

Planificación y Gestión del proyecto Alquiler de Viviendas

Realizado por:

- Daniel Salas Conejo
- Saul Corbelle Hernandez
- Laura Rodriguez Velasco

INDICE

1-Diseños del Sistema.....	4
1.1 Requisitos.....	4
1.1.1 Requisitos Funcionales.....	4
1.1.1.1 Requisitos de Usuario	4
1.1.1.2 Requisitos de Propiedades	4
1.1.1.3 Requisitos de Reservas	4
1.1.1.4 Requisitos de Pago	5
1.1.2 Requisitos no Funcionales	5
1.1.2.1 Requisitos de Rendimiento.....	5
1.1.2.2 Requisitos de Seguridad.....	5
1.1.2.3 Requisitos de Usabilidad.....	6
1.2-Arquitectura en 3 capas.....	6
1.2.1-Capa de Presentación (controladores webs)	6
1.2.2-Capa de Negocio (Lógica de la aplicación)	7
1.2.3-Capa de Persistencia (Datos)	7
1.3-Diseño de la Base de Datos.....	8
1.4-Diagrama de Clases	8
1.5-Diagrama de Secuencia.....	9
2-Metodología de Desarrollo Ágil.....	9
2.1-Estructura del flujo de trabajo	9
3-Metodología de Gestión de Proyectos.....	10

3.1-Roles del equipo	10
3.1.1 Product Owner	10
3.2.1 Scrum Master	10
3.2-Planificacion de Sprints	10
4-Gestion de la configuración	11
4.1 Control de Versiones	11
4.2 Entornos de Desarrollo	11
4.3 Gestión de Datos.....	12
4.4 Gestión de Dependencias	12
4.5 Control y Mantenimiento de la Configuración	12
4.6 Entorno de Ejecución: Java 25	12

1-Diseños del Sistema

1.1 Requisitos

1.1.1 Requisitos Funcionales

Los requisitos funcionales se pueden clasificar en cuatro grupos, requisitos de usuario, que son los que definen las operaciones que se deben realizar con los usuarios, de propiedades, que son aquellas operaciones que puede realizarse hacia las propiedades, de reservas, que son las operaciones que se realizan para la gestión de las reservas de propiedades, y de pago, que definen las operaciones de pago de las reservas.

1.1.1.1 Requisitos de Usuario

Los requisitos de usuario son los siguientes:

- ☐ **RF-01:** El sistema debe permitir el registro de nuevos usuarios como inquilinos o propietarios.
- ☐ **RF-02:** El sistema debe permitir el inicio de sesión para ambos tipos de usuarios.
- ☐ **RF-03:** El sistema debe permitir que inquilinos y propietarios gestionen su perfil (datos personales, medios de pago, etc.).
- ☐ **RF-04:** Un usuario no registrado debe poder buscar alojamientos sin necesidad de iniciar sesión.

1.1.1.2 Requisitos de Propiedades

Los requisitos sobre las propiedades son los siguientes:

- ☐ **RF-05:** El propietario, una vez autenticado, puede registrar una propiedad (vivienda completa o habitación individual).
- ☐ **RF-06:** El propietario puede definir si la propiedad admite reserva inmediata o si requiere confirmación manual.
- ☐ **RF-07:** El propietario puede confirmar o rechazar solicitudes de reserva recibidas.
- ☐ **RF-08:** El propietario puede confirmar la disponibilidad de una propiedad.
- ☐ **RF-09:** El sistema debe notificar al propietario cuando recibe una solicitud de reserva.

1.1.1.3 Requisitos de Reservas

Los requisitos de las reservas son las siguientes.

- ☐ **RF-10:** El inquilino autenticado puede **buscar alojamientos** según destino, fechas y tipo de inmueble.
- ☐ **RF-11:** El sistema debe ofrecer **filtros avanzados** de búsqueda, reserva inmediata, comodidades y política de cancelación.
- ☐ **RF-12:** El inquilino autenticado puede añadir alojamientos a su lista de deseos.

