1 Tecnologías Utilizadas

- **Backend**: Django, Django REST Framework.
- **Base de Datos**: PostgreSQL.
- **Tareas Asíncronas**: Celery.
- **Broker de Mensajes**: Redis.
- **Formato de Reportes**: CSV.

4.2 Estructura del Proyecto

```
restaurant_management/
 ├── manage.py
 ├── restaurant_management/
 │   ├── __init__.py
 │   ├── settings.py
 │   ├── urls.py
 │   ├── wsgi.py
 │   └── celery.py
 ├── restaurants/
 │   ├── migrations/
 │   ├── __init__.py
 │   ├── admin.py
 │   ├── apps.py
 │   ├── models.py
 │   ├── serializers.py
 │   ├── views.py
 │   └── filters.py
 ├── users/
```

```
│   ├── migrations/
│   ├── __init__.py
│   ├── admin.py
│   ├── apps.py
│   ├── models.py
│   ├── serializers.py
│   ├── views.py
│   └── filters.py
├── orders/
│   ├── migrations/
│   ├── __init__.py
│   ├── admin.py
│   ├── apps.py
│   ├── models.py
│   ├── serializers.py
│   ├── views.py
│   └── tasks.py
├── menu/
│   ├── migrations/
│   ├── __init__.py
│   ├── admin.py
│   ├── apps.py
│   ├── models.py
│   ├── serializers.py
│   ├── views.py
```

```
│   └── filters.py
├── requirements.txt
├── docker-compose.yml
└── Dockerfile
```

4.3 Configuración de Celery y Redis

- **Celery**: Se configura en celery.py para usar Redis como broker de mensajes.
- **Redis**: Se ejecuta en un contenedor Docker y se conecta a Celery mediante la URL redis://redis:6379/0.

4.4 Endpoints de la API

Restaurantes

- **Listar Restaurantes**: GET /api/restaurants/
- **Crear Restaurante**: POST /api/restaurants/create/
- **Detalles de un Restaurante**: GET /api/restaurants/<int:pk>/
- **Actualizar Restaurante**: PUT /api/restaurants/update/<int:pk>/
- **Eliminar Restaurante**: DELETE /api/restaurants/delete/<int:pk>/

Usuarios

- **Listar Usuarios**: GET /api/users/
- **Crear Usuario**: POST /api/users/create/
- **Detalles de un Usuario**: GET /api/users/<int:pk>/
- **Actualizar Usuario**: PUT /api/users/update/<int:pk>/
- **Eliminar Usuario**: DELETE /api/users/delete/<int:pk>/

Órdenes

- **Listar Órdenes**: GET /api/orders/
- **Crear Orden**: POST /api/orders/create/
- **Detalles de una Orden**: GET /api/orders/<int:pk>/
- **Actualizar Orden**: PUT /api/orders/update/<int:pk>/

- **Eliminar Orden**: DELETE /api/orders/delete/<int:pk>/

Reportes

- **Generar Reporte de Ventas**: POST /api/restaurants/<int:restaurant_id>/generate-sales-report/
- **Descargar Reporte**: GET /api/reports/download/<str:filename>/

---

5. Instalación y Despliegue

5.1 Requisitos

- Python 3.9 o superior.
- Docker (opcional, para Redis).
- Git.

5.2 Pasos para Desplegar

1. **Clonar el Repositorio**:

   git clone https://github.com/tu-usuario/tu-repositorio.git

   cd tu-repositorio

2. **Crear un Entorno Virtual**:

   python3 -m venv venv

   source venv/bin/activate

3. **Instalar Dependencias**:

   pip install -r requirements.txt

4. **Configurar el Entorno**:
   - Crear un archivo .env con las variables de entorno necesarias.
5. **Iniciar Redis con Docker**:

   docker run -d --name my-redis -p 6379:6379 redis

6. **Iniciar el Proyecto de Django**:

   python manage.py migrate

```
python manage.py runserver
```

7. **Iniciar Celery**:

```
celery -A restaurant_management worker --loglevel=info
```

---

6. Pruebas

6.1 Pruebas Unitarias

Ejecuta las pruebas unitarias con el siguiente comando:

python manage.py test

6.2 Pruebas de Integración

Prueba los endpoints de la API usando herramientas como Postman o Insomnia.

---

7. Licencia

Este proyecto está bajo la licencia MIT. Consulta el archivo LICENSE para más detalles.

---

8. Contribuciones

Las contribuciones son bienvenidas. Por favor, abre un issue o envía un pull request en el repositorio de GitHub.

---

9. Contacto

Para más información, contacta al equipo de desarrollo en hanzbk@gmail.com.