

# Práctica 03: Aplicación de la Metodología XP en un Proyecto Real con Spring Boot

Dylan Jhossue Granizo Ayala<sup>[0000-1111-2222-3333]</sup>

<sup>1</sup> Escuela Politécnica Nacional,

<sup>2</sup> Quito-Ecuador

dylan.granizo@epn.edu.ec

**Abstract.** En esta práctica se aplicaron los principios fundamentales de la metodología ágil Extreme Programming (XP) para mejorar de manera iterativa la aplicación TodoList. Se añadieron nuevas funcionalidades dinámicas, mejoras visuales significativas y se refactorizó la estructura técnica del sistema. El desarrollo se organizó mediante issues en GitHub y se ejecutaron pruebas automatizadas para validar los cambios. Todo el proceso fue guiado por la filosofía de desarrollo continuo, modularidad y enfoque centrado en el usuario.

**Keywords:** Extreme Programming, XP, Spring Boot, TodoList, AJAX, desarrollo ágil, interfaz de usuario, pruebas automatizadas, REST API, DTO, Thymeleaf.

## 1 Objetivos

- Implementar mejoras funcionales y visuales en la aplicación TodoList.
- Aplicar los principios de XP: iteración rápida, feedback continuo, diseño limpio y pruebas automatizadas.
- Utilizar Spring Boot y tecnologías web modernas para desarrollar una aplicación más completa y mantenible.
- Documentar el progreso y dividir tareas mediante GitHub Projects e issues.

## 2 Introducción

Extreme Programming (XP) es una metodología ágil que promueve el desarrollo eficiente de software mediante la simplicidad, comunicación, retroalimentación constante, y coraje para realizar cambios. Esta práctica busca poner en acción estos principios al evolucionar una aplicación de tareas existente. A través de funcionalidades dinámicas, pruebas, refactorización y diseño centrado en el usuario, se logró transformar una herramienta básica en una plataforma rica y profesional.

### 3 Desarrollo

#### 📌 Planificación y Descomposición

El proyecto se organizó por funcionalidades clave, cada una asociada a un **issue** en **GitHub**. Se estableció un flujo de trabajo tipo kanban: ToDo → In Progress → Done.

#### 🔧 Funcionalidades Dinámicas Agregadas

- Marcar tareas como completadas sin recargar (AJAX).
- Edición en línea de títulos, prioridad y fecha límite.
- Eliminación de tareas con confirmación modal.
- Filtros activos por título, prioridad y fecha.
- Opcion de lista de usuarios registrados que recupera su descripción foto de perfil, id , email.

#### 🎨 Mejoras Visuales

- Diseño responsive con Bootstrap 5.
- Rediseño completo de tarjetas, botones y sidebar.
- Animaciones y estilos visuales al completar tareas.
- Badges de urgencia si la fecha está cercana.
- Editor WYSIWYG (Quill.js) para descripción enriquecida.
- Uso de emojis e íconos temáticos para accesibilidad visual.

#### 👤 Perfil de Usuario

- Avatar simbólico redondo con iniciales.
- Edición del nombre visible y foto desde interfaz.
- Saludo personalizado en dashboard.

#### ✳️ Backend estructurado con XP

- Separación en Controller, RestController, Service, y Repository.
- Implementación de DTOs para desacoplar lógica.
- Endpoints REST con @PutMapping, @PostMapping para AJAX.

#### 🧪 Pruebas Automatizadas

- Refactorización y adaptación de UsuarioServiceTest y TareaWebTest.
- Validación de nuevos flujos AJAX.
- Prevención de errores null en Thymeleaf.

## Comprensión y descomposición del sistema

El punto de partida fue una aplicación simple que permitía a los usuarios gestionar tareas. Se descompuso el sistema en unidades funcionales pequeñas, facilitando su desarrollo incremental.

### Estructura técnica

- **Arquitectura por capas:** Separación en Controller, RestController, Service y Repository.
- **Uso de DTOs:** Para aislar la lógica del modelo de la presentación.
- **Control de acceso:** Autenticación por sesión con HttpSession y ManagerUserSession.