- ☐ **RF-13:** El inquilino autenticado puede **realizar una reserva** de un alojamiento.
- ☐ **RF-14:** Si el alojamiento permite reserva inmediata, el sistema debe permitir completar la reserva directamente (con pago incluido).
- ☐ **RF-15:** Si el alojamiento no permite reserva inmediata, el sistema debe enviar una solicitud de reserva al propietario.
- ☐ **RF-16:** El sistema debe notificar al inquilino si su reserva es confirmada o denegada.
- ☐ **RF-17:** En caso de denegación, el sistema debe reembolsar automáticamente el pago al inquilino.

1.1.1.4 Requisitos de Pago

Los requisitos del método de pago son:

- ☐ **RF-18:** El sistema debe permitir realizar pagos mediante tarjeta de crédito, tarjeta de débito o PayPal.

1.1.2 Requisitos no Funcionales

Los requisitos no funcionales se clasifican en diferentes grupos según el tipo de restricción o característica que aportan al sistema: de rendimiento, seguridad, usabilidad, fiabilidad, mantenibilidad, portabilidad, escalabilidad y compatibilidad.

1.1.2.1 Requisitos de Rendimiento

RNF-01: El sistema debe ser capaz de atender al menos 100 usuarios concurrentes sin afectar el rendimiento.

RNF-02: El tiempo de respuesta ante una búsqueda no debe superar los 3 segundos en condiciones normales de carga.

RNF-03: El proceso de reserva y pago debe completarse en menos de 10 segundos tras la validación.

1.1.2.2 Requisitos de Seguridad

RNF-04: Todas las comunicaciones entre el cliente y el servidor deben realizarse bajo el protocolo HTTPS.

RNF-05: Las contraseñas de los usuarios deben almacenarse cifradas mediante algoritmos seguros.

RNF-06: El sistema no almacenará información de pago sensible; las transacciones se realizarán a través de servicios externos certificados.

1.1.2.3 Requisitos de Usabilidad

RNF-08: La interfaz del sistema debe ser clara, intuitiva y coherente, permitiendo una fácil navegación entre las secciones.

RNF-09: El sistema debe ser accesible desde dispositivos móviles, tablets y ordenadores mediante un diseño responsivo.

RNF-10: Las acciones críticas (como eliminar una propiedad o cancelar una reserva) deben requerir confirmación del usuario.

1.1.2.5 Requisitos de Mantenibilidad

RNF-14: El sistema debe desarrollarse con una arquitectura en 3 capas(capas de presentación, negocio y persistencia).

RNF-15: El código debe cumplir estándares de programación Java y estar documentado mediante comentarios y Javadoc.

RNF-16: La base de datos debe diseñarse normalizada, facilitando su mantenimiento y posibles ampliaciones

1.1.2.6 Requisitos de Escalabilidad

RNF-17: El sistema debe permitir la incorporación de nuevas funcionalidades sin afectar las existentes.

RNF-18: La arquitectura debe soportar un aumento en el número de usuarios y propiedades sin comprometer el rendimiento.

1.2-Arquitectura en 3 capas

1.2.1-Capa de Presentación (*controladores webs*)

La Capa de Presentación es el punto de entrada de la aplicación.

Su función principal es gestionar la interacción entre el usuario y el sistema, recibiendo las solicitudes HTTP y retornando las vistas adecuadas.

1.2.2-Capa de Negocio (Lógica de la aplicación)

La Capa de Negocio representa el núcleo funcional del sistema, donde se concentra toda la lógica y las reglas del negocio.

