



ECOLE MAROCAINE DES
SCIENCES DE L'INGENIEUR
Membre de

 HONORIS UNITED UNIVERSITIES

ARCHITECTURE DES
COMPOSANTS DES
ENTREPRISES

Rapport de projet Kanbaniser

Encadré par : Dr. Mohamed Lachgar

Préparé par : Anas Mourad

Imad Maailil

Hamza Lghali

Année Universitaire 2023/2024

Introduction

Aperçu du projet :

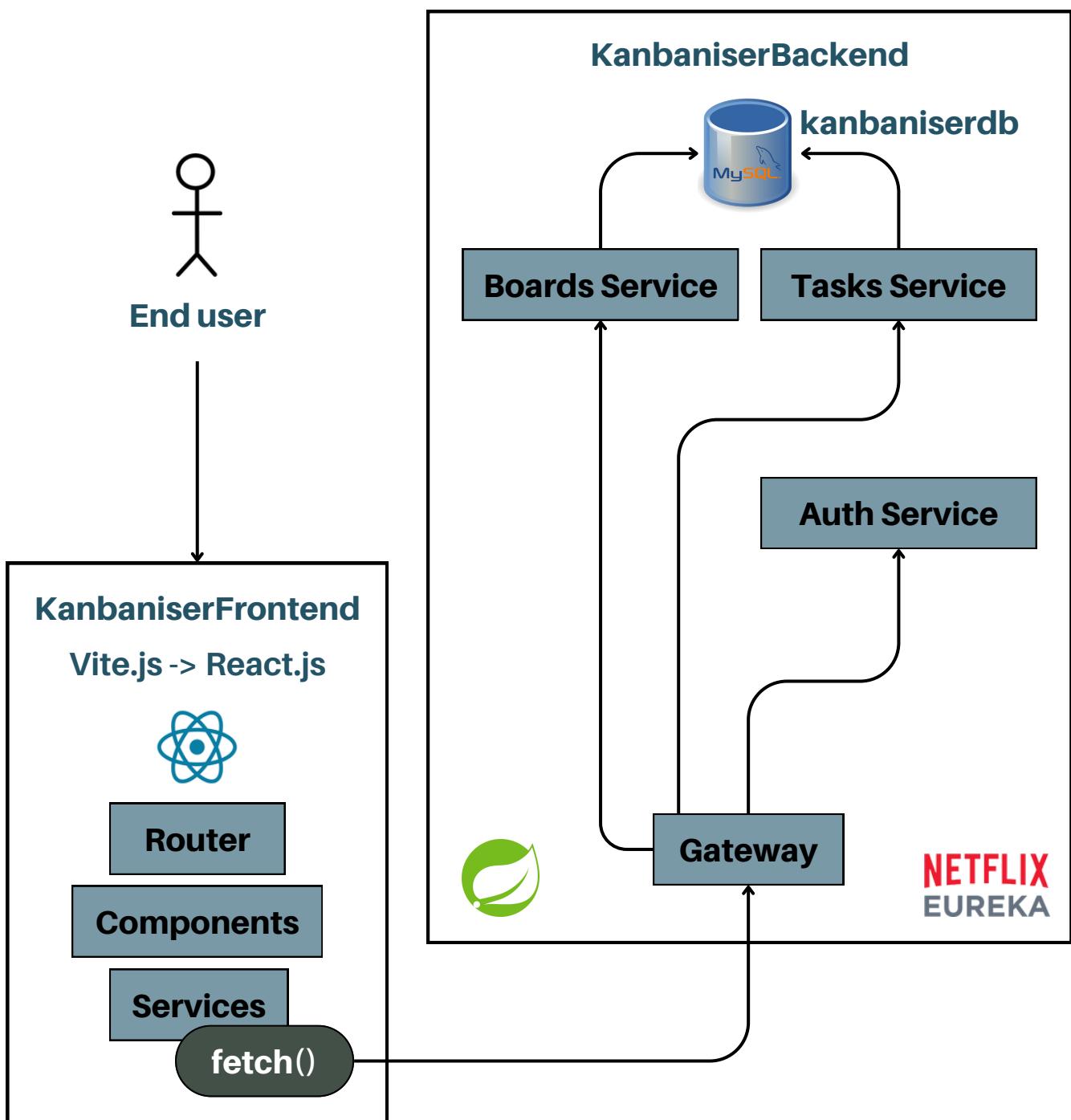
Notre application "Kanbaniser" nous aide à organiser des projets en utilisant des tableaux visuels. Chaque tableau représente une étape du projet, et on peut y ajouter des tâches. La barre latérale offre une navigation pratique entre les tableaux et les membres de l'équipe. On peut également voir les détails des tâches et des utilisateurs. L'application est conçue pour être conviviale et offre une expérience visuelle pour mieux gérer nos projets en utilisant la méthode Kanban.

