


Documento Técnico – Tecnologías y Arquitectura del Proyecto

 Sistema de Gestión de Consumos y Arrendamientos Residenciales
Curso: Ingeniería de Software I (2016701)
Universidad Nacional de Colombia – Sede Bogotá
Julio de 2025

1. Introducción

Nuestro sistema automatiza el registro y la distribución de los costos de servicios públicos y arriendo en un edificio de seis apartamentos. Buscamos entregar una herramienta clara, confiable y fácil de mantener que sirva como ejemplo de buenas prácticas de Ingeniería de Software. Esta solución también busca ser extensible, escalable y amigable con el usuario final. 🏠

2. Tecnologías seleccionadas

Python 3.11+ con PyQt5

Python fue elegido por su sintaxis sencilla, su comunidad activa y la extensa biblioteca de módulos que aceleran el desarrollo. En combinación con PyQt5, logramos construir interfaces modernas desde Qt Designer, manteniendo una separación estricta entre lógica de negocio e interfaz gracias al sistema de señales y slots.

MySQL 8.0 CE

Este motor relacional garantiza transacciones ACID, integridad referencial y herramientas gráficas como DBeaver o MySQL Workbench para una administración eficiente. La edición Community (CE) es gratuita y cubre con creces las necesidades académicas y de pruebas del proyecto.

Librerías complementarias

Para conectarnos con la base de datos utilizamos `mysql-connector-python`, el driver oficial de Oracle. El manejo avanzado de fechas lo cubren `python-dateutil` y `pytz`, garantizando precisión en vencimientos y reportes, incluso considerando zonas horarias.

3. ¿Por qué no otras opciones?

Lenguajes como Java (con JavaFX) o C# (con .NET/WPF) fueron considerados. Ambos ofrecen herramientas robustas, tipado estático y rendimiento optimizado. Sin embargo, para los objetivos del curso, Python demostró ser más ágil, menos verboso y con menor curva de aprendizaje.

Java implicaba mayor tiempo de configuración para cada componente visual, mientras que .NET requería ajustes adicionales para entornos fuera de Windows. En cambio, Python y PyQt5 nos permitieron avanzar rápidamente sin perder claridad ni modularidad. ⚖️

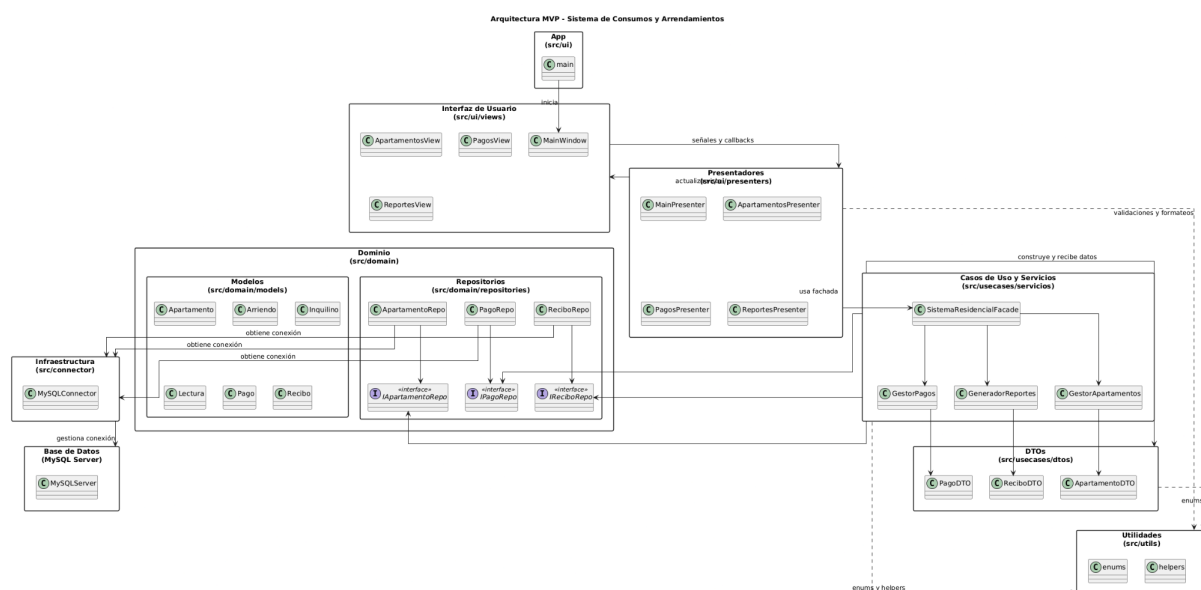
4. Arquitectura de alto nivel


La aplicación se organizó en cinco capas, cada una dependiente únicamente de la inferior, respetando el principio de inversión de dependencias:

1. **UI (src/ui):** vistas construidas con PyQt5 y sus presentadores.
2. **Casos de uso (src/usecases):** orquestación de servicios y fachada.
3. **Dominio (src/domain):** entidades del sistema y adaptadores de acceso a datos.
4. **Infraestructura (src/connector):** lógica de conexión a la base de datos.
5. **Base de datos:** servidor MySQL 8.0 CE para almacenamiento persistente.

📁 Diagrama de Arquitectura (formato PNG):

La siguiente imagen muestra la arquitectura general del sistema, con separación por capas y flujo de datos entre ellas.



A continuación comparto el svg donde se puede visualizar un poco mejor la estructura 

5. Estructura del proyecto

La estructura del proyecto sigue una organización modular que respeta los principios de arquitectura limpia. Cada carpeta representa una capa o responsabilidad específica:

```
None
src/

├─ connector/

|   └─ mysql_connector.py           # Encapsula la conexión a
la base de datos

|

├─ domain/

|   └─ models/

|       └─ apartamento.py          # Modelo del apartamento

|       └─ lectura.py              # Modelo de lectura de
consumo

|       └─ recibo.py                # Modelo de recibo de pago

|       └─ repositories/

|           └─ apartamento_repo.py # Acceso a datos de
apartamentos

|           └─ pago_repo.py         # Acceso a datos de pagos

|

├─ usecases/

|   └─ servicios/
```

```

|   |   |— gestor_apartamentos.py # Lógica de gestión de
apartamentos

|   |   |— generador_reportes.py  # Servicio de creación de
reportes

|   |— dtos/                        # Objetos de transferencia
de datos (DTOs)

|

|— ui/

|   |— views/                      # Ventanas y formularios

|   |— presenters/                # Lógica que conecta UI con
los casos de uso

|   |— main.py                    # Entrada principal de la
app

|

|— utils/

    |— enums.py                   # Enumeraciones
reutilizables


    |— helpers.py                 # Funciones auxiliares




```

6. Conclusiones y siguientes pasos

La combinación de Python, PyQt5 y MySQL ha demostrado ser efectiva para desarrollar un sistema funcional, claro y escalable en un entorno académico. Los patrones MVP y Facade aportaron orden, flexibilidad y claridad tanto en el código como en la experiencia del usuario.

Para las siguientes etapas se proyecta:

-  Integrar pruebas unitarias para presentadores y repositorios.

-  Validar scripts SQL en el entorno de desarrollo.
-  Implementar un pipeline básico de generación de reportes en PDF.
-  Iniciar pruebas para despliegue continuo y mantenimiento automático.

Este trabajo representa una base sólida que permite escalar, mantener y extender funcionalidades sin comprometer la calidad del sistema.