Université Cadi Ayyad - Marrakech
Ecole Supérieure de Technologie – Safi
Département : Informatique
Filière : Licence *ISIR*

Rapport de Projet : Simulation d'un Potager Automatisé

Réalisé par :
ARCHI Reda
NAJIM Soufiane

Encadré par :
Pr.Jamal BAKKAS

Année Universitaire 2024 - 2025

Résumé

Ce projet porte sur la conception et la réalisation d'une simulation d'un potager automatisé virtuel, développée avec le framework Spring Boot. L'objectif est de modéliser un système automatisé d'arrosage et d'application de traitements (engrais, insecticides) sur un potager constitué de parcelles carrées. La simulation gère dynamiquement les entités du potager, notamment les plantes et les insectes, avec leurs propriétés spécifiques telles que la croissance, la production, la mobilité et la résistance aux traitements.

L'architecture logicielle est organisée en trois couches : une couche métier assurant la logique de simulation et la gestion de l'évolution des entités à chaque pas de temps, une couche d'accès aux données pour la gestion des états initiaux et la persistance, ainsi qu'une couche de présentation offrant une interface de visualisation en temps réel de la simulation.

La simulation progresse par pas de temps fixes, permettant d'observer les effets des traitements et les interactions biologiques. Les résultats montrent une gestion efficace de l'humidité et un contrôle des populations d'insectes, favorisant ainsi la croissance optimale des plantes.

Ce travail représente une base solide pour le développement d'un outil intelligent de gestion de potagers automatisés, avec des perspectives futures d'intégration de modèles biologiques plus avancés et d'amélioration de l'interface utilisateur.

Table des matières

Table des figures

1 Introduction

Ce projet vise à simuler un potager automatisé en modélisant l'évolution des plantes, insectes et dispositifs de traitement (arrosage, engrais, insecticides) au sein de parcelles. La simulation s'appuie sur des règles précises et avance par pas de temps.

Pour sa réalisation, nous avons utilisé un backend Spring Boot avec JPA pour la gestion des données dans une base PostgreSQL, et un frontend React.js communiquant via Axios. Cette architecture full-stack garantit modularité, réactivité et persistance efficace.

Ce rapport présente la modélisation des entités, l'architecture technique ainsi que les résultats et perspectives du projet.

2 Description du projet

Le projet consiste à créer une simulation virtuelle d'un potager automatisé composé de parcelles carrées de taille identique. Chaque parcelle est identifiée par ses coordonnées (x, y) et caractérisée par un taux d'humidité variable. Plusieurs entités interagissent sur ces parcelles : les plantes, les insectes et les dispositifs de traitement.

2.1 Plantes

Chaque plante est définie par son espèce, son âge actuel, et son âge de maturité, à partir duquel elle commence à produire des fruits. À chaque pas de simulation, l'âge de la plante augmente d'une unité. Certaines plantes sont drageonnantes, c'est-à-dire qu'elles ont la capacité de coloniser les parcelles adjacentes selon une probabilité définie.

2.2 Insectes

Les insectes vivent en nombre illimité sur les parcelles. Ils se nourrissent des plantes présentes et peuvent se déplacer vers des parcelles voisines, se reproduire et proliférer. Chaque insecte possède des caractéristiques telles que l'espèce, le sexe, un indice de santé (entre 0 et 10), la mobilité (probabilité de déplacement) et la résistance aux insecticides (probabilité de survie aux traitements).
Lorsqu'un insecte ne trouve pas de plante pour se nourrir, il perd un point de santé à chaque pas. S'il ne se nourrit pas pendant cinq pas consécutifs, il meurt.

2.3 Dispositifs de traitement

Les dispositifs de traitement ont pour rôle d'arroser les parcelles dans un rayon donné ou d'appliquer des traitements spécifiques, tels que des engrais ou des insecticides. Chaque dispositif dispose d'un programme d'activation précisant le moment de début, la durée, et le type de traitement appliqué.

