

```
var b = w.scrollTop();
var o = t.offset();
var x = o.left;
```

DOCUMENTACIÓN DEL PROYECTO

INGENIERÍA DEL SOFTWARE II

Sergio Mirado Tiemblo

Alejandro Arnanz Fernández

Azucena Fernández Bernal

Zaira Muñoz Hernández



```
//mark the element as visible
t.appeared = true;
```

```
//is this supposed to happen only once?
if (settings.one) {
```

```
    //remove the check
    w.unbind('scroll', check);
    var i = $.inArray(check, $.fn.appear.checks);
    if (i >= 0) $.fn.appear.checks.splice(i, 1);
}
```

```
//trigger the original fn
fn.apply(this, arguments);
```

```
the element
settings.data, modifiedFn);
```

Índice

1. Análisis y Definición del Sistema	3
1.1 Objetivo del proyecto	3
1.2 Actores	3
1.3 Requisitos funcionales.....	3
1.4 Requisitos no funcionales	4
1.5 Casos de uso principales	4
1.6 Reglas de negocio.....	5
1.7 Requisitos de datos	5
2. Metodología y Gestión Ágil (SCRUM)	6
2.1 Enfoque Scrum.....	6
2.2 Estructura del equipo y roles	6
2.3 Artefactos y herramientas	6
2.4 Eventos Scrum	6
2.5 Planificación	6
2.6 Control y seguimiento	7
2.7 Backlog y planificación de Sprints.....	7
3. Planificación y gestión del proyecto	8
3.1 Planificación temporal.....	8
3.2 Diagramas UML	9
3.3 Seguimiento y control de tareas.....	10
3.4 Métricas y revisión de Sprints	10
4 Gestión de la configuración.....	11
4.1 Sistema de Control de Versiones	11
4.2 Control de Peticiones de Cambio	11
4.3 Plan de Gestión de Configuración	11
4.5 Construcción y Distribución del Software.....	11
4.6 Gestión de versiones	12
4.7 Control de archivos en el repositorio	12

1. Análisis y Definición del Sistema

1.1 Objetivo del proyecto

El proyecto tiene como objetivo desarrollar un sistema similar a Airbnb, donde particulares puedan registrar propiedades para alquilar y otros usuarios puedan realizar reservas. Los usuarios deben registrarse previamente y pueden actuar como inquilinos o propietarios, mientras que el sistema también interactúa con una pasarela de pago y un sistema de notificaciones.

1.2 Actores

- **Visitante (no registrado):** puede buscar inmuebles y aplicar filtros básicos.
- **Usuario registrado:** puede autenticarse, mantener su perfil y actuar como inquilino o propietario.
- **Inquilino:** busca propiedades, añade inmuebles a su lista de deseos, realiza reservas y pagos, y puede aplicar filtros avanzados.
- **Propietario:** registra, modifica y elimina propiedades y gestiona solicitudes de reserva.
- **Pasarela de pago (externa):** procesa pagos y reembolsos mediante tarjeta de crédito, débito o PayPal.
- **Sistema de notificaciones (interno):** envía confirmaciones y avisos a los usuarios sobre reservas, pagos y solicitudes.

1.3 Requisitos funcionales

1. Registro y autenticación

El sistema permitirá registrar usuarios, diferenciando inquilinos y propietarios (un usuario puede ser ambos).

Se podrá iniciar y cerrar sesión, gestionando roles y permisos de cada usuario.

2. Gestión de propiedades

Los propietarios podrán registrar propiedades indicando tipo de inmueble, dirección, precio por noche, disponibilidad, comodidades, política de cancelación y tipo de reserva (inmediata o bajo solicitud).

Podrán modificar o eliminar las propiedades registradas.

3. Búsqueda y exploración

Cualquier usuario podrá buscar propiedades sin registro.

Los filtros básicos incluyen destino, fechas y tipo de inmueble.

