

**Dossier de conception**

>> Diginamic.fr

|  |  |  |  |
| --- | --- | --- | --- |
| **Révision** | **Rédacteurs** | **Date** | **Objet** |
| 1 | Lucas Preaux | 02/12/2024 | Création du document |
| 2 | Florian Furnari | 02/12/2024 | Création du document |
|  |  |  |  |

# Introduction

## Objet du document

Ce document a pour objectif de présenter l’essentiel des questions techniques liées à la mise en place du projet ScrumTogether.

Ce document présente :

* L’architecture logicielle de l’API
* Le diagramme de classes
* Le diagramme entités relations
* Le découpage en couches
* Les règles de développement et la qualité de code

# Sommaire

### Table des matières

[1 Introduction 2](#_Toc184046130)

[1.1 Objet du document 2](#_Toc184046131)

[2 Sommaire 3](#_Toc184046132)

[2.1.1 Table des matières 3](#_Toc184046133)

[3 Architecture logicielle 4](#_Toc184046134)

[3.1 Produits et versions 4](#_Toc184046135)

[3.1.1 Langages, frameworks et librairies spécifiques 4](#_Toc184046136)

[3.1.2 Serveur de base de données 4](#_Toc184046137)

[4 Focus Technique 5](#_Toc184046138)

[4.1 Diagramme de classes 5](#_Toc184046139)

[4.2 Diagramme entités relations 6](#_Toc184046140)

[4.3 Découpage en couches 6](#_Toc184046141)

[4.4 Liste des endpoints 7](#_Toc184046142)

[Règles de développement et qualité de code 8](#_Toc184046143)

# Architecture logicielle

## Produits et versions

### Langages, frameworks et librairies spécifiques

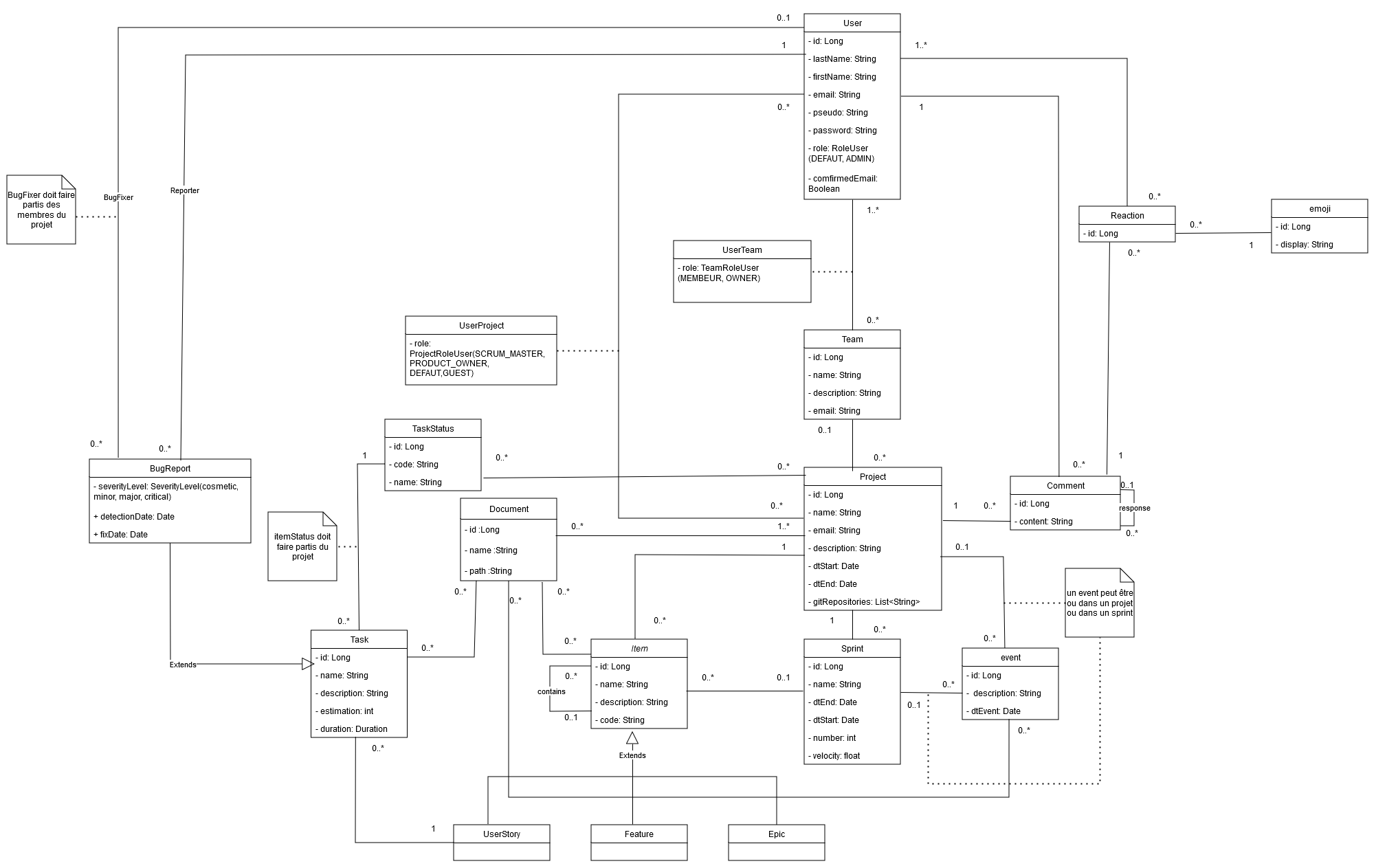
|  |  |
| --- | --- |
| **Nom** | **Version** |
| Langage Java | 21 |
| SPRING BOOT | 3.4.0 |
| JPA / Hibernate |  |
| OpenAPI |  |
| Spring Hateoas |  |
|  |  |
|  |  |

