Projet Jardin 🌱

Table des matières

[Ce que fait l’appli / pitch du client 1](#_Toc197638569)

[La structure de la base de données 2](#_Toc197638570)

[Table culture : 2](#_Toc197638571)

[Table jardin : 2](#_Toc197638572)

[Table plante : 2](#_Toc197638573)

[Table utilisateur : (single table) 2](#_Toc197638574)

[Table badges\_obtenus 3](#_Toc197638575)

[Les classes 4](#_Toc197638576)

[Trame pour la présentation 5](#_Toc197638577)

[1. Introduction 5](#_Toc197638578)

[2. Architecture Globale 5](#_Toc197638579)

[2 bis. Gestion de projet & Design 5](#_Toc197638580)

[3. Backend (Spring Boot) 6](#_Toc197638581)

[4. Frontend (Angular) 6](#_Toc197638582)

[5. Sécurité 6](#_Toc197638583)

[6. Conteneurisation avec Docker 6](#_Toc197638584)

[7. CI/CD avec Jenkins 7](#_Toc197638585)

[8. Défis rencontrés & Solutions 7](#_Toc197638586)

[9. Conclusion 7](#_Toc197638587)

[10. Questions/Réponses 7](#_Toc197638588)

# Ce que fait l’appli / pitch du client

L’utilisateur se connecte avec un compte client d’où il peut **gérer son jardin**.

A son jardin est associé un lieu (ville).

Pour son jardin, il peut :

* **Planter** une plante en une certaine quantité

L’utilisateur peut accéder à des **suggestions** de plants de saison.

Il a également accès à un **dictionnaire** pour se renseigner sur les plantes (géré par un admin).

Les plantes à ajouter au jardin sont uniquement celles disponibles dans le dictionnaire.

* **Consulter** l’état de ses plantes (à récolter, à arroser)

**→** *Affichage****:*** *ajout de filtres*

**→** *Affichage****:******To Do List*** *avec les actions à réaliser avec des rappels*

Arrosage :

La date d’arrosage est calculée en fonction de la fréquence d’arrosage et de la date du dernier arrosage. S’il a plu dans la ville, la date d’arrosage est décalée.

Récolte :

La date de récolte est calculée en fonction de la date de plantation et du délai de récolte propre à la plante. Chaque plante possède une période de récolte (durée durant laquelle la plante peut être récoltée une fois mature).

Avant la date de récolte la plante n’est pas mature (pas mûr pour fruit/légume, bourgeon pour fleur).

Pendant la période de récolte la plante est mature (mûr pour fruit/légume, en fleur pour fleur).

Après la période de récolte la plante est morte (pourri pour fruit/légume, fanée pour fleur).

**→** *Affichage****:******jauge*** *état maturité de plante*

* **Récolter** une plante
* **Arroser** une plante
* **Consulter** l’historique / les statistiques de son jardin

L’utilisateur peut collectionner des **badges** qu’il obtient en fonction de son score. Le score est calculé à partir de ses récoltes (possibilité de mettre un nombre de points différent en fonction de difficulté de culture de la plante)

# La structure de la base de données

Une image contenant texte, capture d’écran, Police, nombre

Le contenu généré par l’IA peut être incorrect.

### Table culture :

Une image contenant texte, capture d’écran, Police, nombre

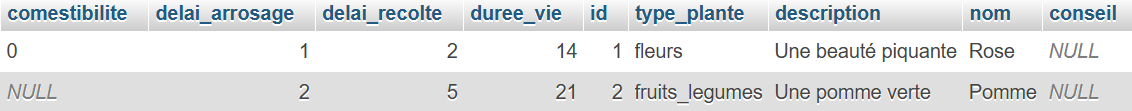
Le contenu généré par l’IA peut être incorrect.

### Table jardin :

Une image contenant texte, capture d’écran, Police, ligne

Le contenu généré par l’IA peut être incorrect.

### Table plante :



### Table utilisateur : (single table)

Une image contenant texte, capture d’écran, Police, nombre

Le contenu généré par l’IA peut être incorrect.

### Table badges\_obtenus

Une image contenant texte, Police, capture d’écran

Le contenu généré par l’IA peut être incorrect.

# Les classes

Une image contenant texte, capture d’écran, Rectangle, conception

Le contenu généré par l’IA peut être incorrect.

# Trame pour la présentation

### 1. Introduction

* Présentation rapide du projet (contexte et objectif)
* Technologies principales utilisées :
  + Frontend : Angular
  + Backend : Spring Boot
  + Sécurité : Spring Security + JWT
  + Conteneurisation : Docker
  + Intégration continue : Jenkins

### 2. Architecture Globale

* Schéma d’architecture (frontend, backend, base de données, etc.)
* Communication entre les composants (API REST, authentification…)
* Choix des patterns ou des standards (?)

### 2 bis. Gestion de projet & Design

Outils utilisés :

* + Trello : pour la gestion des tâches et le suivi agile (kanban, backlog, sprints)
  + Figma : pour les maquettes et le design UI/UX
  + (Optionnel) Autres outils : GitHub Projects, Notion, etc.
* Trello
  + Capture d’écran du tableau Trello : listes (To Do, In Progress, Done)
  + Expliquer comment l’équipe a suivi l’avancement
  + Méthode de gestion : Kaban
* Figma
  + Présentation des maquettes ou prototypes réalisés
  + Expliquer comment Figma a aidé à définir l’interface avant le développement Angular
  + Montrer la cohérence entre la maquette Figma et le rendu final Angular

### 3. Backend (Spring Boot)

* Structure des packages (controller, service, repository)
* Gestion des erreurs et validation
* Tests unitaires / tests d’intégration (si réalisés)

### 4. Frontend (Angular)

* Structure du projet Angular (modules, services, composants)
* Gestion des routes sécurisées
* Intégration avec l’API backend
* Gestion de l’authentification (intercepteur HTTP pour le token JWT, par ex.)
* Démo rapide d’une fonctionnalité importante (si possible)

### 5. Sécurité

* Méthode d’authentification choisie JWT
* Gestion des rôles et des autorisations
* Schéma explicatif du processus d’authentification
* (Optionnel) Gestion du rafraîchissement de token

### 6. Conteneurisation avec Docker

* Dockerisation du frontend (Angular) et du backend (Spring Boot)
* Présentation du docker-compose.yml
* Avantages obtenus : portabilité, environnement reproductible
* Démonstration rapide : démarrage avec Docker

### 7. CI/CD avec Jenkins

* Présentation du pipeline Jenkins :
  + Étapes (build, tests, analyse, déploiement)
  + Outils utilisés (ex : Maven, SonarQube, Nexus, etc.)
* Exemple de Jenkinsfile (simplifié pour l’explication)
* Démonstration (ou capture) d’un build réussi
* Bénéfices du pipeline : automatisation, qualité, rapidité

### 8. Défis rencontrés & Solutions

* Problèmes majeurs pendant le développement (ex : gestion CORS entre Angular et Boot, configuration sécurité, erreurs Docker)
* Comment ils ont été résolus

### 9. Conclusion

* Bilan général : ce qui a été appris
* Éventuelles améliorations futures
* Remerciements

### 10. Questions/Réponses

* Inviter l’audience à poser des questions