2.4 Notion de temps

La simulation est organisée par pas de temps fixes (par exemple, une minute par pas), ce qui permet de suivre l'évolution des entités dans le temps et d'observer l'impact des traitements et interactions.

3 Architecture du projet

L'architecture de notre projet de Simulation d'un Potager Automatisé est conçue selon un modèle full-stack moderne, reposant sur une séparation claire entre le frontend et le backend, avec une base de données relationnelle pour la persistance des données.

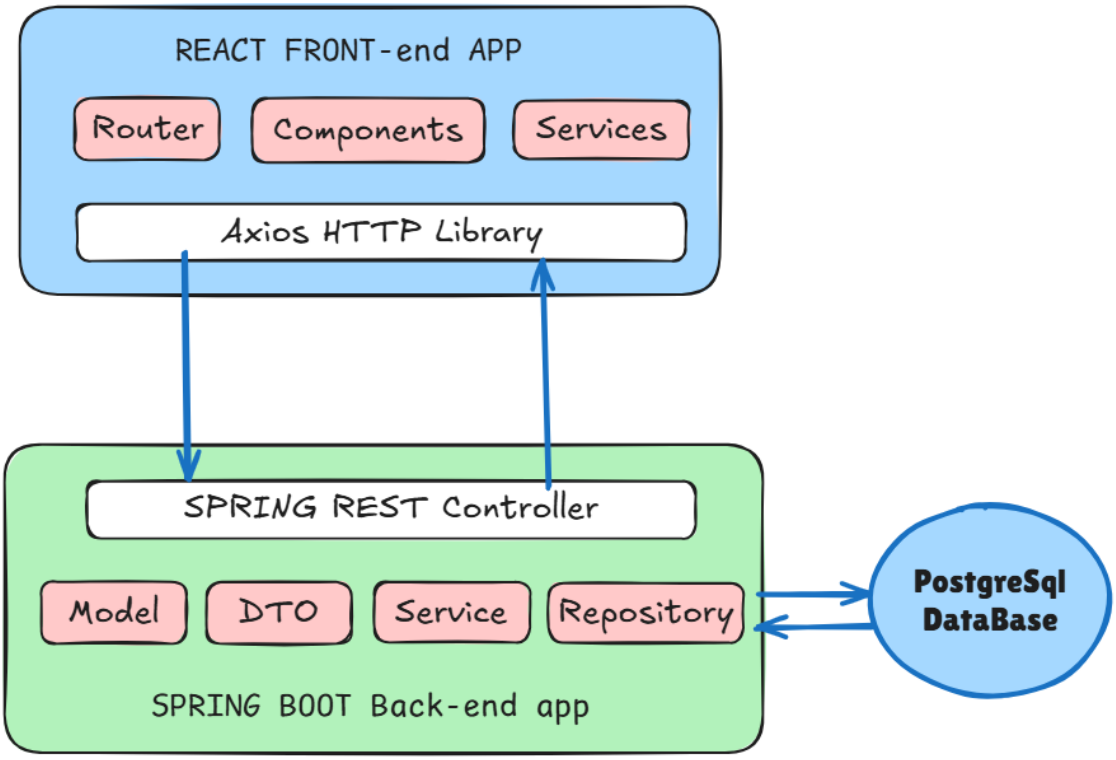

FIGURE 1 – Architecture du projet

Frontend — React.js La partie frontend est développée avec React.js, qui permet de construire une interface utilisateur dynamique et réactive. Elle est composée de plusieurs éléments clés :

Router : Gère la navigation entre les différentes pages ou vues de l'application.

Components : Regroupe les composants réutilisables de l'interface (formulaires, affichage des parcelles, tableaux de bord, etc.).

Services : Contient les fonctions pour appeler les API backend.

Axios HTTP Library : Bibliothèque JavaScript utilisée pour effectuer les requêtes HTTP vers le backend de manière asynchrone.

Cette architecture côté client permet une expérience utilisateur fluide, avec une communication efficace et sécurisée avec le serveur.