Importance de l'architecture microservices :

- L'architecture microservices améliore la scalabilité de notre application Kanbaniser en permettant une gestion indépendante de chaque service.
- La flexibilité accrue de cette architecture facilite le déploiement et la mise à jour des fonctionnalités spécifiques de notre projet Kanban.
- La maintenance devient plus aisée avec des services indépendants, favorisant une évolutivité optimale pour l'application Kanbaniser.
- La résilience améliorée de l'architecture microservices assure la stabilité et la disponibilité continue de notre solution de gestion de projet Kanban.
- La modernisation de notre infrastructure grâce à l'architecture microservices renforce la robustesse de l'ensemble de l'application Kanbaniser.

Architecture microservices

Architecture :



Architecture microservices

Description des services :

- **Gateway** : En tant que point d'entrée centralisé, il prend en charge le routage, l'authentification, et d'autres aspects transversaux, offrant ainsi une interface unifiée pour les clients externes de KanbaniserMicroservices.
- **Netflix Eureka Server** : Fondamental pour la découverte et l'enregistrement des microservices, il crée un environnement hautement scalable et résilient en facilitant la communication dynamique entre les services de KanbaniserMicroservices.
- **Boards Service** : Au cœur de la gestion des tableaux Kanban, ce service offre une gamme complète de fonctionnalités, de la manipulation des entités à la communication transparente avec la base de données.
- **Tasks Service** : Gérant les opérations liées aux tâches, il offre une solution modulaire dédiée à la gestion efficace des tâches associées aux tableaux Kanban.
- **Auth Service** : Gérant de manière sécurisée les aspects cruciaux de l'authentification et de l'autorisation, il assure la protection des données sensibles dans le contexte du système KanbaniserMicroservices.

Architecture microservices

Mécanismes de communication :

- **HTTP Requests et Responses :**

L'échange d'informations entre les microservices s'effectue via des requêtes HTTP et des réponses, garantissant une communication légère et standardisée. Les données sont généralement transmises au format JSON, favorisant ainsi une interchangeabilité et une lisibilité accrues.

- **Envoi de Requêtes :**

Lorsqu'un microservice doit solliciter des informations d'un autre service, il formule une requête HTTP, spécifiant les détails requis. Les paramètres et les données pertinentes sont inclus dans le corps de la requête au format JSON.

Exemple :

GET /api/boards/{boardId} HTTP/1.1

Host: boards-service:8082

Architecture microservices

Mécanismes de communication :

- **Réception de Réponses :**

Le service destinataire répond avec une réponse HTTP, contenant les données demandées. Les informations sont encapsulées dans le corps de la réponse au format JSON pour une transmission claire et structurée.

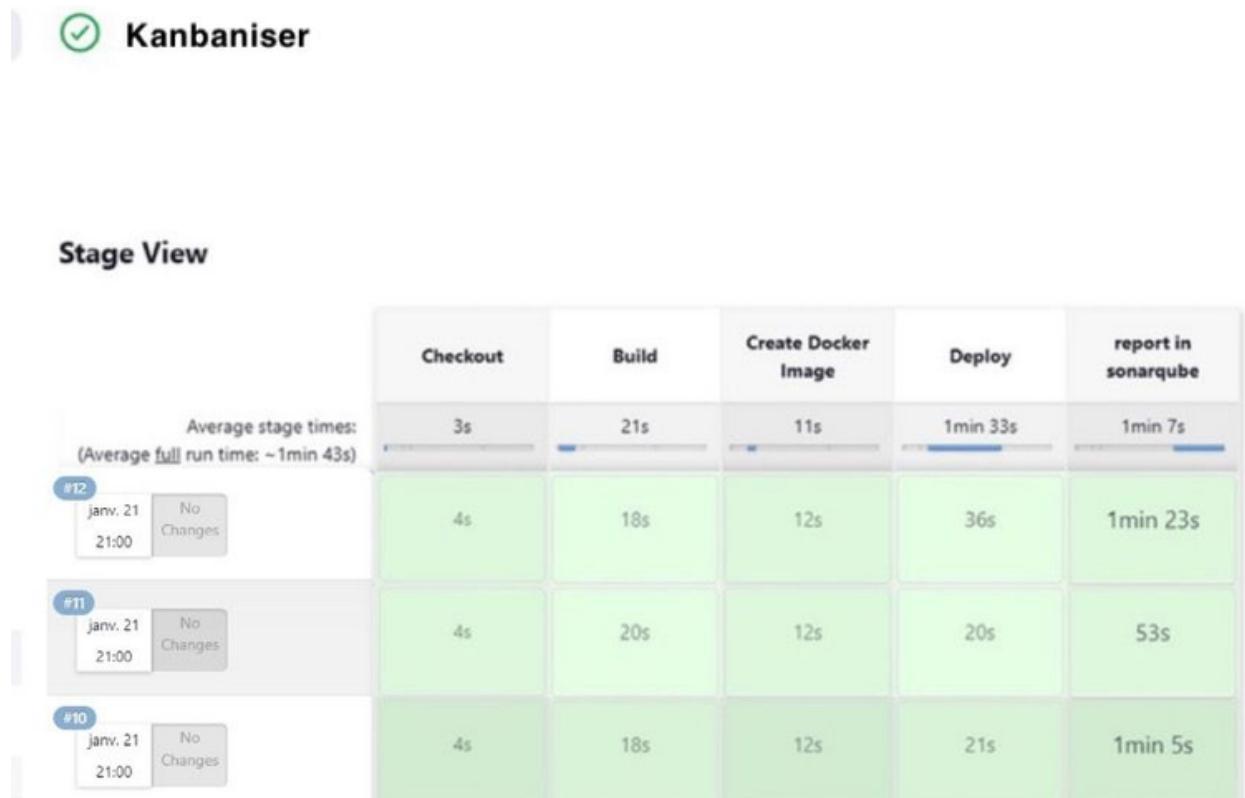
Exemple :