### Serveur de base de données

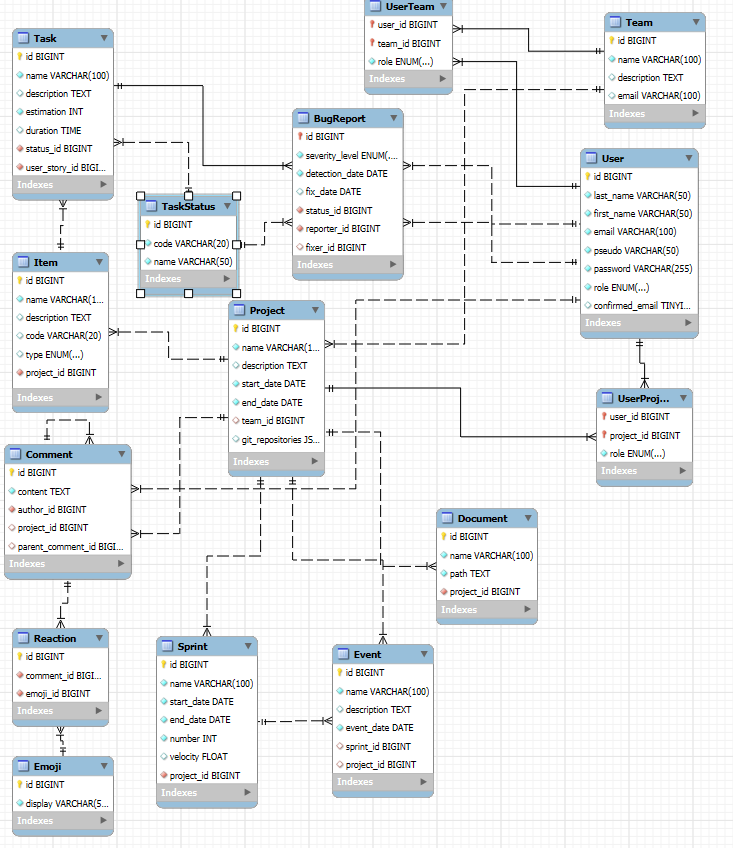
|  |  |
| --- | --- |
| **Nom** | **Version** |
| MySQL | 9 |
|  |  |

# Focus Technique

## Diagramme de classes



## Diagramme entités relations



## Découpage en couches

Présenter les diverses couches de votre API REST.

Notre application serra repartie sur différente couche :  
- la couche controller  
Cette couche a pour but d’interagir avec les différentes requête HTTP (GET, POST etc…), envoyées par le frontend.

- La couche metier (services)  
qui ont pour but de servir d’intermédiaire entre le contrôleur et les différentes logiques

- la couche d’accès aux donnée (repositoies)  
qui nous servira à récupérer les données dans la base de données