Dentro de esta capa, se encuentran dos carpetas principales:

- Entidades: donde se definen las clases que representan los objetos del dominio del sistema (por ejemplo, Usuario, Producto, Pedido, etc.). Estas clases suelen estar anotadas con `@Entity` y mapeadas a tablas de la base de datos.
- Controladores o Gestores de Negocio: donde se encuentran las clases encargadas de aplicar la lógica del sistema y coordinar las operaciones con la capa de persistencia

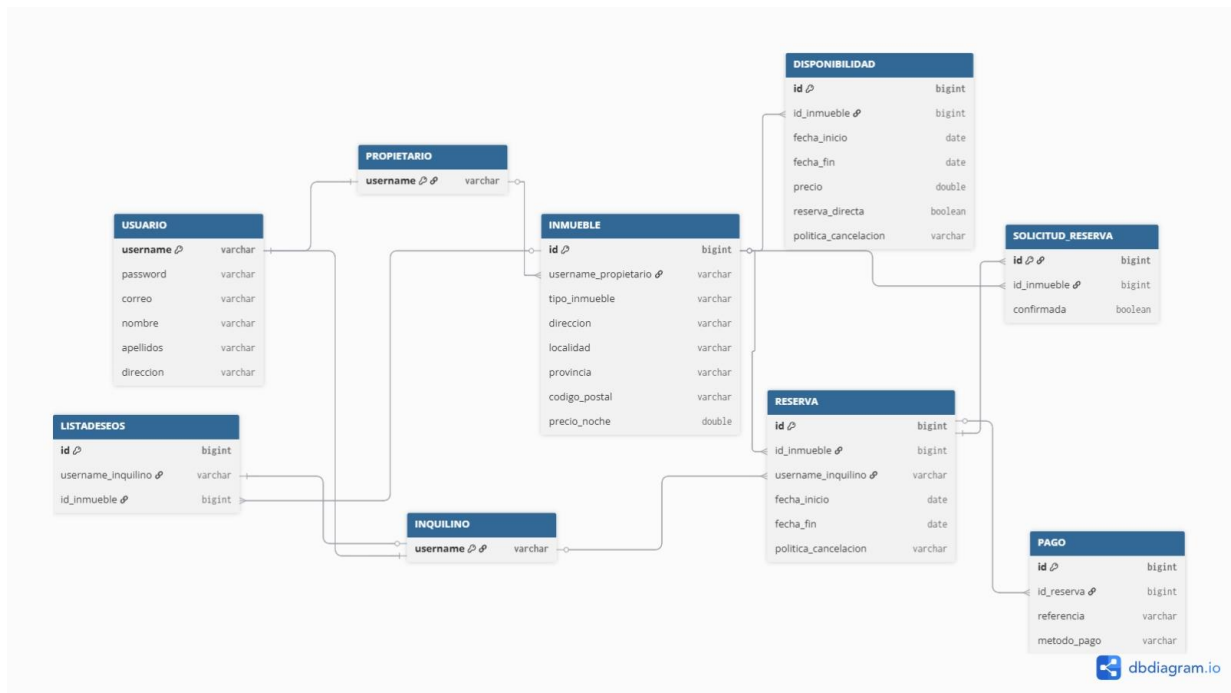
1.2.3-Capa de Persistencia (Datos)

La Capa de Persistencia es responsable de gestionar el acceso y almacenamiento de los datos.