HTTP/1.1 200 OK
Content-Type: application/json

```
{  
  "id":1,  
  "title":"test",  
  "description":"test",  
  "tasks": [  
    {  
      // omitted details for space  
    },  
    {  
      // omitted details for space  
    },  
    {  
      // omitted details for space  
    }  
  ]  
}
```

CI/CD avec Jenkins

Processus :



CI/CD avec Jenkins

Configuration :

```
pipeline {
    agent any

    stages {
        stage('Checkout') {
            steps {
                script {
                    checkout([$class:'GitSCM', branches: [[name:'main']], userRemoteConfigs: [[url:'https://github.com/Bajmo/KanbaniserMicroservices.git']]])
                }
            }
        }

        stage('Build') {
            steps {
                script {
                    dir('Auth') {
                        bat 'mvn clean install'
                    }
                }
            }
        }

        stage('Create Docker Image') {
            steps {
                script {
                    dir('Auth') {
                        bat 'docker-compose build'
                    }
                }
            }
        }

        stage('Deploy') {
            steps {
                script {
                    dir('Auth') {
                        bat 'docker-compose up -d'
                    }
                }
            }
        }

        stage('report in sonarqube') {
            steps {
                bat "mvn clean verify sonar:sonar -Dsonar.projectKey=kanbaniser -Dsonar.projectName='Kanbaniser' -Dsonar.login=8d65fcea686fac653dafc7d1380594d726345f8f"
            }
        }
    }
}
```

CI/CD avec Jenkins

- **Checkout:**

- Le pipeline commence par récupérer le code source du référentiel GitHub (<https://github.com/Bajmo/KanbaniserMicroservices.git>).
- Il récupère spécifiquement la branche 'main'.

- **Build:**

- À l'intérieur du répertoire 'Auth', on lance la construction Maven en utilisant la commande **mvn clean install**.
- Cette étape est responsable de la compilation du code source, de l'exécution des tests et de l'emballage de l'application dans un fichier JAR ou WAR.

- **Create Docker Image:**

- Toujours dans le répertoire 'Auth', il exécute une commande de construction Docker Compose (**docker-compose build**).
- Cette étape est responsable de la création d'une image Docker pour le service 'Auth' en fonction de la configuration Docker Compose.

CI/CD avec Jenkins

- **Deploy:**

- Encore une fois, dans le répertoire 'Auth', il déploie les conteneurs Docker en mode détaché (**docker-compose up -d**).
- Cette étape démarre les conteneurs Docker pour le service 'Auth'.

- **Report in SonarQube:**

- Cette étape effectue une analyse statique du code en utilisant SonarQube.
- Il exécute la commande Maven mvn clean verify sonar:sonar avec des paramètres supplémentaires :
- **-Dsonar.projectKey=kanbaniser** : Spécifie la clé du projet dans SonarQube.
- **-Dsonar.projectName='Kanbaniser'** : Spécifie le nom du projet dans SonarQube.
- **-Dsonar.login= <code>** : Spécifie la clé d'authentification pour SonarQube.