Backend — Spring Boot Le backend est développé avec Spring Boot, qui offre un cadre robuste pour créer des API REST et gérer la logique métier.

Spring REST Controller : Point d'entrée des requêtes HTTP envoyées par le frontend via Axios. Ces contrôleurs traitent les requêtes, appliquent la logique métier, et renvoient les réponses appropriées.

Model : Représente les entités du domaine métier, telles que les plantes, insectes, parcelles et traitements.

DTO (Data Transfer Objects) : Objets utilisés pour transférer les données entre le backend et le frontend, permettant de contrôler les informations échangées.

Service : Contient la logique métier et les règles de la simulation. Cette couche gère les opérations sur les entités et coordonne les interactions.

Repository : Interface avec la base de données via Spring Data JPA, permettant la persistance et la récupération des données.

Base de données — PostgreSQL La base de données relationnelle PostgreSQL stocke toutes les données persistantes liées au potager : caractéristiques des plantes, état des insectes, paramètres des parcelles, et historique des traitements appliqués. La communication entre le backend et la base se fait via les repositories JPA, qui abstraient les opérations SQL.

4 Outils utilisés

Pour la réalisation de ce projet de Simulation d'un Potager Automatisé, nous avons utilisé un ensemble d'outils et de technologies modernes, répartis entre la partie frontend, backend, base de données, et déploiement. Voici les principaux outils utilisés :

4.1 Front-end

4.1.1 ReactJs

ReactJS est une bibliothèque JavaScript développée par Facebook, permettant de construire des interfaces utilisateurs interactives, dynamiques et modulaires. Elle repose sur le concept de composants réutilisables et utilise un DOM virtuel pour optimiser les performances d'affichage. React facilite le développement d'applications monopages (SPA) en assurant une synchronisation efficace entre les données et l'interface.

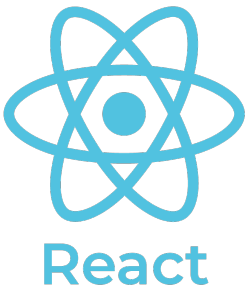

FIGURE 2 – ReactJs logo

4.1.2 Axios

Axios est une bibliothèque JavaScript permettant d'effectuer des requêtes HTTP depuis le navigateur ou depuis Node.js. Utilisée dans les applications React pour interagir avec une API, elle facilite les opérations de type GET, POST, PUT et DELETE. Axios supporte également la gestion des promesses, l'interception des requêtes/réponses, et la gestion des erreurs.

FIGURE 3 – Axios logo

4.1.3 HTML

Le HTML (HyperText Markup Language) est le langage standard utilisé pour structurer le contenu des pages web. Il permet de définir les éléments tels que les titres, les paragraphes, les images, les liens ou encore les formulaires. HTML constitue la base de toute interface web.

FIGURE 4 – HTML logo

4.1.4 CSS

Le CSS (Cascading Style Sheets) est un langage utilisé pour la mise en forme des documents HTML. Il permet de définir le style des éléments (couleurs, tailles, marges, alignements, animations, etc.) et de gérer l'apparence responsive des pages sur différents appareils. Il joue un rôle essentiel dans l'expérience utilisateur.

FIGURE 5 – CSS logo

4.2 Back-end

4.2.1 Spring Boot

Spring Boot est un framework Java open-source basé sur le framework Spring. Il simplifie la création d'applications Java autonomes, robustes et prêtes à être mises en production. Grâce à une configuration minimale et une large intégration avec d'autres modules Spring, Spring Boot accélère le développement back-end.

FIGURE 6 – Spring boot logo

4.2.2 Spring MVC

Spring MVC (Model-View-Controller) est un module de Spring qui permet de structurer les applications web en séparant clairement les responsabilités : la logique métier (Model), la présentation (View) et le contrôle des requêtes (Controller). Il facilite la création de services web RESTful dans une architecture propre et maintenable.