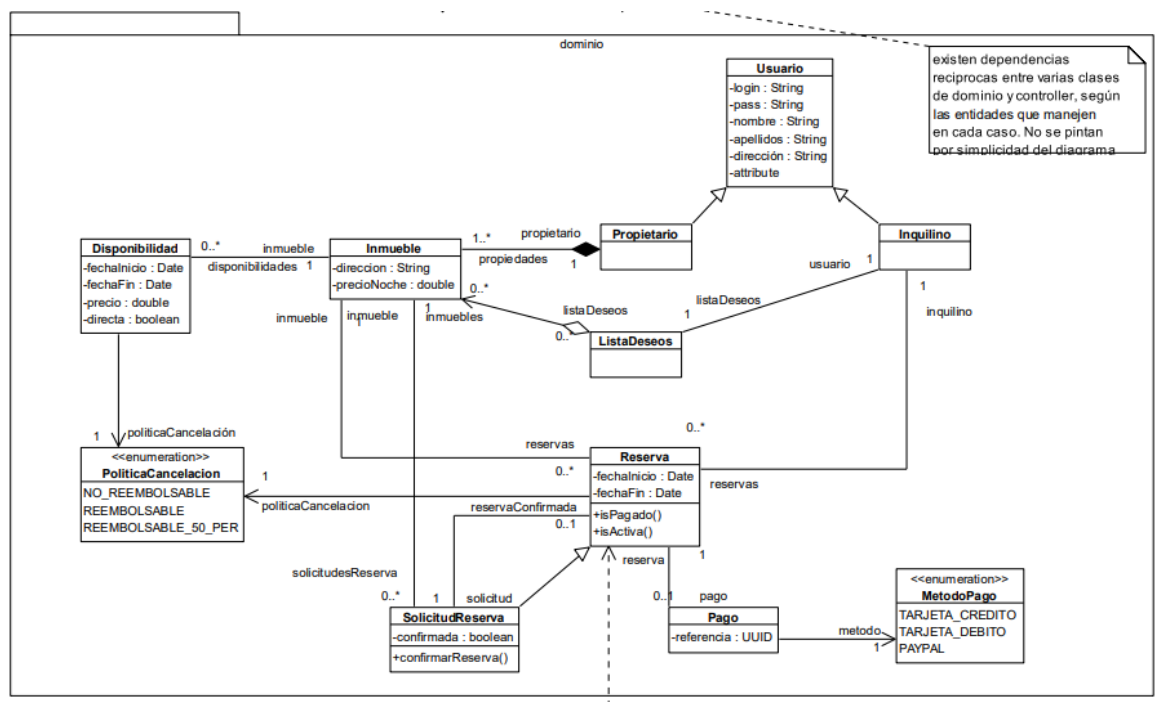
Su función es abstraer la comunicación con la base de datos, de modo que las demás capas no necesiten conocer detalles sobre el lenguaje SQL o las conexiones.

En este proyecto, la persistencia se implementa mediante el uso de Spring Data JPA, que permite interactuar con las entidades del modelo de datos a través de interfaces `@Repository`.