Déploiement Automatique

Webhooks / Manage webhook

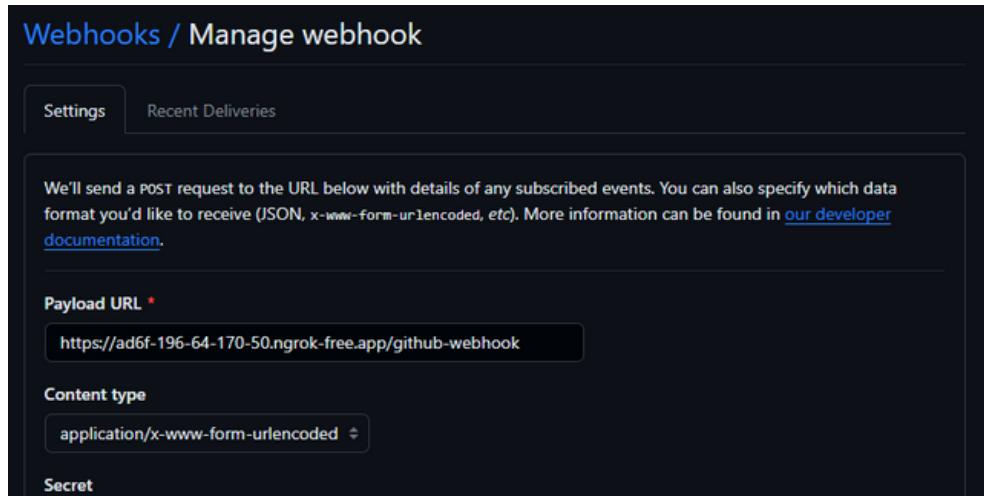
Settings Recent Deliveries

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#).

Payload URL *
https://ad6f-196-64-170-50.ngrok-free.app/github-webhook

Content type
application/x-www-form-urlencoded

Secret



```
ngrok http 8082 --verify-webhook twilio --verify-webhook-secret "{twilio webhook secret}"
```

```
Command Prompt - ngrok http 8082 --verify-webhook twilio --verify-webhook-secret "{twilio webhook secret}"
ngrok

Build better APIs with ngrok. Early access: ngrok.com/early-access

Session Status          online
Account                 Anas Mourad (Plan: Free)
Version                 3.5.0
Region                  Europe (eu)
Latency
Web Interface
Forwarding
Connections            ttl     opn      rt1      rt5      p50      p90
                        0       0       0.00    0.00    0.00    0.00
```



Analyse SonarQube

HamzaLghali > KanbaniserBackend > master ✓

Summary Issues Security Hotspots Measures Code Activity

603 Lines of Code ⓘ Last analysis 26 minutes ago ⓘ 9eac9890 testphase1

Quality Gate ⓘ Passed New Code Overall Code
New code: Since about 8 hours ago

✓ Reliability A Maintainability A
0 Bugs 4 Code Smells

8 Security A Security Review A
0 Vulnerabilities 0 Security Hotspots

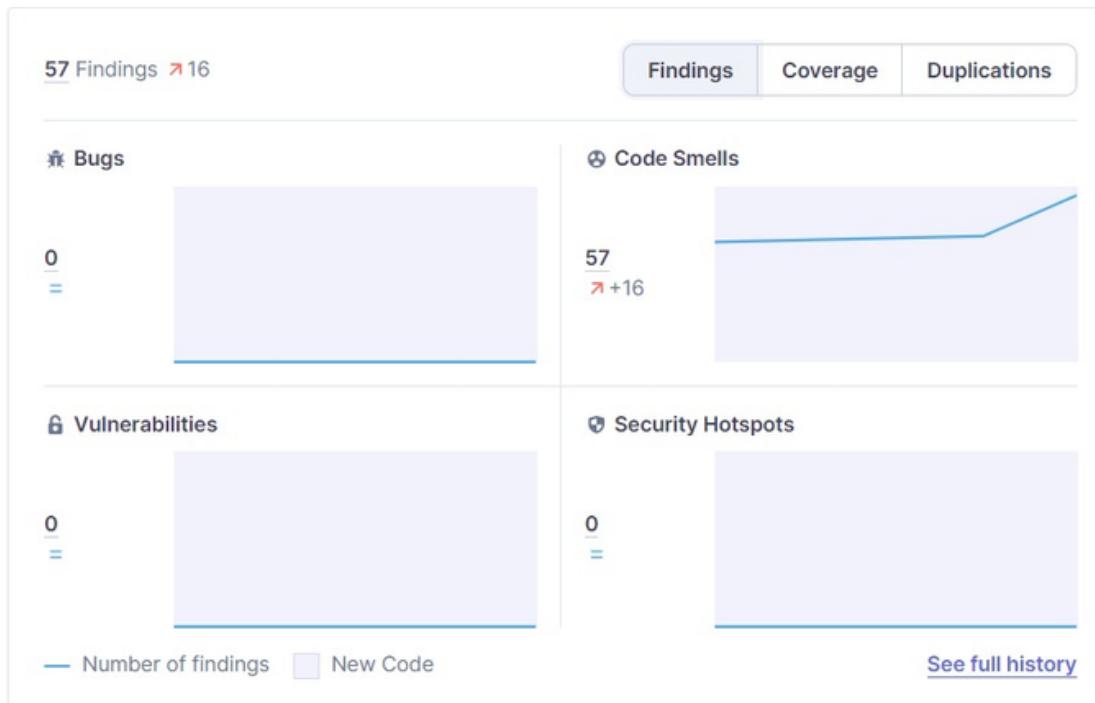
Coverage Duplications
A few extra steps are needed for SonarCloud to analyze your code coverage. [Setup coverage analysis](#) ⓘ
0.0% Duplications on 408 New Lines