4.2.3 Spring Data JPA

Spring Data JPA est un sous-module de Spring qui facilite l'interaction avec des bases de données relationnelles via JPA (Java Persistence API). Il permet d'effectuer des opérations CRUD (Create, Read, Update, Delete) avec peu de code, en utilisant des interfaces de repository et en simplifiant l'écriture des requêtes.

FIGURE 7 – Spring data logo

4.3 Base de données

4.3.1 PostgreSQL

PostgreSQL est un système de gestion de base de données relationnelle open-source, reconnu pour sa robustesse, sa conformité aux standards SQL et ses performances. Il supporte les types de données avancés, les transactions, les fonctions stockées, et il est bien adapté aux applications de grande échelle.

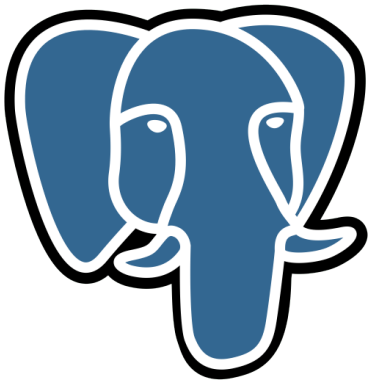

FIGURE 8 – Postgresql logo

4.4 Conteneurisation et déploiement

4.4.1 Docker

Docker est une plateforme permettant de créer, déployer et exécuter des applications dans des conteneurs. Un conteneur Docker regroupe l'application et toutes ses dépendances, assurant ainsi la portabilité, la reproductibilité et la cohérence des environnements de développement, de test et de production. Grâce à Docker, le déploiement devient plus rapide et plus fiable.

FIGURE 9 – Docker logo

5 Démonstration Visuelle

5.1 Vue initiale du potager — Step 2

La simulation commence avec l'état initial des parcelles. Chaque case de la grille affiche le taux d'humidité, le nombre de plantes et d'insectes présents. Cette image correspond au Step 2, montrant un potager fraîchement configuré avant l'évolution des entités.

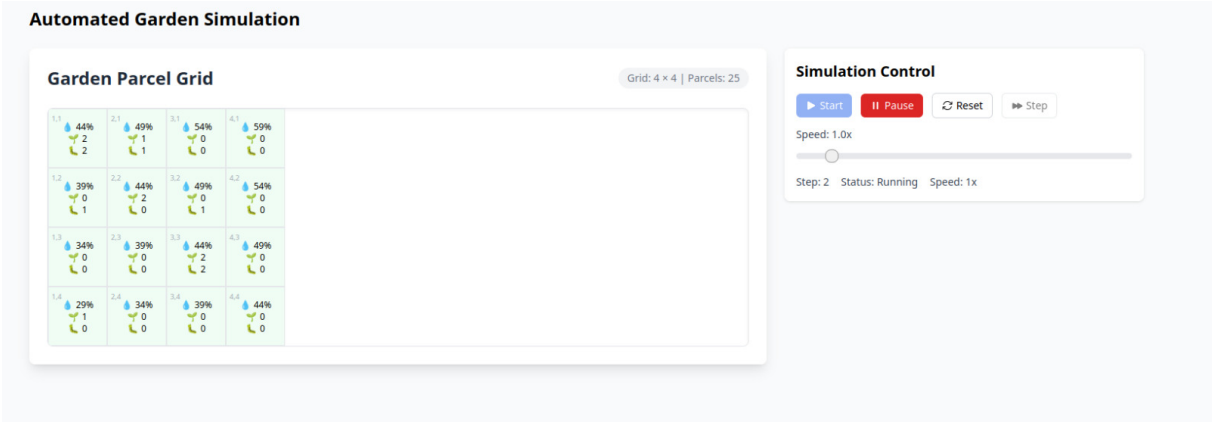

FIGURE 10 – Step 2 — État initial du potager

5.2 Évolution intermédiaire — Step 9

À ce stade intermédiaire, Step 9, on observe les premiers changements : certains taux d'humidité ont diminué, la croissance des plantes est visible avec plus de jeunes pousses, et les populations d'insectes évoluent. La simulation est en cours d'exécution avec une vitesse normale.