### Pruebas automatizadas

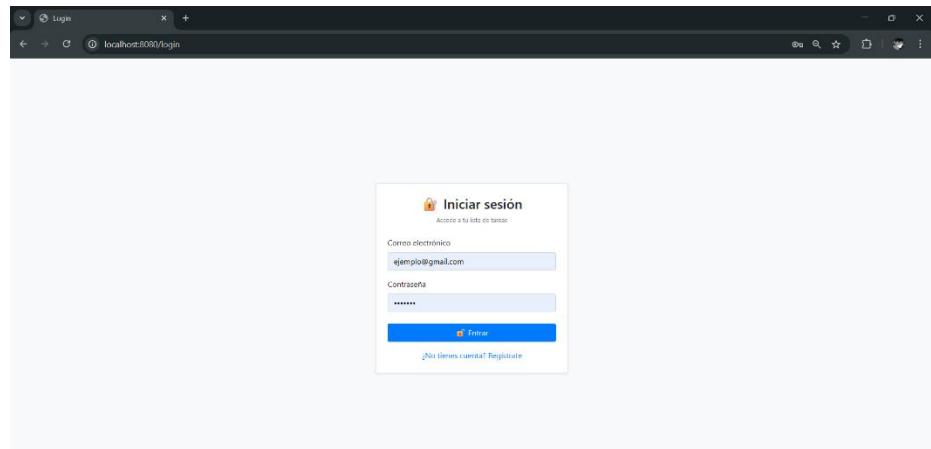
Se adaptaron y refactorizaron los tests existentes (TareaWebTest, UsuarioServiceTest) para las nuevas rutas y lógica dinámica.

## 4 Resultados

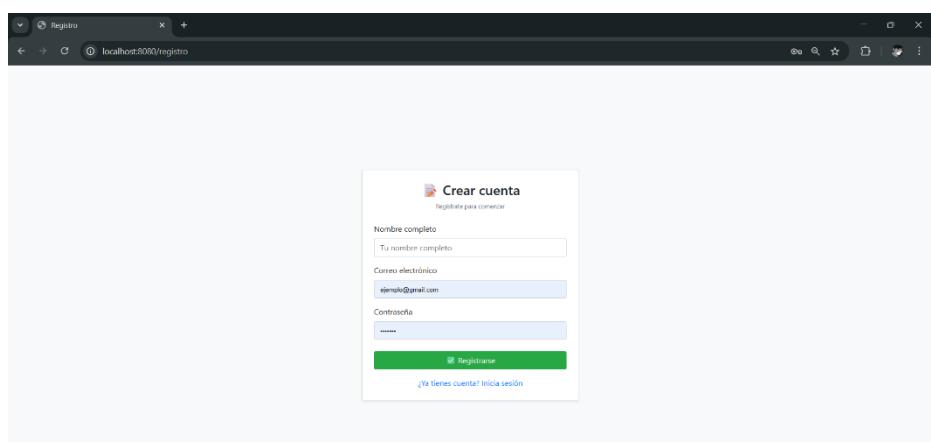
Se alcanzó una aplicación más moderna, interactiva, clara y mantenible. La estructura basada en XP permitió avanzar por ciclos, agregando funcionalidad sin comprometer la calidad del código. Se gestionaron múltiples issues desde GitHub Projects, reflejando una ejecución ágil y profesional del proyecto.

## 5 Anexos:

Inicio de sesión modificado (Es importante destacar que el inicio y registro de sesión da alerta de si el usuario existe o no con alertas y la restricción de campos obligatorios en el registro)