25.9% Coverage 25.9% Findings Coverage Duplications

378 Lines to Cover +206
65 Covered Lines +65

— Lines to Cover — Covered Lines □ New Code See full history

Analyse SonarQube



Apperçu du projet Front-end

The image displays two screenshots of a web application named "Kanbaniser".

Top Screenshot: Shows the main dashboard. The header includes a navigation menu icon, the user's name "Anas Mourad" with a dropdown arrow, and a profile picture. Below the header, a large "Welcome to Kanbaniser!" message is centered. A horizontal line connects this screenshot to the sidebar in the bottom one.

Bottom Screenshot: Shows the sidebar on the left and the main dashboard area on the right. The sidebar lists "Boards" (Project Board 1, Project Board 2, Test Project) and "Team Members" (John Doe, Anas Mourad, Sally Smith). The main dashboard area also features a "Welcome to Kanbaniser!" message. The user's name "Anas Mourad" is at the top, and a dropdown menu is open, showing "Profile" and "Sign out".

Apperçu du projet Front-end

The image displays three screenshots of a web application interface, likely a user profile editor, showing different states of form validation.

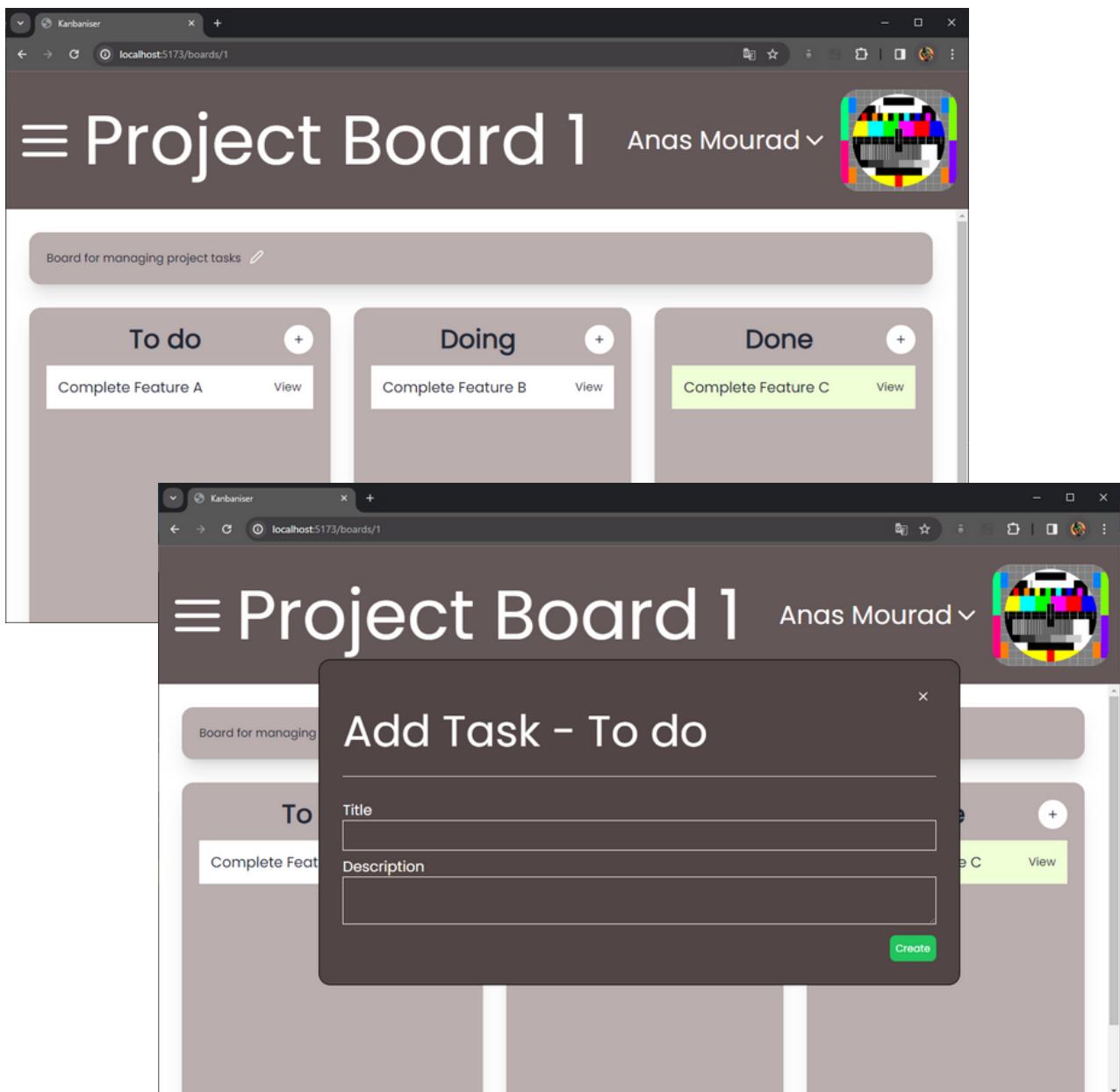
Screenshot 1 (Top Left): Shows a user profile for "Anas Mourad". The profile picture is a standard test pattern. The form fields are displayed without validation:

- First Name: Anas (with a small edit icon)
- Last Name: Mourad (with a small edit icon)
- Email: anas.mourad@example.com (with a small edit icon)
- Phone number: 123-456-7890 (with a small edit icon)
- Change Password: ***** (with a small edit icon)

Screenshot 2 (Top Right): Shows the same user profile and form fields, but the email field now has a red border and a validation message: "Invalid email format".

Screenshot 3 (Bottom): Shows the user profile and form fields again. The email field now has a red border and a validation message: "Invalid email format". The phone number field also has a red border and a validation message: "Phone number must be a valid number".

Apperçu du projet Front-end



Apperçu du projet Front-end

The image displays two screenshots of a web-based project management tool, likely Kanbanizer, showing a task creation and editing process.

Screenshot 1: Task Creation

A modal window titled "Complete Feature A" is open, showing the following details:

- Description:** Finish implementing Feature A by the deadline
- Created by:** Anas Mourad
- Created at:** 25/11/2023, 1:50:26 AM

The background shows a "Project Board 1" with columns labeled "To do", "In progress", and "Done". A task card for "Complete Feature A" is visible in the "To do" column.

Screenshot 2: Task Editing

A modal window titled "Edited task title" is open, showing the following details:

- Description:** Finish implementing Feature A by the deadline (highlighted with a red border and a "Too long" message)
- Created by:** Anas Mourad
- Created at:** 25/11/2023, 1:50:26 AM

The background shows the same "Project Board 1" with the edited task card now in the "In progress" column.

Apperçu du projet Front-end

The image displays two screenshots of a web-based project management application named "Kanbaniser".

Screenshot 1: Add Board

This screenshot shows the "Add Board" modal window. The title of the board is "Project Board 1". The modal contains fields for "Title" and "Description", both of which are currently empty. A "Create" button is located at the bottom right of the modal.

Screenshot 2: Add User

This screenshot shows the "Add User" modal window. It requires input for "First name", "Last name", "Phone number", "Email address", and "Initial password – Please change after first login". There is also a field for "Paste a link to your profile image". Similar to the first modal, it has a "Create" button at the bottom right.

Left Sidebar (Common to both screenshots):

- Boards:** Project Board 1, Project Board 2, Test Project.
- Team Members:** John Doe, Anas Mourad, Sally Smith.

Right Sidebar (Common to both screenshots):

- A list of boards: Project Board 1, Project Board 2, Test Project.
- A "View" button next to the Test Project entry.
- A "Create" button in the top right corner of the sidebar.
- A decorative "TV Test Pattern" icon in the top right corner.

Tests front-end Selenium IDE

The image displays two side-by-side screenshots of the Selenium IDE interface, both titled "Selenium IDE - Kanbaniser".

Top Screenshot: This screenshot shows a test case for a task named "Delete task". The test steps are as follows:

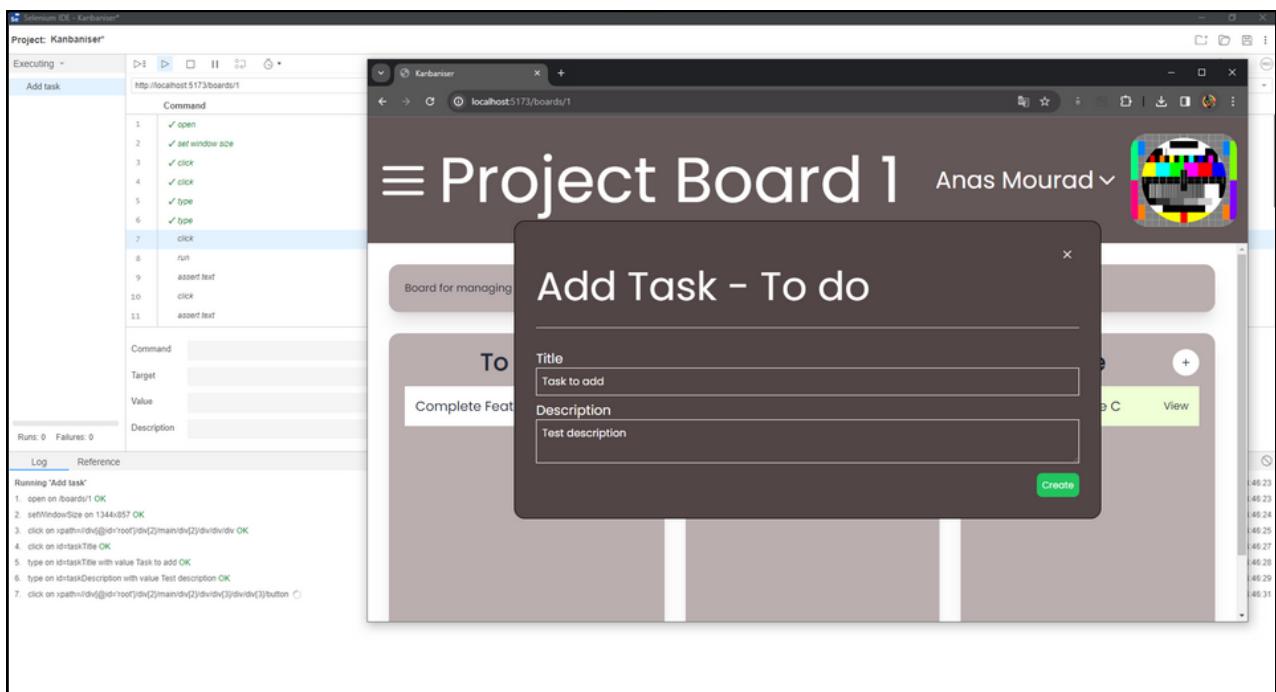
Line	Command	Target	Value
20	assert editable	xpath=//div[@id='task-2']/div[2]/div/div[2]/div/div/textarea	
21	type	css=.text-md	Test description - edit
22	click	css=.justify-between > .self-center:nth-child(2) > svg	
23	assert text	xpath=//div[@id='task-2']/div[2]/div/div[2]/div	Test description - edit

Bottom Screenshot: This screenshot shows a different part of the test case for the same "Delete task". The test steps are as follows:

Line	Command	Target	Value
13	click	xpath=//div[@id='task-1']/div/span[2]	
14	assert element not present	xpath=//div[@id='task-1']/div	
15	assert element present	xpath=//div[@id='task-2']/div	

Both screenshots include a toolbar at the top with various icons for file operations, search, and recording. The bottom of each window has tabs for "Log" and "Reference".

Tests front-end Selenium IDE



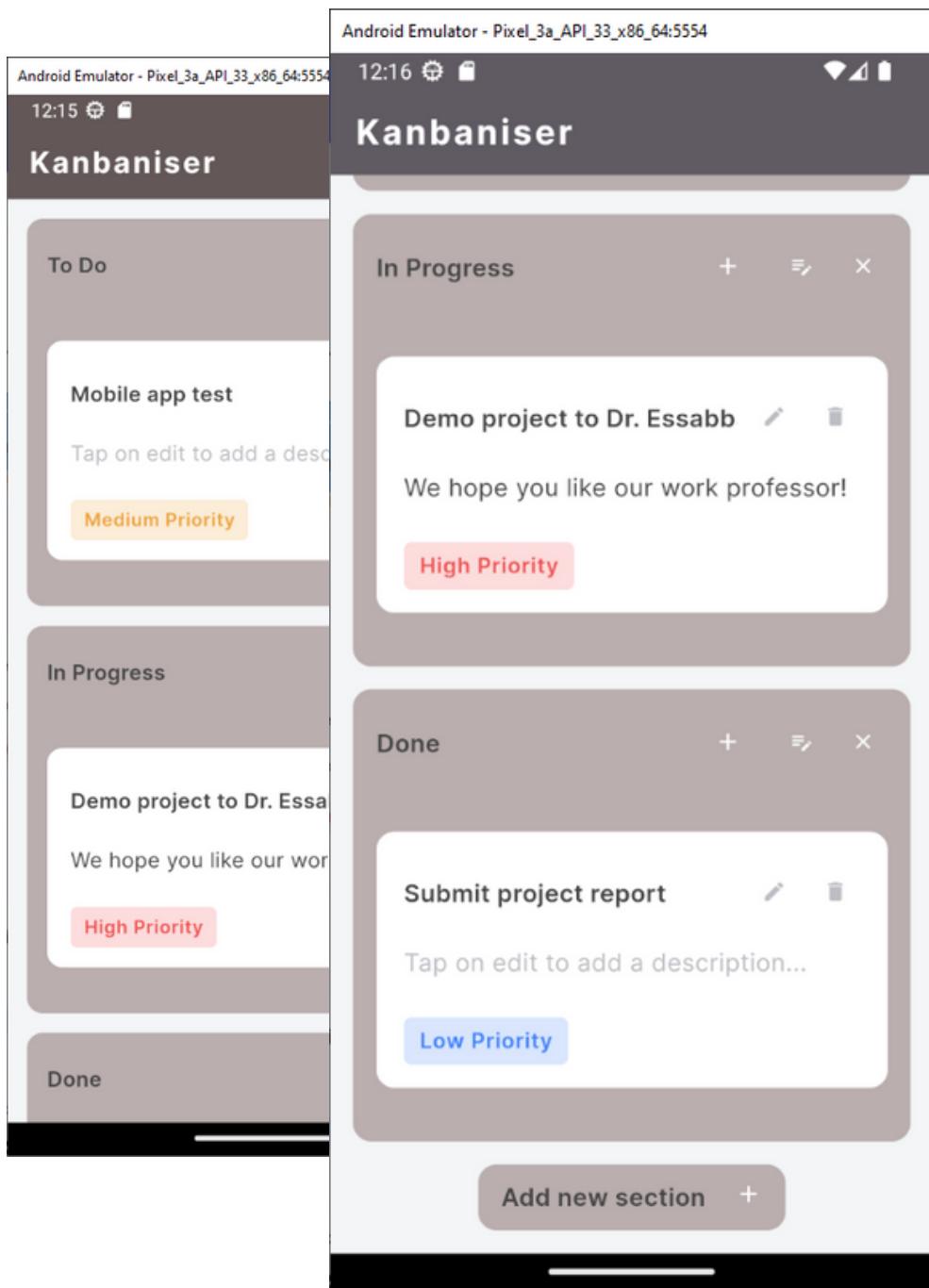
Voir la vidéo :