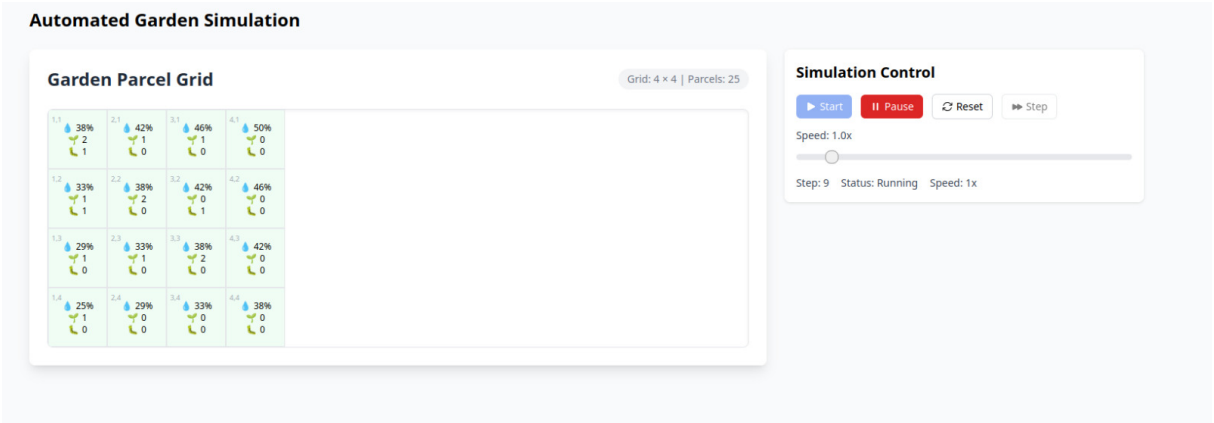

FIGURE 11 – Step 9 — Début de l'évolution

5.3 Progression avancée — Step 11

La grille à Step 11 illustre une progression notable de la simulation. Le taux d'humidité continue de varier, les plantes ont gagné en nombre, et la dynamique entre insectes et plantes est plus marquée, traduisant les interactions biologiques modélisées.

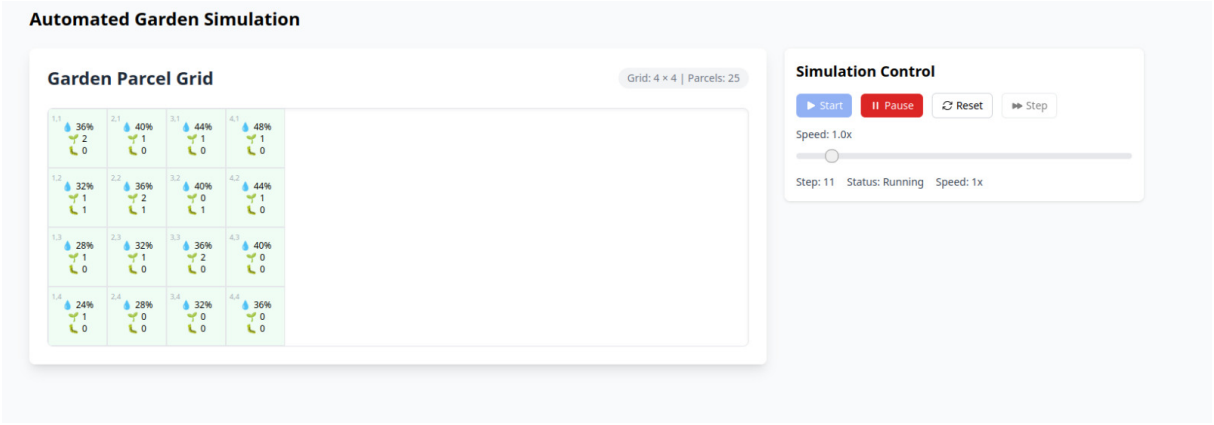

FIGURE 12 – Step 11 — Croissance des entités

5.4 Accélération de la simulation — Step 22

Au Step 22, la vitesse a été augmentée pour accélérer la simulation. Les changements sont plus rapides, avec des taux d'humidité plus fluctuants et des populations d'entités qui se développent plus visiblement.