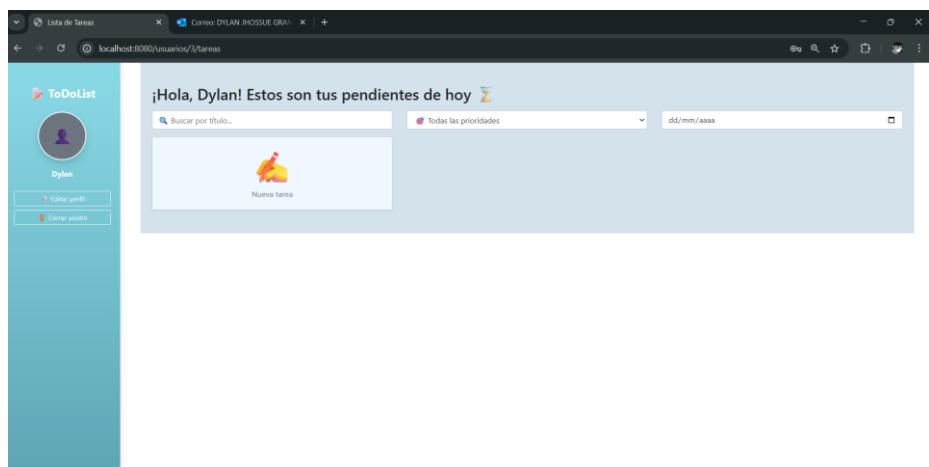


Registro modificado:

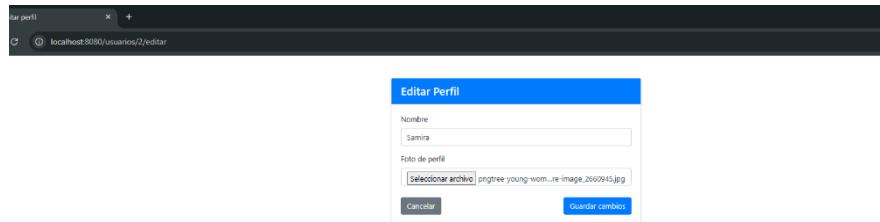


The screenshot shows a web browser window with the title bar 'Registro' and the URL 'localhost:8080/registro'. The main content is a registration form titled 'Crear cuenta' with the sub-instruction 'Regístrate para comenzar'. The form fields are: 'Nombre completo' (placeholder 'Tu nombre completo'), 'Correo electrónico' (placeholder 'ejemplo@gmail.com'), and 'Contraseña' (placeholder '\*\*\*\*\*'). Below the form is a green 'Registrarse' button with a checked checkbox and a link '¿Ya tienes cuenta? Inicia sesión'.

