

**Documentación Técnica – Proyecto APT: Sistema de Gestión Inmobiliaria**

**Nombres:** Sebastian Gajardo

Jose Herrera

Diego Leiva

**Docente:** Cindy Contador

**Índice**

[**1. Introducción Técnica**](#_heading=h.2in19k3ashj6) **3**

[**2. Modelo 4+1**](#_heading=h.ofc6csv0ce1d) 3

[2.1 Vista Lógica](#_heading=h.ccpsaaxhlua1) 4

[2.2 Vista de Desarrollo](#_heading=h.hxgvwxrxpjdz) 4

[2.3 Vista de Procesos](#_heading=h.4odal9jywabz) 4

[2.4 Vista Física](#_heading=h.5ykp0wz2g44k) 5

[2.5 Vista de Escenarios](#_heading=h.a6dvte4gi7lt) 5

[**3. Estilo Arquitectónico Utilizado**](#_heading=h.rophrbrm8iar) 6

[Justificación](#_heading=h.vi5qt1ro9m92) 6

[**4. Conclusión**](#_heading=h.thxe5ccpcm2z) 6

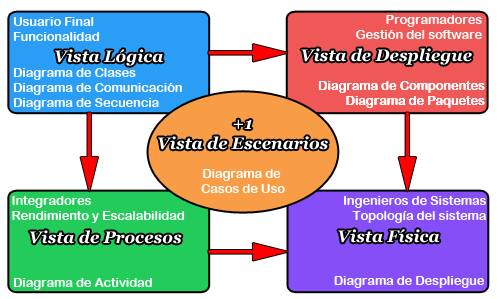
# 1. Introducción Técnica

El presente documento describe la arquitectura técnica del Sistema APT, una plataforma web desarrollada para digitalizar y centralizar los procesos de corretaje de propiedades. Su diseño se basa en el modelo arquitectónico 4+1, que permite abordar la solución desde diferentes puntos de vista técnicos y funcionales, asegurando claridad, escalabilidad, trazabilidad y mantenibilidad.

Además, se describe el estilo arquitectónico utilizado, las tecnologías seleccionadas y la justificación de decisiones técnicas adoptadas a lo largo del proyecto.

## 2. Modelo 4+1

El modelo 4+1 organiza la arquitectura en cinco vistas que representan las necesidades de distintos stakeholders. A continuación, se detallan cada una aplicadas al sistema APT:



### 

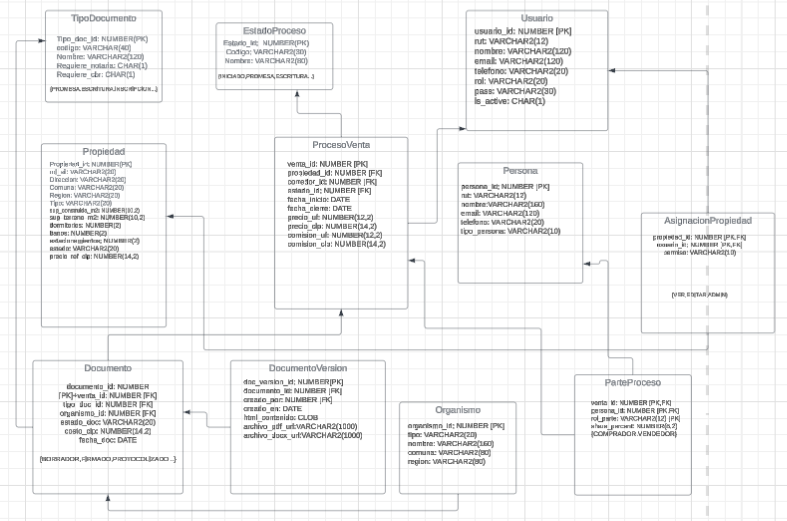
### 2.1 Vista Lógica

Objetivo: mostrar la estructura funcional y los objetos del dominio.

* Entidades principales: Propiedad, Cliente, Usuario, Documento.
* Relación entre propiedades y clientes; usuarios con distintos roles; documentos vinculados a propiedades.
* Diagramas: diagrama de clases UML y diagrama de casos de uso.

Esta vista representa cómo se modelan los conceptos del negocio y la lógica de aplicación, incluyendo operaciones como CRUD, validaciones y asociaciones.

**Diagrama de clases**



**Link diagrama:**https://lucid.app/lucidchart/261de513-e0b9-4554-bee7-f46e0d827fe8/edit?invitationId=inv\_1adc1f3a-7e99-49b1-9d76-c74d6f8fd0ab&page=0\_0#

#### 

#### 2.2 Vista de Desarrollo

Objetivo: mostrar la organización del software desde el punto de vista de los desarrolladores.

* Frontend: HTML, CSS, Bootstrap
* Backend:Django/Flask
* Base de datos: SQL (modelo relacional)
* Control de versiones: GitHub