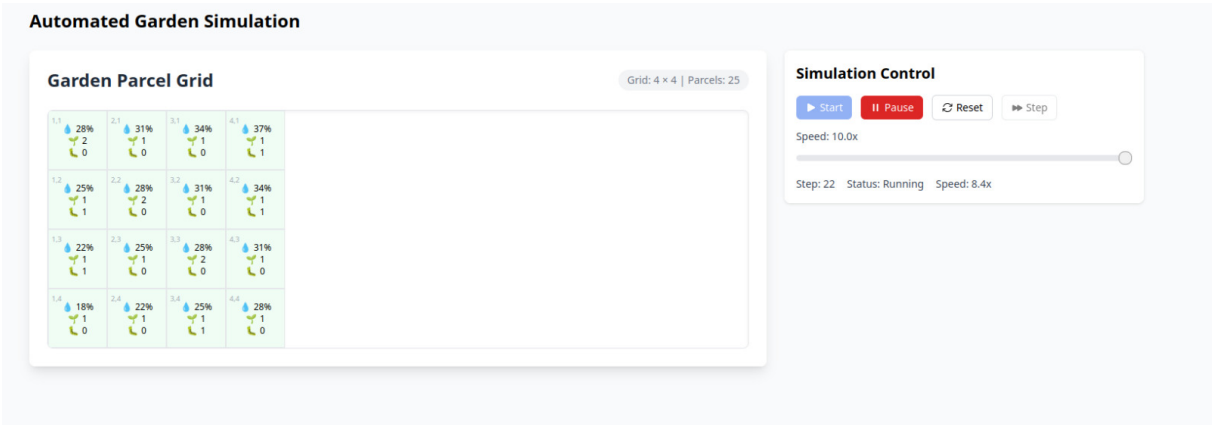

FIGURE 13 – Step 22 — Simulation accélérée

5.5 Phase avancée — Step 26

L'état à Step 26 présente une évolution poussée : plusieurs parcelles affichent des taux d'humidité plus faibles, la répartition des plantes et insectes varie selon les traitements appliqués, indiquant les effets de la simulation sur le potager.