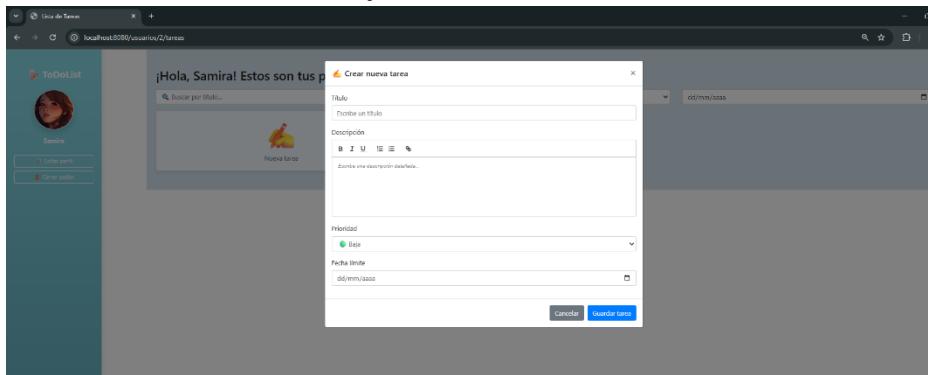
Página Principal de TODO-LIST



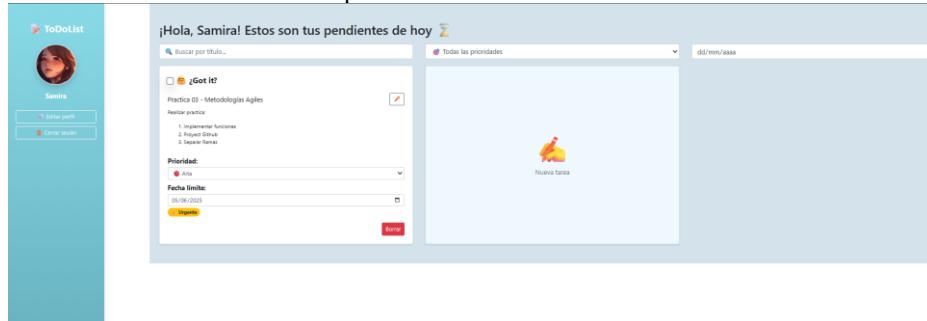
### Edición de perfil



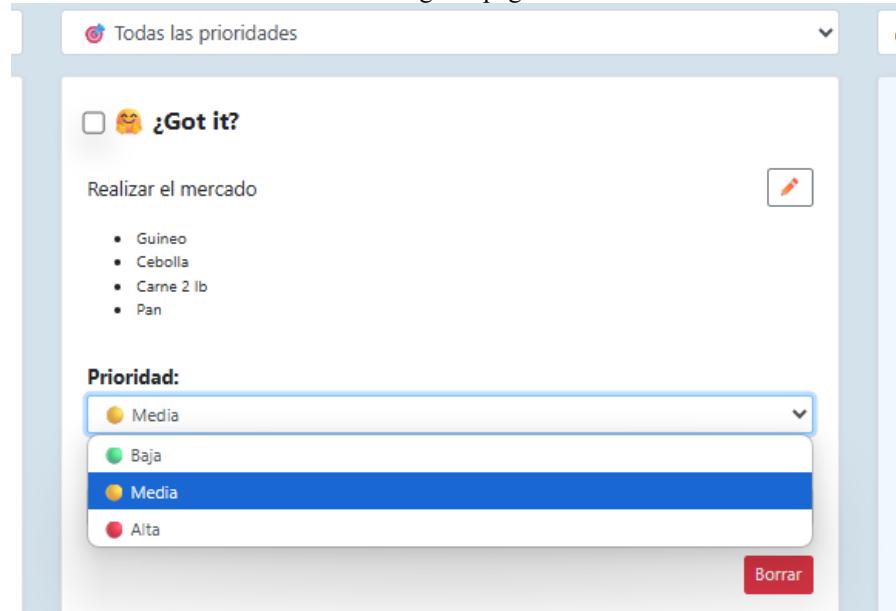
### Creación de nueva Tarea con tarjeta modal:



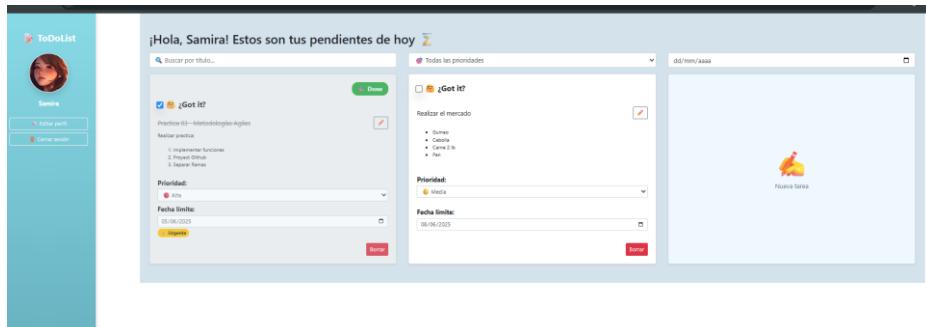
### Vista de Tareas si vence en los próximos 3 días:



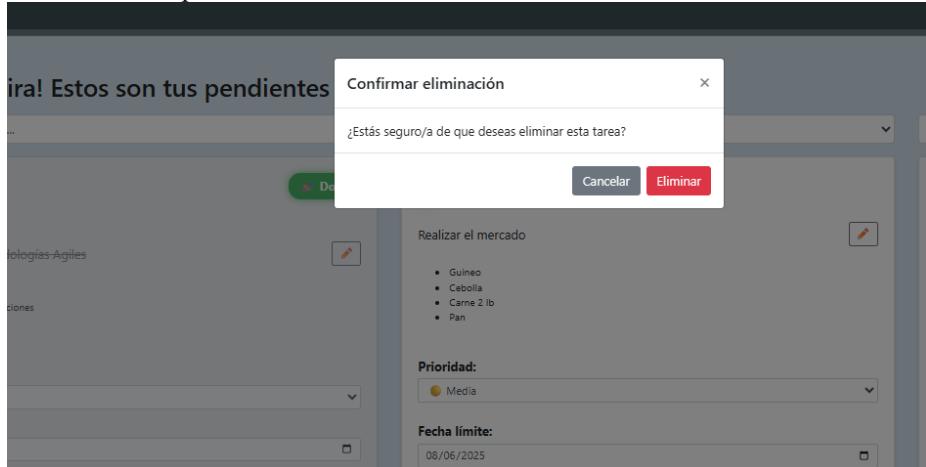
### Edición en misma ventana sin recarga de pagina de la Tarea:



Vista de tarea finalizada con etiqueta done y sombreada para denotar que esta finalizada:



Modal de alerta por si desea eliminar tarea:



Filtrado de buscado de Tareas por nombre, prioridad o fecha todo sin necesidad de recargar la página :



Vista de usuarios registrados junto con su descripción:

The screenshot shows the user profile 'Samira' on the left. The main area is titled 'Usuarios Registrados' and shows a list of users. It displays '3' users found. The first user is 'Usuario Ejemplo' (ID: 1, Email: user@ua) with a placeholder profile picture. The second user is 'Usuario Ejemplo' (ID: 2, Email: ejemplo@gmail.com) with a placeholder profile picture. The third user is '1234' (ID: 3, Email: 1234@gmail.com) with a placeholder profile picture.

### Código Principal añadido:

- **Clase TareaRestController**



Java code for a REST API controller (TareaController.java):

```
1 package com.martinmazza.todos.todoservice;
2
3 import com.martinmazza.todos.todoservice.TareaService;
4 import org.springframework.beans.factory.annotation.Autowired;
5 import org.springframework.http.ResponseEntity;
6 import org.springframework.web.bind.annotation.*;
7
8 @RestController
9 @RequestMapping("/api/tareas")
10 public class TareaController {
11
12     @Autowired
13     TareaService tareaService;
14
15     @PutMapping("/{id/titulo}")
16     public ResponseEntity<Tarea> actualizarTitulo(@PathVariable long id, @RequestBody Map<String, String> payload) {
17         String nuevoTitulo = payload.get("titulo");
18         if (nuevoTitulo == null || nuevoTitulo.isEmpty()) {
19             return ResponseEntity.badRequest().body("Body vacío");
20         }
21         tareaService.modificarTitulo(id, nuevoTitulo);
22         return ResponseEntity.ok().build();
23     }
24 }
```

`@RestController`: Indica que esta clase es un controlador de tipo REST (retorna JSON).

• `@RequestMapping("/api/translate")`: Define la ruta base para las peticiones que maneja este controlador.

Se está inyectando automáticamente una clase de servicio llamada `TranslateService`, que contiene la lógica de negocio para traducir textos.

Escucha peticiones POST en la ruta /api/translate/text/{id}.

Recibe:

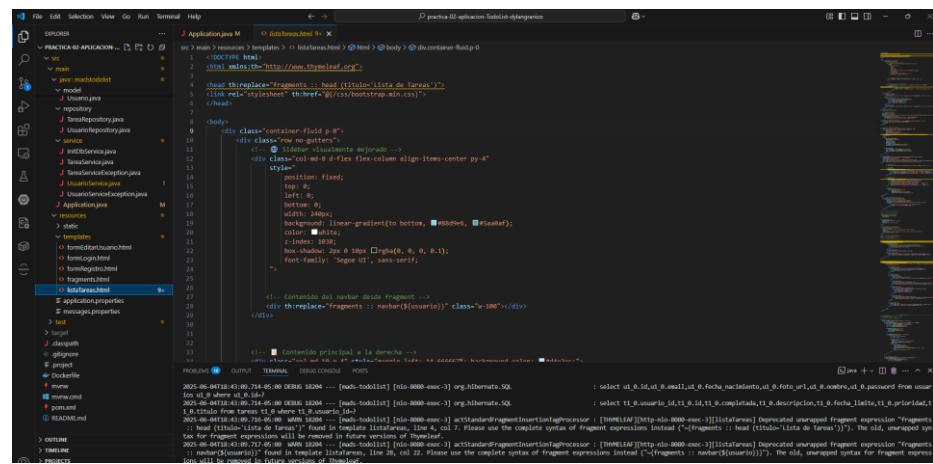
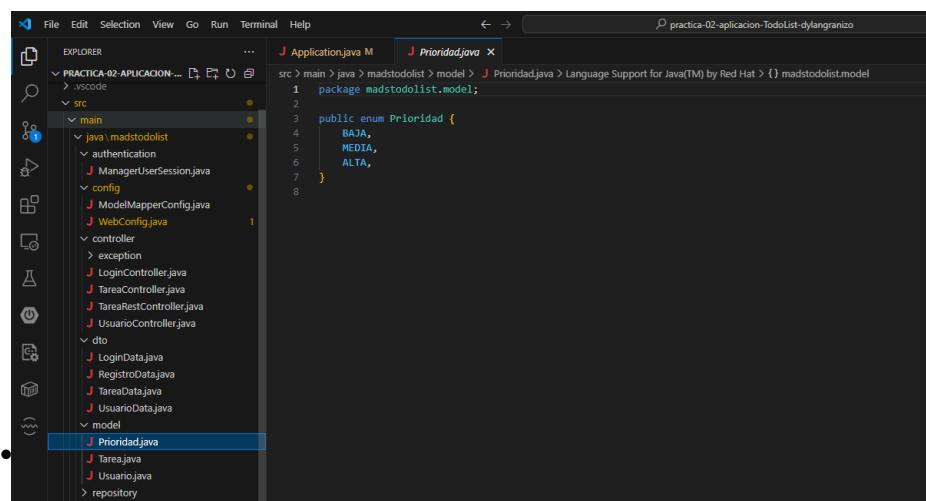
- Un ID como variable en la URL (@PathVariable Long id).
  - Un cuerpo JSON (@RequestBody) que debe contener un campo llamado "text".