Los filtros avanzados permiten buscar por reserva inmediata, comodidades específicas o políticas de cancelación.

Los usuarios registrados podrán guardar propiedades en su lista de deseos.

4. Reservas

Los inquilinos registrados podrán realizar reservas.

Si la propiedad permite reserva inmediata, la reserva se confirma tras el pago.

Si requiere solicitud, el inquilino envía una solicitud con pago retenido; el propietario la acepta o rechaza.

En caso de rechazo, se procesa automáticamente el reembolso y se notifica al inquilino.

Cada reserva se asocia a un pago único y cada solicitud puede generar una reserva confirmada si es aceptada.

5. Lista de deseos

Cada inquilino tiene una lista de deseos única.

Una lista puede contener varias propiedades y un inmueble puede estar en varias listas de deseos.

6. Pagos

Se integrará con una pasarela de pagos para validar transacciones antes de confirmar reservas.

Los reembolsos se procesarán automáticamente cuando corresponda.

7. Notificaciones

El sistema notificará sobre confirmación o rechazo de solicitudes de reserva, pagos realizados, reembolsos procesados y nuevas solicitudes recibidas por los propietarios.

1.4 Requisitos no funcionales

1. **Disponibilidad** 24/7 con tolerancia a fallos.
2. **Seguridad** mediante cifrado de contraseñas, HTTPS y protocolos seguros.
3. **Escalabilidad** para miles de usuarios concurrentes.
4. **Usabilidad** e interfaz intuitiva y accesible.
5. **Multiplataforma**: web y app móvil (iOS/Android).
6. **Tiempo de respuesta** óptimo.
7. **Mantenibilidad**: arquitectura modular y documentación clara.

1.5 Casos de uso principales

- Registro de usuarios (inquilino/propietario).
- Inicio y cierre de sesión.
- Registro, modificación y eliminación de propiedades.
- Búsqueda de propiedades (básica y avanzada).

- Agregar propiedades a la lista de deseos.
- Realizar reservas (inmediata o bajo solicitud).
- Realizar pagos.
- Aceptar o rechazar solicitudes de reserva.
- Procesar reembolsos.
- Enviar notificaciones a usuarios.

1.6 Reglas de negocio

- Solo los propietarios registrados pueden registrar propiedades.
- Cada inmueble pertenece a un único propietario; un propietario puede tener varias propiedades.
- No se podrán reservar fechas ya ocupadas.
- Una reserva debe estar asociada a una política de cancelación.
- Los pagos se completan antes de confirmar reservas o solicitudes.
- Los reembolsos se gestionan automáticamente.
- Los usuarios deben aceptar los términos y condiciones al registrarse.
- Las notificaciones a los propietarios son inmediatas ante nuevas solicitudes

1.7 Requisitos de datos

Usuarios: id, nombre, email, contraseña, rol, datos de pago; cada inquilino tiene una lista de deseos.

Propiedades: id, propietario, ubicación, tipo, precio, disponibilidad, comodidades, política de cancelación, tipo de reserva.

Reservas: id, propiedad, inquilino, fechas, estado (pendiente, confirmada, rechazada, cancelada).

Pagos: id, reserva, monto, método, estado (exitoso, rechazado, reembolsado).

Lista de deseos: id usuario, id propiedad.

Notificaciones: id, destinatario, tipo (reserva, pago, reembolso), estado (pendiente, enviada).

Solicitudes de reserva: id, inmueble, inquilino, estado, reserva asociada si fue aceptada.

Disponibilidad: id, inmueble, rango de fechas.

2. Metodología y Gestión Ágil (SCRUM)

2.1 Enfoque Scrum

El desarrollo del sistema se gestionará mediante la metodología ágil **Scrum**, que promueve la entrega incremental del producto, la colaboración continua y la adaptación al cambio. El equipo trabajará en iteraciones cortas denominadas **Sprints**, con entregables funcionales y revisiones periódicas.