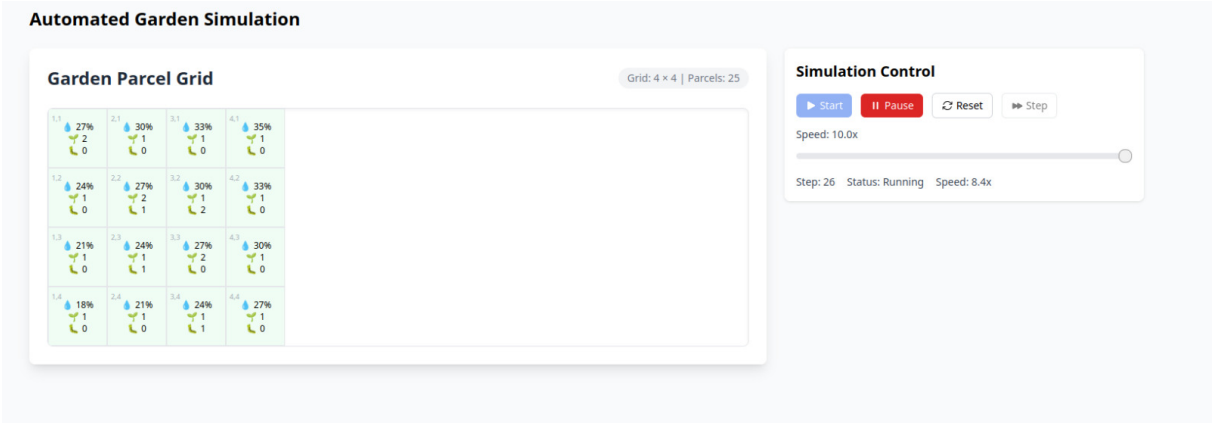

FIGURE 14 – Step 26 — Phase avancée

5.6 Dernières étapes — Step 28 à Step 34

Les images de Step 28 jusqu'à Step 34 montrent la simulation proche de sa phase finale. Les taux d'humidité ont tendance à baisser, la croissance des plantes est visible avec davantage de pousses et de maturité, et les insectes continuent de se déplacer et proliférer, reflétant un système dynamique et interactif.

