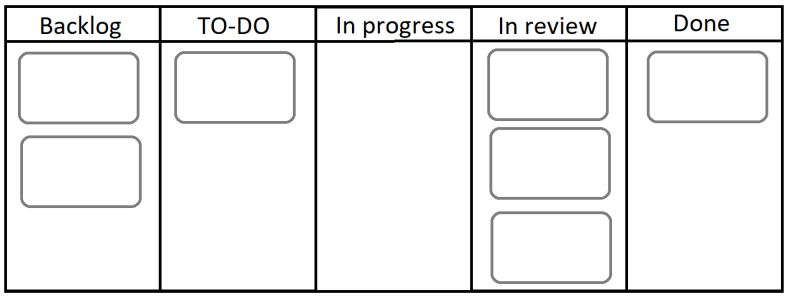
# Anteproyecto Jira-clone:
# Sistema de tickets y kanban

---

## 1. Introducción

**Problema:** En el desarrollo de software (profesional y no profesional) se necesita, de alguna forma, el poder controlar el trabajo de todo el equipo, tanto para no pisarse entre colegas como para poder demostrar el trabajo realizado día a día. Por esto mismo, se requiere realizar un app que tenga un sistema de tickets en un kanban.
Ejemplo:




Siendo cada tarjeta gris un ticket y el tablero un tablero kanban de 5 estados: Backlog, TO-DO, In progress, In review y Done.

---

## 2. Alcance y Limitaciones

- **Funcionalidades:**
    - Los usuarios tendrán 3 roles: Máster (podrá usar el CRUD de usuarios, el CRUD de proyectos y el CRUD de tareas), Leader (podrá usar el CRUD de proyectos, además del de tareas) y Developer (podrá usar el CRUD de tareas).
    - Se podrá navegar por la aplicación y modificar tus propios tickets, modificar los proyectos o directamente los usuarios (depende tu rol).
    - Será responsive, por lo que se podrá usar en la mayoría de dispositivos.
- **Plataformas:** Web.
- **Limitaciones:** En una primera versión, se hará el backend.

---

## 3. Análisis de Viabilidad

- **Recursos:**
    - Herramientas: VsCode, JetBrains, Docker, Postman y MySQL.
- **Riesgos:**
    - El deploy de Java suele ser costoso y complicado.

---

## 4. Metodología

- **Metodología Ágil:** Scrum para un desarrollo iterativo y flexible.
- **Tecnologías:**
    - **Java:** Como lenguaje de programación del backend.
    - **Spring:** Framework para desarrollo backend de Java (SpringBoot, SpringSecurity, Hibernate, JPA, etc).
    - **HTML, CSS y Javascript:** Paquete frontend básico.
    - **Typescript:** Para mejorar la robustez del frontend con Js.
    - **Tailwind:** Framework de frontend para CSS.
    - **React:** Framework de frontend para Js.
    - **SQL:** Se utilizará como DDBB.
    - **Nube**: a revisar si Aws, Azure o Cloud.

---

## 5. Arquitectura

- **Cliente:** Pagina web.
- **Servidor:** A decidir, primeramente se hará en local (posiblemente con la URL tuneleada), luego quizá se deployé en una nube.
- **API:** Son 3, users, projects, tasks.

---

## 6. Cronograma

- **Fase 1 (1 semana):** Debería estar listo el diseño de las 3 API's (users, projects, tasks) con sus 3 diseños de POO, relaciones, DDBB, etc.
- **Fase 2 (2 semanas):** Deberían estar programadas las 3 API's y probadas con Postman.
- **Fase 3 (2 semanas):** Se le aplica SpringSecurity para ya poder tener el sistema de roles.
- **Fase 4 (2 semanas):** Debería estar listos los estilos de la página, sin Js.
- **Fase 5 (2 semanas):** Se debería haber agregado la interactividad.
- **Fase 6 (1 semana):** Se realiza el deploy (o al menos el tuneleo de la url) para poder probarlo.

---

## 7. Equipo de Trabajo

- **lautajam** - Desarrollador Backend
- **lautajam** - Desarrollador frontend
- **lautajam** - DevOps

---