2.2 Estructura del equipo y roles

- **Scrum Master:** este rol lo desempeña Azucena Fernández, facilita las reuniones, elimina impedimentos y asegura la correcta aplicación de Scrum.
- **Product Owner:** define las prioridades, gestiona el Product Backlog y vela por el valor del producto.
- **Equipo de desarrollo (Developers):** formado por Alejandro Arnanz, Sergio Mirado y Zaira Muñoz, diseña, implementa, prueba y documenta las funcionalidades. Cada miembro asume responsabilidades específicas según su área (frontend, backend, documentación o testing), pero colaborando de forma transversal.

2.3 Artefactos y herramientas

Los principales artefactos Scrum empleados serán:

- **Product Backlog:** lista priorizada de funcionalidades y tareas.
 - **Sprint Backlog:** conjunto de ítems seleccionados para cada Sprint.
 - **Incremento:** versión funcional del sistema entregada tras cada Sprint.
- Para la gestión se utilizará **GitHub Projects** (tablero Kanban) junto con **Issues** y **Labels** para el control de tareas y seguimiento del avance.

2.4 Eventos Scrum

- **Sprint Planning:** definición de objetivos y tareas del Sprint.
- **Daily Scrum:** breve reunión diaria de seguimiento (puede realizarse asincrónicamente).
- **Sprint Review:** demostración del incremento y revisión de objetivos.
- **Sprint Retrospective:** análisis de mejora continua del proceso.

2.5 Planificación

- El proyecto se desarrollará a lo largo de **X semanas** distribuidas en **n Sprints** de **X días** cada uno.

- Cada Sprint culminará con un incremento funcional verificable
- El **calendario de entregas** incluirá revisiones parciales y una entrega final.

2.6 Control y seguimiento

El control de tareas se realizará mediante **Issues** en GitHub, categorizados por **Labels** que indiquen su tipo y estado (por ejemplo: feature, bug, documentation, in progress, done). Se empleará el **tablero de proyectos (Kanban)** para visualizar el flujo de trabajo y métricas de avance (burn-down chart), revisando en cada Sprint la carga de trabajo completada frente a la planificada.

2.7 Backlog y planificación de Sprints

El **Product Backlog** se construirá a partir de los requisitos funcionales del sistema y se refinará progresivamente.

En cada **Sprint Planning**, el equipo seleccionará los elementos más prioritarios del backlog y los convertirá en **issues detallados**, asignando responsables y estimaciones de esfuerzo. Los Sprints incluirán: desarrollo, revisión de código, testing y documentación.

3. Planificación y gestión del proyecto

3.1 Planificación temporal

Los Sprints del proyecto se han planificado para garantizar la entrega incremental y funcional del sistema. La organización ha sido la siguiente:

- **Sprint 1 (1,5 semanas):**

Durante este Sprint se completaron los issues correspondientes al desarrollo inicial, incluyendo:

- Formulario para el registro de usuarios.
- Creación de entidades JPA.
- Implementación de los DAOs.
- Diseño de la base de datos.
- Configuración de la base de datos Derby.
- Configuración de Maven y gestión de dependencias.