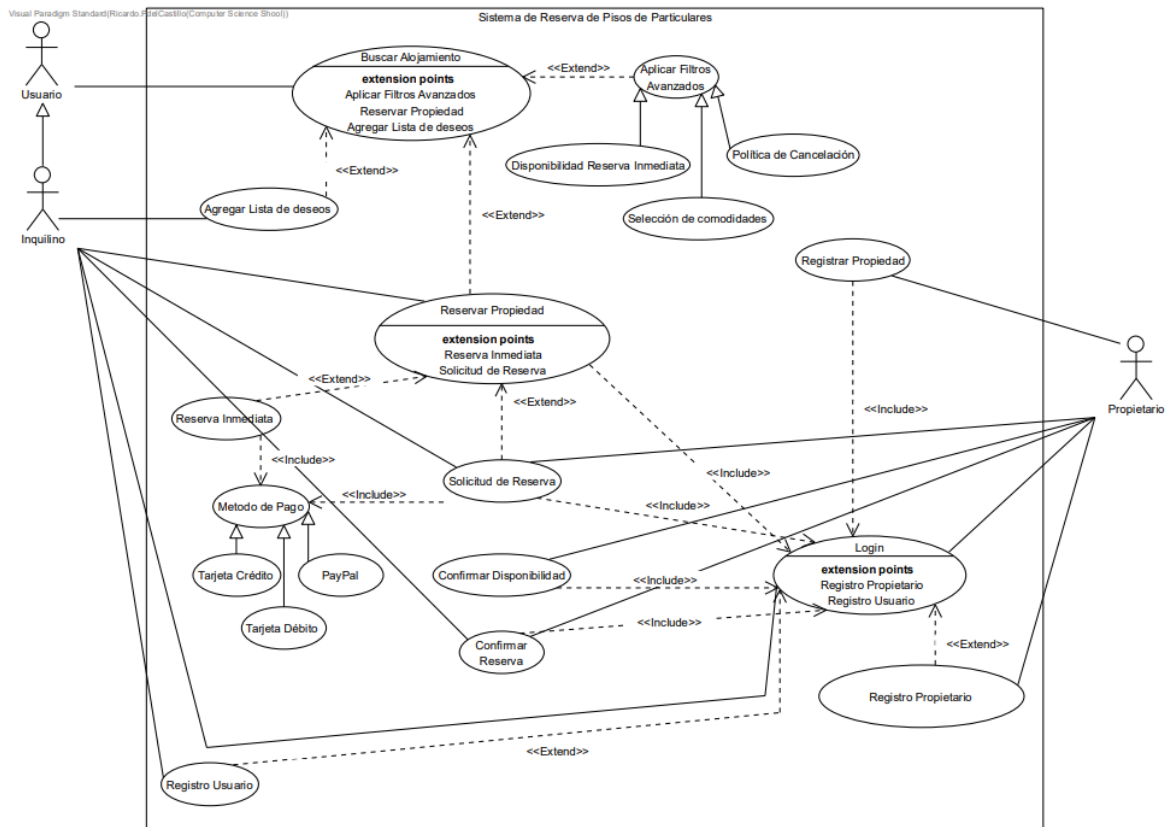
1.3-Diseño de la Base de Datos



1.4-Diagrama de Clases



1.5-Diagrama de Casos de Uso



2-Metodología de Desarrollo

Para el desarrollo del proyecto se ha optado por utilizar la metodología ágil Kanban, debido a su enfoque flexible y visual en la gestión del trabajo. Kanban permite **organizar y controlar las tareas del equipo** mediante un tablero dividido en columnas que representan los distintos estados del flujo de trabajo

2.1-Estructura del flujo de trabajo

El tablero Kanban se divide en las siguientes columnas:

- **Backlog** → tareas pendientes.
- **To Do** → tareas pendientes de iniciar en el sprint actual.
- **In Progress** → tareas en desarrollo.
- **Testing** → tareas completadas pendientes de revisión o pruebas.
- **Done** → tareas finalizadas e integradas en el repositorio principal.

Cada tarea incluye una breve descripción, y una etiqueta que identifica su categoría (backend, frontend, base de datos, documentación, etc.)

3-Metodología de Gestión de Proyectos

Para la gestión del proyecto se ha adoptado la **metodología ágil Scrum**, debido a su enfoque iterativo, colaborativo y orientado a la mejora continua. Esta metodología permite adaptarse a cambios de manera rápida y mantener una comunicación constante entre los miembros del equipo.

3.1-Roles del equipo

Cada rol clave del equipo tiene un responsable asignado, pero la toma de decisiones se realiza de manera colaborativa entre todos los miembros para garantizar una visión integral del proyecto.

3.1.1 Product Owner

Responsable: Daniel Salas

Funciones principales:

- Definir y comunicar la visión general del proyecto.
- Priorizar las funcionalidades y los objetivos del sistema según su valor para el negocio.
- Revisar y validar las historias de usuario antes de considerarlas completadas.
- Actuar como enlace entre el equipo de desarrollo y los stakeholders, recogiendo feedback constante.

3.2.1 Scrum Master

Responsable: Saul Corbelle

Funciones principales:

- Facilitar la comunicación y colaboración entre los miembros del equipo.
- Garantizar que el equipo siga las buenas prácticas de Scrum y Kanban.
- Identificar obstáculos o bloqueos que puedan afectar el progreso y ayudar a resolverlos.
- Supervisar el flujo de trabajo y la gestión del tablero Kanban para mantener un ritmo equilibrado.

3.2-Planificación de Sprints

El desarrollo se divide en **iteraciones quincenales** que agrupan tareas afines.

Cada sprint representa una evolución del producto, avanzando desde la base del sistema hasta la versión final.

Sprint	Contenidos principales
Sprint 1 – Inicialización del proyecto	Configuración del entorno de trabajo, creación del repositorio GitHub, configuración de Maven y Spring Boot, definición de estructura de paquetes, diseño inicial de la base de datos Derby.
Sprint 2 – Entidades base y Login	Implementación de entidades Usuario, Inquilino y Propietario, creación de DAOs y servicios, sistema de autenticación y registro.
Sprint 3 – Gestión de propiedades	Desarrollo de la funcionalidad para que los propietarios puedan dar de alta, modificar y eliminar propiedades; conexión con la base de datos.
Sprint 4 – Búsqueda y reservas	Implementación de la búsqueda de alojamientos con filtros; sistema de reservas con validaciones y control de disponibilidad.
Sprint 5 – Interfaz de usuario	Creación de vistas con Thymeleaf, navegación entre páginas, formularios, plantillas y estilos.
Sprint 6 – Integración, pruebas y documentación final	Pruebas integradas del sistema, revisión del código, corrección de errores, generación de documentación técnica y entrega final del proyecto.