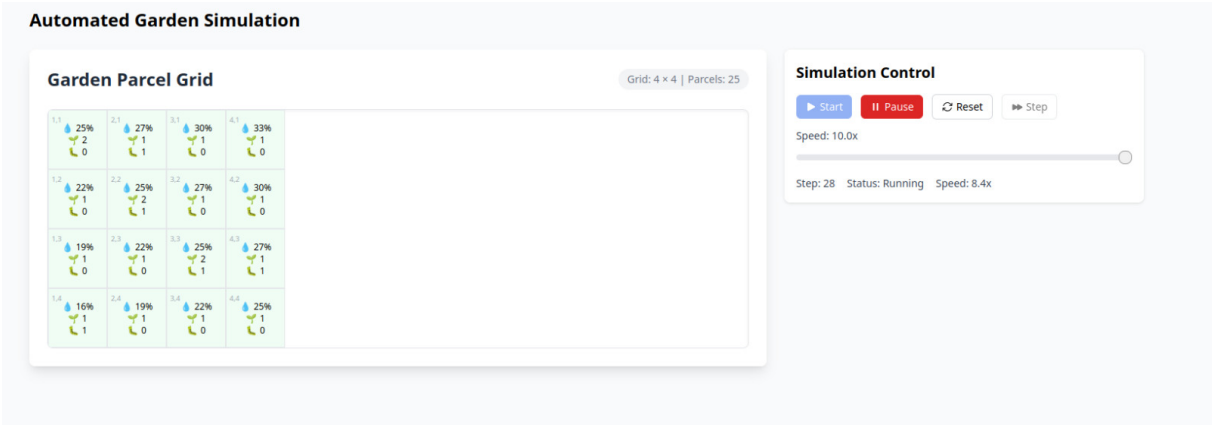

FIGURE 15 – Step 28 — Simulation proche de la fin

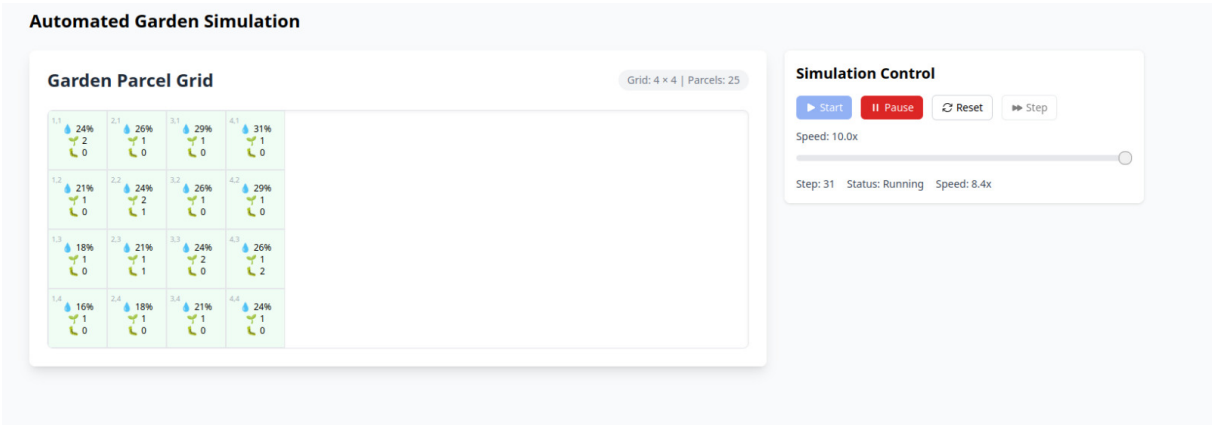

FIGURE 16 – Step 31 — État évolué du potager

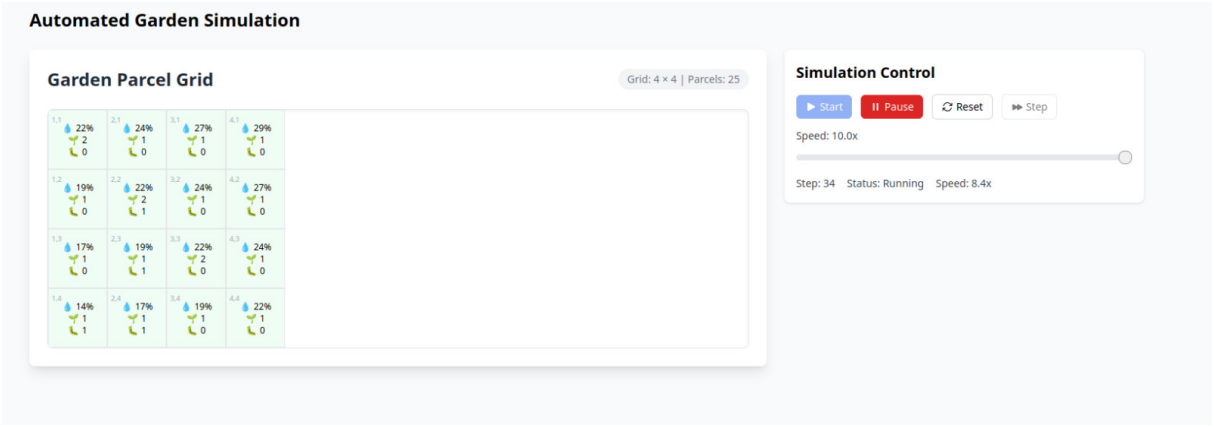

FIGURE 17 – Step 34 — Phase finale de simulation

6 Conclusion

Ce projet de simulation d'un potager automatisé a permis de concevoir et de développer une application complète, alliant modélisation précise des entités du potager (plantes, insectes, dispositifs de traitement) et architecture logicielle moderne. L'utilisation conjointe de Spring Boot pour le backend et de React.js pour le frontend a assuré une séparation claire des responsabilités, une communication efficace via des API REST, et une interface utilisateur réactive et intuitive.

Grâce à la base de données PostgreSQL et à l'intégration de Docker pour la conteneurisation, l'application est robuste, évolutive et facilement déployable dans différents environnements.

Les résultats obtenus démontrent la pertinence de la simulation pour évaluer l'efficacité des systèmes d'arrosage et de traitements, tout en offrant une base solide pour des extensions futures, telles que l'ajout de modèles biologiques plus complexes, l'intégration de capteurs réels, ou encore le développement d'outils d'analyse avancée.

En somme, ce travail représente une étape importante vers la mise en place d'outils intelligents de gestion agricole automatisée, adaptés aux besoins actuels de durabilité et d'optimisation des ressources.

Références

— Spring boot Documentation : https://docs.spring.io/spring-boot/index.html

— Spring Data Documentation : https://docs.spring.io/spring-data/jpa/reference/index.html

— ReactJs Documentation : https://react.dev/learn

— Axios Documentation : https://axios-http.com/docs/intro

— TutorialsPoint - Docker Tutorial : https://www.tutorialspoint.com/docker/index.htm

— Docker Curriculum : https://docker-curriculum.com/