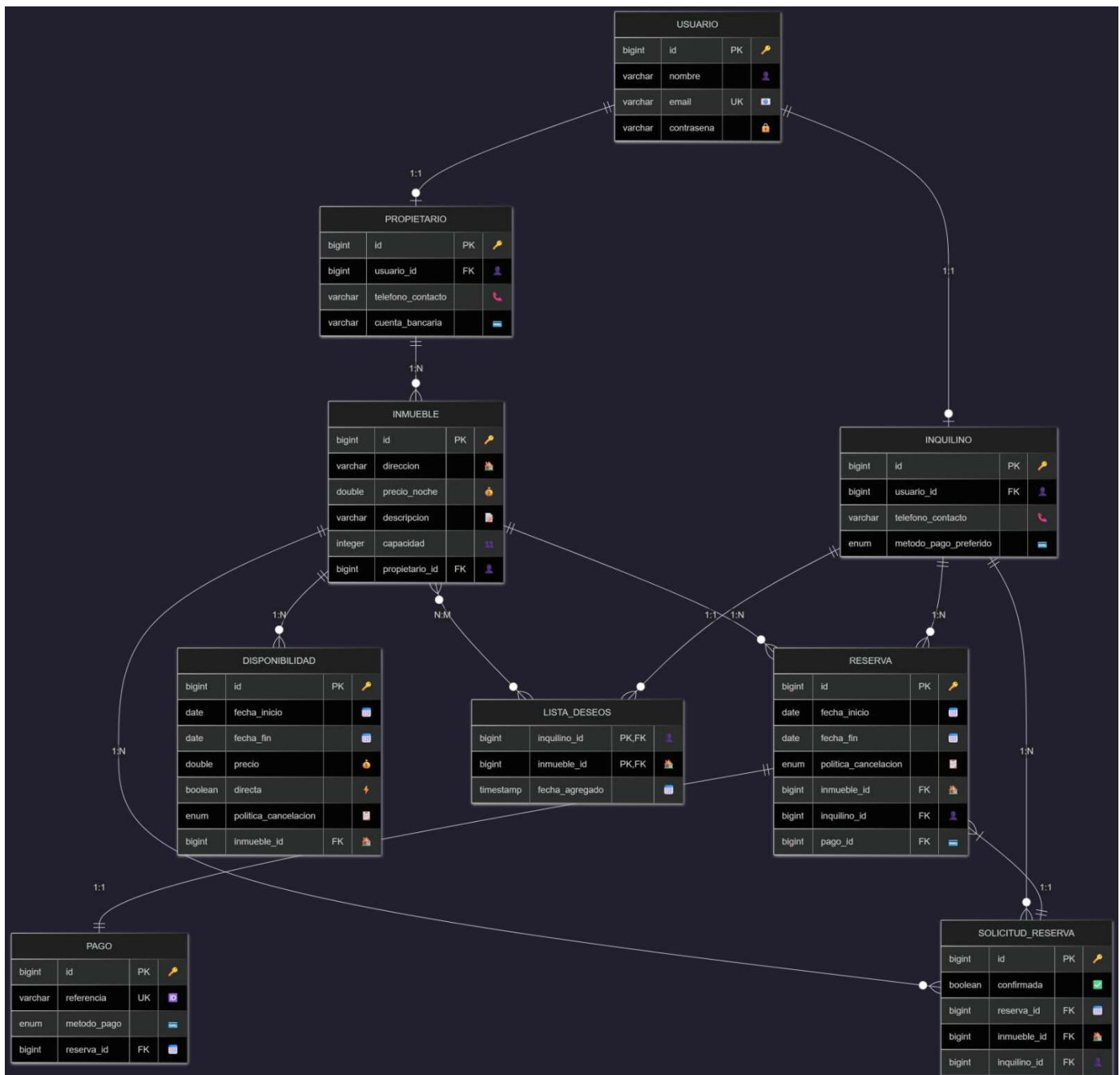
- **Sprint 2 (más de 2 semanas):**

En este Sprint se finalizaron los siguientes issues:

- Detalle de inmueble.
- Registro de propiedades.
- Creación del catálogo de alojamientos.
- Implementación de la página de inicio.
- Login de usuarios.
- Gestión de errores.
- Diseño de la interfaz web.
- Eliminación de la carpeta database del proyecto para mejorar la organización y control de versiones.

3.2 Diagramas UML

- Relaciones de la base de datos



3.3 Seguimiento y control de tareas

- **Issues de GitHub:** cada historia de usuario se convierte en un issue, etiquetado según tipo (frontend, backend, database...).
- **Kanban board:** seguimiento de tareas en columnas (Product backlog, sprint backlog...).
- **Burn-down chart:** gráfica de progreso de cada Sprint (historias completadas vs. tiempo).