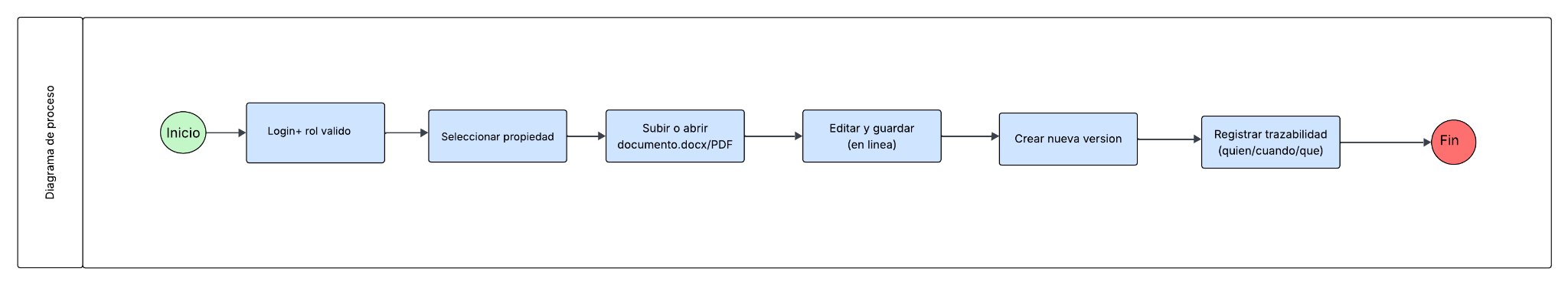
Se adopta una estructura modular que separa las capas de presentación, lógica y acceso a datos para facilitar el trabajo colaborativo y el mantenimiento del código.

##### 2.3 Vista de Procesos

Objetivo: describir los procesos en tiempo de ejecución y su interacción.

* Comunicación entre cliente y servidor mediante API REST.
* Procesos concurrentes como autenticación, edición de documentos, subida de archivos y auditoría de acciones.
* Manejadores de sesión y control de accesos por rol.

Diagrama sugerido: diagrama de secuencia que muestre el flujo de un proceso como la subida y edición de un contrato.



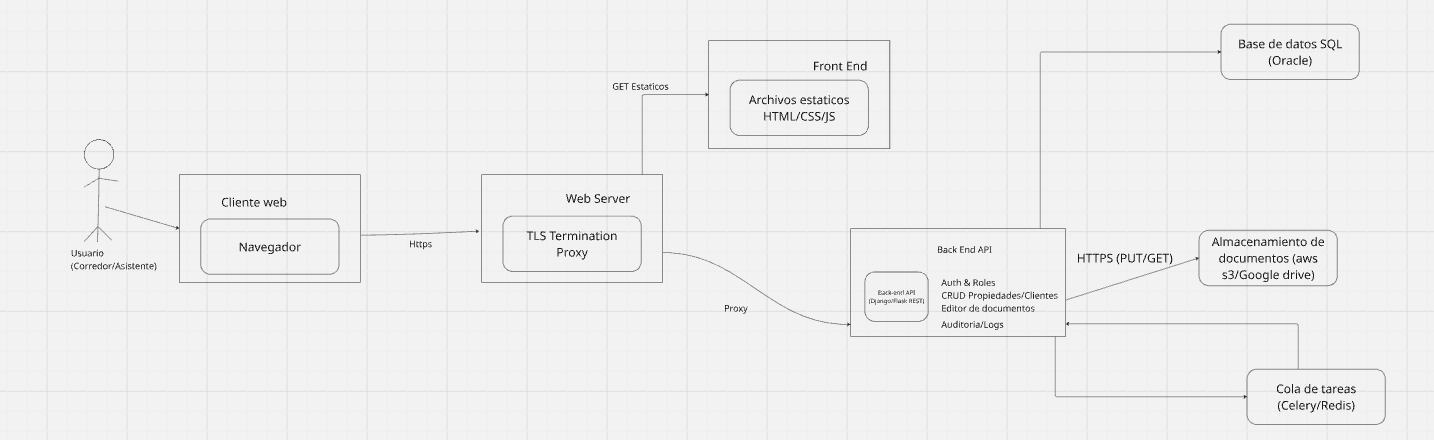
###### 2.4 Vista Física

Objetivo: ilustrar cómo se despliega el sistema en la infraestructura de hardware y red.

* Separación entre frontend, backend y base de datos
* Documentos alojados en servicios externos como AWS S3 o Driv

Se puede representar con un diagrama de despliegue que visualice la arquitectura distribuida cliente-servidor.

Diagrama de despliegue



###### 

###### 2.5 Vista de Escenarios

Objetivo: Validar la arquitectura con escenarios concretos de uso.

Casos principales:

* Registrar nueva propiedad
* Registrar y editar clientes
* Autenticación con control de roles
* Subida de documentos legales
* Edición de documentos Word en línea
* Visualización de métricas en dashboard
* Registro de historial de acciones del usuario

Estos casos garantizan que la arquitectura técnica soporte los requerimientos del negocio.

###### 3. Estilo Arquitectónico Utilizado

El sistema adopta una arquitectura en capas con un patrón cliente-servidor, ampliamente aceptado para aplicaciones web modernas.

Capas:

* Capa de presentación: HTML + CSS + Bootstrap
* Capa lógica: Django/Flask
* Capa de persistencia: SQL
* Servicios REST: permiten la interacción entre cliente y servidor de manera desacoplada.

###### 

###### Justificación

Este estilo fue seleccionado por su simplicidad, modularidad y alineación con el enfoque ágil del proyecto. Permite a los desarrolladores trabajar de forma paralela y garantiza un crecimiento ordenado del sistema.

El uso de Bootstrap asegura un diseño responsive y accesible desde distintos dispositivos. Django/Flask y SQL ofrecen buen rendimiento y escalabilidad para aplicaciones de este tipo.

###### 4. Conclusión

La arquitectura técnica del sistema APT fue definida para garantizar robustez, claridad estructural y escalabilidad. El uso del modelo 4+1 permitió documentar de forma integral las decisiones técnicas tomadas, considerando tanto la perspectiva del usuario como la del desarrollador.

Este enfoque facilita la evolución futura del sistema, el mantenimiento continuo y su integración con nuevas funcionalidades o servicios externos como notificaciones, firmas digitales o plataformas de pago.