https://drive.google.com/file/d/1c79f7HHqg9YkBr_qwwWDcifg4QXQ0xXf/view?usp=drive_link

Apperçu du projet Application mobile - WIP

Pile technologique : Flutter

Dart



Tests JUnit

```
@Test
public void testTaskGettersAndSetters() {
    Task task = new Task();
    User user = new User();
    Board board = new Board();
    Date now = new Date();

    task.setId(1);
    task.setTitle("Sample Task");
    task.setDescription("This is a sample task description");
    task.setCreatedAt(now);
    task.setCreatedBy(user);
    task.setBoard(board);

    assertEquals(1, task.getId());
    assertEquals("Sample Task", task.getTitle());
    assertEquals("This is a sample task description", task.getDescription());
    assertEquals(now, task.getCreatedAt());
    assertEquals(user, task.getCreatedBy());
    assertEquals(board, task.getBoard());
}

}

// Test
public void testTaskEqualsHashCodeToStringAndConstructor() {
    User user = new User(); // Assuming User has an empty constructor
    Board board = new Board(); // Assuming Board has an empty constructor
    Date date = new Date();

    Task task1 = new Task(1, "Task 1", "Description 1", date, user, board);
    Task task2 = new Task(1, "Task 1", "Description 1", date, user, board);
    Task task3 = new Task(2, "Task 2", "Description 2", new Date(), user, board);

    // equals and hashCode
    assertEquals(task1, task2);
    assertNotEquals(task1, task3);
    assertEquals(task1.hashCode(), task2.hashCode());
    assertEquals(task1.hashCode(), task3.hashCode());

    // toString
    assertNotNull(task1.toString());
    assertTrue(task1.toString().contains("Task 1"));

}
```

Voir plus :

<https://github.com/HamzaLghali/KanbaniserBackend/tree/master/src/test/java/com/project/kanbaniser>

Tests JUnit

Taux de couverage

Kanbaniser

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.project.kanbaniser.Services		5%		0%	50	53	110	113	43	46	0	3
com.project.kanbaniser.Controllers		39%		n/a	16	27	32	51	16	27	0	3
com.project.kanbaniser.Entities		87%		50%	60	132	0	34	0	56	0	3
com.project.kanbaniser		37%		n/a	1	2	2	3	1	2	0	1
com.project.kanbaniser.Exceptions		100%		n/a	0	4	0	8	0	4	0	4
Total	807 of 1,733	53%	90 of 166	45%	127	218	144	209	60	135	0	14

com.project.kanbaniser.Controllers

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
BoardController		7%		n/a	9	10	18	19	9	10	0	1
TaskController		10%		n/a	7	8	14	15	7	8	0	1
UserController		100%		n/a	0	9	0	17	0	9	0	1
Total	123 of 202	39%	0 of 0	n/a	16	27	32	51	16	27	0	3