3.4 Métricas y revisión de Sprints

- Evaluación de avance mediante porcentaje de historias completadas.
- Revisión de objetivos en Sprint Review, ajustes en el Backlog según prioridades.
- Retrospectiva para identificar mejoras en el proceso y corregir desviaciones.

4 Gestión de la configuración

4.1 Sistema de Control de Versiones

- **Sistema utilizado:** Git
- **Repositorio:** GitHub
- **Convenciones básicas:**
 - Rama principal: main
 - Rama de desarrollo: development
 - Ramas de funcionalidades: feature/*
 - Ramas de corrección de errores: hotfix/*
- **Etiquetado de releases:** versión semántica vX.Y.Z

4.2 Control de Peticiones de Cambio

- Cada cambio importante se registra en **GitHub Issues**, indicando:
 - Tipo: feature, bug, documentation...
 - Estado: open, in progress, revision, done
 - Descripción del cambio
- Las peticiones de cambio se priorizan y asignan a miembros del equipo.

4.3 Plan de Gestión de Configuración

- **Elementos de configuración:**
 - Código fuente (src/main/java)
 - Plantillas y recursos (src/main/resources/templates, application.properties)
 - Archivos de build (pom.xml)
- **Procedimiento de cambio:**
 1. Crear una rama nueva (feature/* o bugfix/*)
 2. Implementar el cambio
 3. Abrir Pull Request para revisión
 4. Mergear en develop tras aprobación
 5. Mergear en main solo si pasa pruebas y build

4.5 Construcción y Distribución del Software

- **Herramienta:** Maven
- **Comandos básicos:**
 - mvn clean install → build local
 - mvn spring-boot:run → ejecutar la aplicación
- **Distribución:** se entrega la versión compilada (target/) y el repositorio actualizado.

4.6 Gestión de versiones

El equipo de desarrollo utilizará única y exclusivamente las siguientes versiones:

- **Java:** 21
- **Apache Derby:** 10.16.1.1
- **Apache Maven:** 3.9.11
- **Spring Boot:** 3.5.6

Se recomienda mantener estas versiones consistentes en todos los entornos de desarrollo para asegurar compatibilidad y reproducibilidad del proyecto.

4.7 Control de archivos en el repositorio (gitignore)

Con el fin de garantizar la integridad y portabilidad del proyecto, así como evitar la inclusión de archivos temporales, binarios o específicos del entorno de desarrollo, se ha definido una política de exclusión para Git mediante el archivo **.gitignore**. Esto permite mantener el repositorio limpio y asegurar que solo el código fuente y los recursos necesarios para la construcción y ejecución del proyecto sean compartidos.

- **Archivos compilados y paquetes**
Se excluyen los archivos generados durante la compilación y empaquetado, como binarios y paquetes, evitando subir artefactos que se pueden reconstruir.
- **Archivos de registro**
Se excluyen los archivos de log generados durante la ejecución de la aplicación, incluyendo logs de frameworks específicos como Spring Boot.
- **Directorios de construcción y dependencias**
Se omiten los directorios de construcción y dependencias generados por herramientas de gestión de proyectos como Maven o Gradle, evitando conflictos entre entornos.
- **Archivos y configuraciones de entornos de desarrollo**
Se excluyen los archivos y carpetas generados automáticamente por los distintos IDEs (Eclipse, IntelliJ IDEA, VS Code) que no son necesarios para la construcción del proyecto ni para su colaboración.
- **Archivos temporales del sistema operativo**
Se evitan los archivos generados automáticamente por el sistema operativo, que no aportan información relevante al proyecto.
- **Archivos de entorno local**
Se excluyen las configuraciones locales y variables de entorno que pueden contener información sensible o específica de cada desarrollador.
- **Otros archivos específicos del proyecto**
Se omiten otros archivos generados por el proyecto, como bases de datos embebidas o directorios temporales, que no deben subirse al repositorio.