- Clase WebConfig, necesario para edición de foto de perfil:



```
Application.java M  J WebConfig.java 1 x
...
1 package madodolist.config;
2
3 import org.springframework.context.annotation.Configuration;
4 import org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;
5 import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
6
7 @Configuration
8 public class WebConfig implements WebMvcConfigurer {
9
10     @Override
11     public void addResourceHandlers(ResourceHandlerRegistry registry) {
12         registry
13             .addResourceHandler("/imgonis/*")
14             .addResourceLocations("file:///C:/Users/Dylan/Desktop/Metodologias%20Agiles%20ImagerF/");
15     }
16 }
17
```

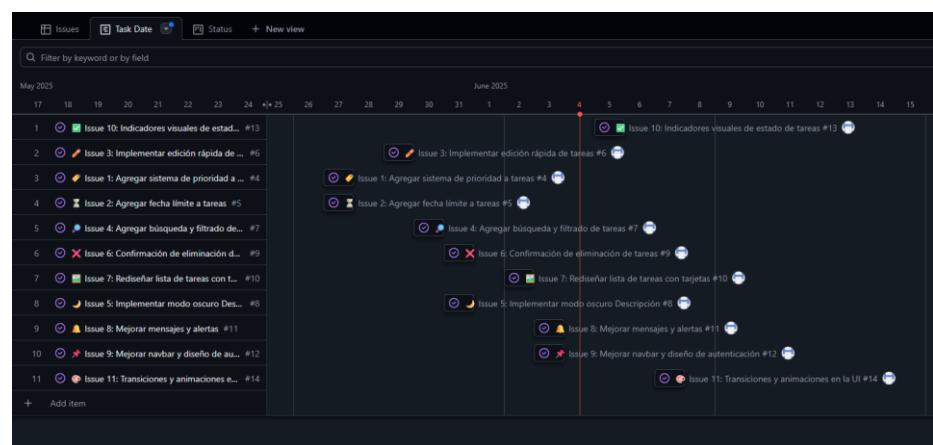
- `registry.addResourceHandler("/imagenes/perfiles/**")` indica que todas las peticiones HTTP que lleguen a URLs que empiecen con `/imagenes/perfiles/` serán manejadas como recursos estáticos.
- `addResourceLocations("file:///C:/Users/Dylan/Desktop/Metodologías Agiles/imagperf/")` indica la ubicación física en el disco duro donde están almacenados esos recursos estáticos.
- **Clase Prioridad en model/Prioridad.java para las prioridades**

