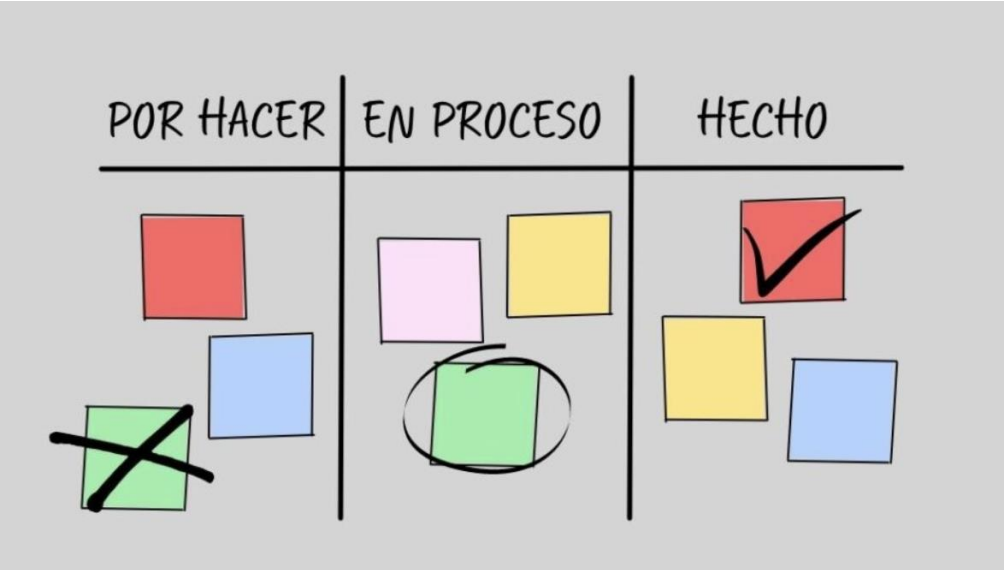
Sistema Kanban para la gestión de proyectos

Introducción

El reto consiste en desarrollar un sistema **Kanban** basado en la web para la gestión de proyectos. El sistema debe permitir a los usuarios **gestionar proyectos**, **organizar tareas** en diferentes etapas y realizar **seguimiento del progreso de las tareas**.

Si aún no sabes que es un tablero **Kanban** puedes leer esta guia, donde encontrarás los conceptos principales : https://kanbanize.com/es/recursos-de-kanban/primeros-pasos/que-es-kanban

En términos simples Un **tablero Kanban** es una herramienta visual que se utiliza para organizar y gestionar el flujo de trabajo de un equipo o proyecto. Consiste en una serie de columnas que representan diferentes etapas del proceso y tarjetas que representan tareas o elementos de trabajo. Cada tarjeta se mueve a través de las columnas conforme avanza en su progreso, desde la etapa inicial hasta su finalización.



Objetivo

Tu objetivo es desarrollar una **API REST** usando **Springboot** y **Java**, que cumpla con los features solicitados a continuación:

Features

Como desarrollador **backend** debes exponer un **API Rest** que permita las siguientes funcionalidades:

Requerimientos funcionales

1. Gestionar proyectos
2. Gestionar tareas
3. Seguimiento de tareas
4. Implementar Autenticación en el API REST

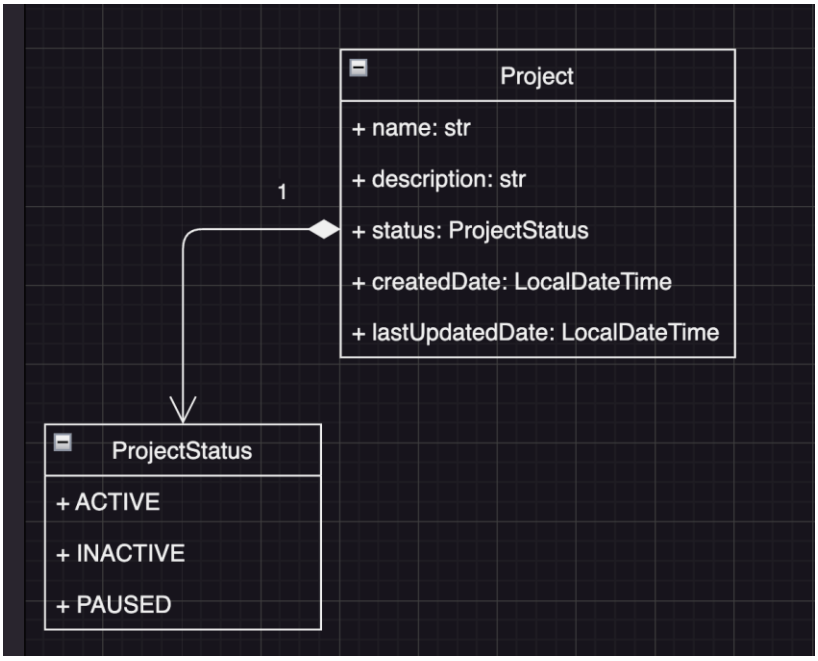
Requerimientos no funcionales

1. Documentar el API

2. Gestionar proyectos

1.1 Definir modelo

Los proyectos deben estar definidos por el concepto **Project**, a continuación una descripción:



1.2 Endpoints CRUD

Debes exponer los endpoints necesarios que permitan las operaciones CRUD sobre el concepto de dominio **Project.**

```
POST      -> /v1/projects       // crear  un Project
PUT       -> /v1/projects/{id}  // editar un Project
DELETE    -> /v1/projects/{id}  // eliminar un Project
GET       -> /v1/projects/{id}  // obtener un Project por id
```

Notas:

- al crear un proyecto se debe asignar un identificador único, se recomienda usar un **UUID**
- el endpoint ***POST*** *-> /v1/projects* solo debe recibir los siguientes datos:

```
{
    “name”: “project #1”,                    // requerido
    “description”: “project #1 description”  // no requerido
}
```

- al crear un proyecto el **status** por defecto debe ser **ACTIVE**

1.3 Endpoint para consultar proyectos

Adicional debes crear un endpoint que permita la consulta de todos los proyectos pero de manera paginada. para la estructura de paginación se puede usar la siguiente:

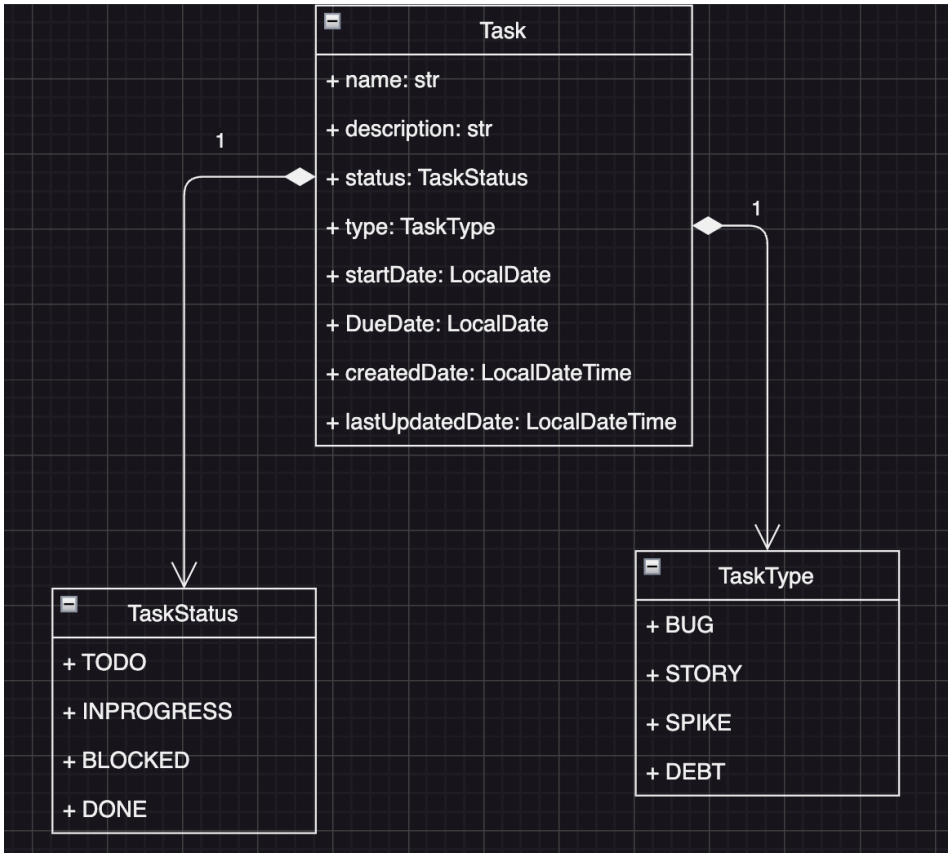
```
{
  total_elements: 10
  page: 1,
  content: [
    {
      "name": "project #1",
      "description": "project #1 description"
      "status": "active",
      "createdDate": "2023-08-15T12:45:00",
    },
    {
      …
    }
  ]

}
```