## Liste des endpoints

Liste des endpoints de l’API :

|  |  |
| --- | --- |
| Description | Endpoint |
| Créer un nouvel utilisateur | POST /users |
| Récupérer les détails d’un utilisateur. | GET /users/:id |
| Lister tous les utilisateurs | GET /users |
| Mettre à jour les informations d’un utilisateur. | PUT /users/:id |
| Supprimer ou désactiver un utilisateur. | DELETE /users/:id |
| Authentifier un utilisateur et générer un token | POST /auth/login |
| Déconnecter un utilisateur. | POST /auth/logout |
|  |  |
| Créer une nouvelle équipe. | POST /teams |
| Lister toutes les équipes. | GET /teams |
| Récupérer les détails d’une équipe spécifique. | GET /teams/:id |
| Mettre à jour les informations d’une équipe. | PUT /teams/:id |
| Supprimer une équipe. | DELETE /teams/:id |
|  |  |
| Créer un projet | POST /projects |
| Lister tous les projet | GET /projects |
| Récupérer les détails d’un projet. | GET /projects/:id |
| Mettre à jour les informations d’un projet. | PUT /projects/:id |
| Supprimer un projet. | DELETE /projects/:id |
| Récupérer les utilisateurs d’un projet | GET /projects/:id/users/ |
| Mettre à jour la liste d’utilisateurs d’un projet | PUT /projects/:id/users/ |
| Retierer un utilisateur d’un projet | DELETE /projects/:id/users/:userId |
|  |  |
| Créer un nouvel Épic. | POST /epics |
| Lister tous les Epics. | GET /epics |
| Récupérer les détails d’un Epic | GET /epics/:id |
| Mettre à jour un Epic | PUT /epics/:id |
| Supprimer un Epic | DELETE /epics/:id |
|  |  |
| Créer une nouvelle Feature | POST /features |
| Lister toutes les Features | GET /features |
| Récupérer les détails d’une Feature | GET /features/:id |
| Mettre à jour une Feature. | PUT /features/:id |
| Supprimer une Feature. | DELETE /features/:id |
|  |  |
| Créer une nouvelle User Story. | POST /userstories |
| Lister toutes les User Stories. | GET /userstories |
| Récupérer les détails d’une User Story. | GET /userstories/:id |
| Mettre à jour une User Story. | PUT /userstories/:id |
| Supprimer une User Story. | DELETE /userstories/:id |
| Lister les Tâches et Rapport de bugs associées à une User Story | GET /userstories/:id/tasks |
|  |  |
| Créer une nouvelle tâche | POST /tasks |
| Lister toutes les tâches. | GET /tasks |
| Récupérer les détails d’une tâche. | GET /tasks/:id |
| Mettre à jour une tâche. | PUT /tasks/:id |
| Supprimer une tâche. | DELETE /tasks/:id |
|  |  |
| Créer un nouveau rapport de bug | POST /bugreports |
| Lister tous les rapports de bugs. | GET /bugreports |
| Récupérer les détails d’un rapport de bug | GET /bugreports/:id |
| Mettre à jour un rapport de bug | PUT /bugreports/:id |
| Supprimer un rapport de bug | DELETE /bugreports/:id |
|  |  |
| Créer un sprint | POST /sprints |
| Lister tous les sprints d’un projet | GET /projects/:projectId/sprints |
| Récupérer les détails d’un sprint. | GET /sprints/:id |
| Mettre à jour un sprint | PUT /sprints/:id |
| Supprimer un sprint. | DELETE /sprints/:id |
|  |  |
| Créer un événement | POST /events |
| Lister les événements d’un projet | GET /projects/:projectId/events |
| Lister les événements d’un sprint | GET /sprints/:sprintId/events |
| Lister tous les événements. | GET /events |
| Récupérer un événement spécifique | GET /events/:id |
| Mettre à jour un événement | PUT /events/:id |
| Supprimer un événement | DELETE /events/:id |
|  |  |
| Créer un document | POST /documents |
| Lister les documents associés à un Epic | GET /epics/:epicId/documents |
| Lister les documents associés à un Feature | GET /features/:featureId/documents |
| Lister les documents associés à un User Story | GET /userstories/:userStoryId/documents |
| Récupérer les détails d’un document | GET /documents/:id |
| Supprimer un document | DELETE /documents/:id |
|  |  |
| Créer un commentaire | POST /comments |
| Lister les commentaires d’un projet | GET /projects/:projectId/comments |
| Récupérer un commentaire spécifique | GET /comments/:id |
| Mettre à jour un commentaire | PUT /comments/:id |
| Supprimer un commentaire | DELETE /comments/:id |
|  |  |
| Ajouter une réaction à un commentaire | POST /comments/:id/reactions |
| Supprimer une réaction | DELETE /comments/:id/reactions/:reactionId |
|  |  |
| Ajouter un nouveau statut | POST /statuses |
| Lister tous les statuts d’un projet | GET /projects/:projectId/statuses |
| Récupérer les détails d’un statut | GET /statuses/:id |
| Mettre à jour un statut | PUT /statuses/:id |
| Supprimer un statut | DELETE /statuses/:id |
|  |  |

## Règles de développement et qualité de code

Règles de développement :

* Javadoc à 100%
* Respect des normes de nommage

Il y aura des tests unitaires à développer avec Mockito pour tester le code de toutes les classes de services.