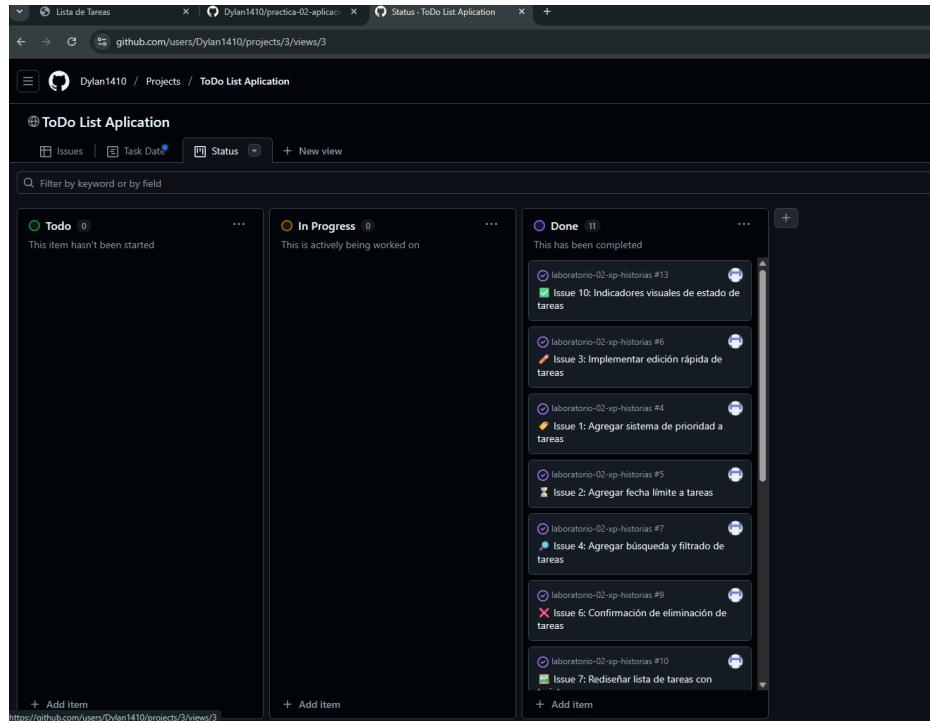


Se destaca también el uso importante de Implementación de DTOs para desacoplar lógica y Endpoints REST con `@PutMapping`, `@PostMapping` para AJAX.

## Uso de Github Proyect para desarrollo de Issues

Title	Assignee	Status	Priority
Issue 10: Indicadores visuales de estado de tareas #13	Dylan1410	Done	Baja
Issue 3: Implementar edición rápida de tareas #6	Dylan1410	Done	Baja
Issue 1: Agregar sistema de prioridad a tareas #4	Dylan1410	Done	Baja
Issue 2: Agregar fecha límite a tareas #5	Dylan1410	Done	Baja
Issue 4: Agregar búsqueda y filtrado de tareas #7	Dylan1410	Done	Media
Issue 6: Confirmación de eliminación de tareas #9	Dylan1410	Done	Media
Issue 7: Rediseñar lista de tareas con tarjetas #10	Dylan1410	Done	Baja
Issue 5: Implementar modo oscuro Descripción #8	Dylan1410	Done	Media
Issue 8: Mejorar mensajes y alertas #11	Dylan1410	Done	Baja
Issue 9: Mejorar navbar y diseño de autenticación #12	Dylan1410	Done	Baja
Issue 11: Transiciones y animaciones en la UI #14	Dylan1410	Done	Baja





## 6 Conclusiones

El desarrollo de esta práctica permitió aplicar de forma real los principios de XP, destacando la importancia de la iteración continua, las pruebas automatizadas y la arquitectura limpia. Las funcionalidades agregadas transformaron una aplicación básica en una plataforma web más rica, dinámica y profesional. El trabajo evidenció que XP no es solo una metodología, sino una filosofía de trabajo orientada a la mejora constante, el feedback y la entrega de valor.

## 7 Github Repository/Github Project (links)/ Docker Hub

<https://github.com/Dylan1410/practica-02-aplicacion-TodoList-dylangranizo.git>  
<https://github.com/users/Dylan1410/projects/3>

<https://hub.docker.com/r/dylan1410/mads-todolist/tags>

## 8 Referencias

- [1] K. Beck and C. Andres, *Extreme Programming Explained: Embrace Change*, 2nd ed. Boston, MA, USA: Addison-Wesley, 2004.