3. Gestionar Tareas

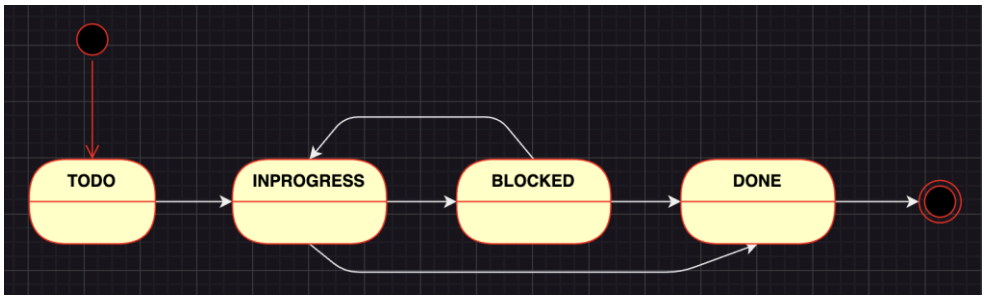
2.1 Definir modelo

Las tareas están definidas por el modelo **Task,** a continuación descrito:



El flujo de estados de un **Task** está definido como se muestra a continuación.

estado inicial: **TODO**
estados intermedios: **INPROGRESS, BLOCKED**
estado final: **DONE**



2.3 Exponer endpoints CRUD

Debes exponer los endpoints necesarios para lograr un CRUD sobre el concepto de dominio **TASK,** como se describe a continuación:

```
DELETE  -> /v1/tasks/{id}  // eliminar un Task
GET     -> /v1/tasks/{id}  // obtener  un Task
```

2.3.1 Crear un task

Una tarea (**Task**) sólo puede ser creada dentro de un proyecto(**Project**), por ende no se pueden crear tareas que no pertenezcan a un proyecto.

POST */v1/projects/{id}/tasks*

Notas:

- Para crear una tarea todos los datos son requeridos, a excepción de **createdDate** y **lastUpdatedDate**(estos dos datos debe crearlos el sistema)

4. Seguimiento de tareas

3.1 Crear endpoint para actualizar estado de una tarea

Para actualizar el estado de una tarea se debe crear el siguiente endpoint:

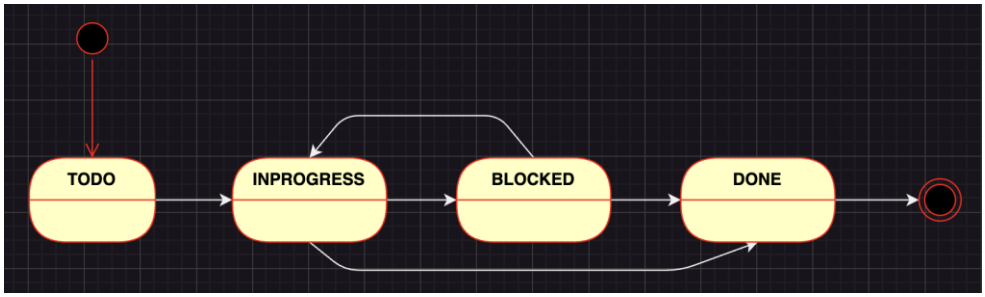
endpoint: *PATCH /v1/task/{id}*
request:

```
{
    “status”: ( TODO | INPROGRESS | BLOCKED | DONE )
}
```

Notas:

- si se envía un estado no válido, se espera que el backend retorna un mensaje de error: **`{ “mensaje”: “el estado {estado} no es válido” }`**

- el cambio de estado de un **Task** debe seguir el flujo de estados definido:



por lo que si se hace el cambio de un estado a un estado inválido de acuerdo al flujo se espera un mensaje de error como el siguiente:

`{ "mensaje": "no es posible asignar al estado {estado} una tarea con estado {estado_actual}" }`

3.2 Crear endpoint para obtener tablero

Se debe crear un endpoint que permita consultar la todas las tareas de un proyecto en forma de tablero, esto se logra agrupando las tareas de acuerdo a su estado. se espera que el endpoint siga la siguiente especificación:

endpoint: ***GET*** */v1/projects/{id}/board*
respuesta:

```
{
    "project": {
        "id": "01fbd470-9f2d-426a-a10f-3a0a2196a5d3",
        "name" : "project #1"
     },
    "board": [
       { "status": "TODO", tasks: [ tareas en estado TODO ] },
       { "status": "INPROGRESS", tasks: [ tareas en estado TODO ] },
       { "status": "BLOCKED", tasks: [ tareas en estado TODO ] },
       { "status": "DONE", tasks: [ tareas en estado TODO ] },
     ]
}
```

3.3 Crear endpoint para consultar tareas por fecha de vencimiento

Para darle seguimiento a las tareas de un proyecto es necesario consultar la tareas que están vencidas con el propósito de ver qué está pasando con dichas tareas por lo que es necesario crear un endpoint que retorne las tareas vencidas de un proyecto.

endpoint: GET /v1/projects/{id}/due-task
respuesta: debe retornar la lista de tareas vencidas al día actual.