4-Gestion de la configuración

4.1 Control de Versiones

Para el manejo del código fuente se ha utilizado Git (<https://git-scm.com/downloads>), una herramienta de control de versiones distribuido que permite registrar el historial de cambios, trabajar en diferentes ramas de desarrollo y facilitar la colaboración entre los integrantes del equipo.

El repositorio del proyecto se encuentra alojado en GitHub (<https://github.com/Danielol04/ISO2-LAB.git>), lo que permite acceder de manera remota, compartir el código, gestionar incidencias (issues), realizar revisiones de código mediante *pull requests* y mantener un flujo de integración continua. Gracias a GitHub, se garantiza la trazabilidad de todas las versiones del sistema y se evita la pérdida de información ante posibles errores o conflictos durante el desarrollo.

4.2 Entornos de Desarrollo

Para la implementación y despliegue de los diferentes módulos del sistema se han empleado dos entornos principales:

- **Eclipse IDE** (<https://www.eclipse.org/downloads/packages/>): utilizado principalmente para el desarrollo de componentes Java del sistema, integrando de forma nativa el uso de **Maven** y proporcionando herramientas de depuración, compilación y empaquetado del proyecto.
- **Visual Studio Code** (<https://code.visualstudio.com/download>): empleado como entorno complementario para la edición de código, scripts y archivos de configuración. Su flexibilidad y extensibilidad mediante plugins facilita el trabajo con diferentes lenguajes y tecnologías del proyecto.

4.3 Gestión de Datos

Como herramienta de administración de bases de datos se ha utilizado **DBeaver** (<https://dbeaver.io/download/>), un cliente universal que permite la conexión y gestión de distintos motores de bases de datos. DBeaver facilita la visualización de tablas, ejecución de consultas SQL y verificación de la integridad de los datos empleados durante las pruebas del sistema.

4.4 Gestión de Dependencias

Para la gestión de dependencias y la construcción del proyecto se ha empleado **Maven** (<https://maven.apache.org/>), una herramienta de automatización de compilaciones y empaquetado ampliamente utilizada en el entorno Java. Maven permite definir las bibliotecas y componentes requeridos en un archivo de configuración centralizado (*pom.xml*), garantizando la correcta instalación y compatibilidad de las versiones necesarias para la ejecución del sistema.

Además, Maven facilita la integración con repositorios remotos y contribuye a la reproducibilidad del entorno de desarrollo en distintos equipos.

4.5 Control y Mantenimiento de la Configuración

El proceso de gestión de configuración se complementa con prácticas de mantenimiento continuo, como la documentación de versiones, la verificación de dependencias actualizadas y la estandarización de los entornos de desarrollo. Estas medidas aseguran que el sistema pueda ser desplegado y mantenido de manera eficiente a lo largo del tiempo, reduciendo la posibilidad de errores y facilitando futuras ampliaciones o correcciones.

4.6 Entorno de Ejecución: Java 25

El sistema ha sido desarrollado y probado utilizando Java 25 (<https://www.oracle.com/es/java/technologies/downloads/>), la versión más reciente del lenguaje y entorno de ejecución de Java. Esta versión incorpora mejoras en

rendimiento, seguridad y soporte de nuevas características del lenguaje, lo que permite aprovechar las ventajas de un entorno moderno, estable y compatible con las herramientas de desarrollo empleadas.

El uso de Java 25 garantiza la compatibilidad con Maven, Eclipse y los demás componentes del proyecto, asegurando así un entorno unificado y actualizado para todo el ciclo de vida del software

