CONTEXT DIAGRAM

A diagram of a computer network

AI-generated content may be incorrect.

Players:

Send and receive game actions and updates.

Join or leave games.

Participate in voice chat.

Web Browser:

Displays the game interface.

Handles user inputs and sends them to the game server.

Receives and displays game updates.

WebSocket Server:

Facilitates real-time communication between the game server and players.

Handles WebRTC signaling for voice chat.

Game Server:

Processes game logic and updates.

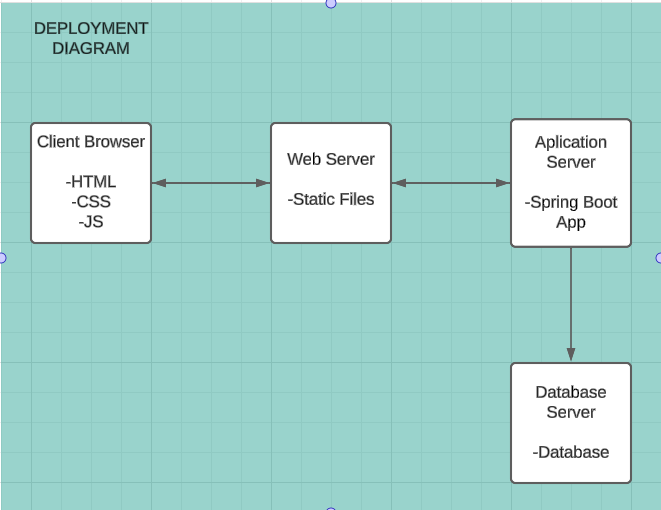
Manages player sessions and game states.

Interacts with the database to store and retrieve game data.

Database:

Stores persistent game data.

Provides data retrieval for game state management.



Nodes:

Client Browser: Runs the game interface and handles user interactions.

Web Server: Hosts the static files (HTML, CSS, JavaScript) and serves them to the client.

Application Server: Runs the Spring Boot application, handles game logic, WebSocket communication, and interacts with the database.

Database Server: Stores persistent game data.

Artifacts:

Static Files: HTML, CSS, JavaScript files served by the web server.

Spring Boot Application: The backend application running on the application server.

Database: The database schema and data.

A diagram of a game

AI-generated content may be incorrect.

Entities

Player: Represents a player in the game with attributes such as name, type, powerUpCount, stamina, and infectedCount.

Game: Represents a game session with attributes such as state, gameCode, startTime, and an array of players.

GameBoard: Represents the game board with attributes such as gameId (reference to Game) and boardState.

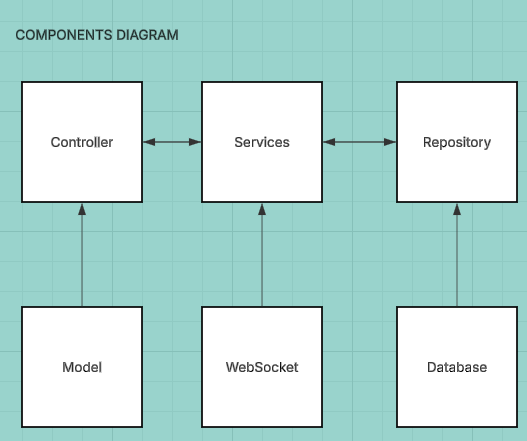
WebSocketMessage: Represents messages sent over WebSocket with attributes such as gameId (reference to Game), messageType, and content.

Relationships

Player and Game: Many-to-many relationship, where a game can have multiple players and a player can participate in multiple games.

Game and GameBoard: One-to-one relationship, where each game has one game board.

Game and WebSocketMessage: One-to-many relationship, where a game can have multiple WebSocket messages.



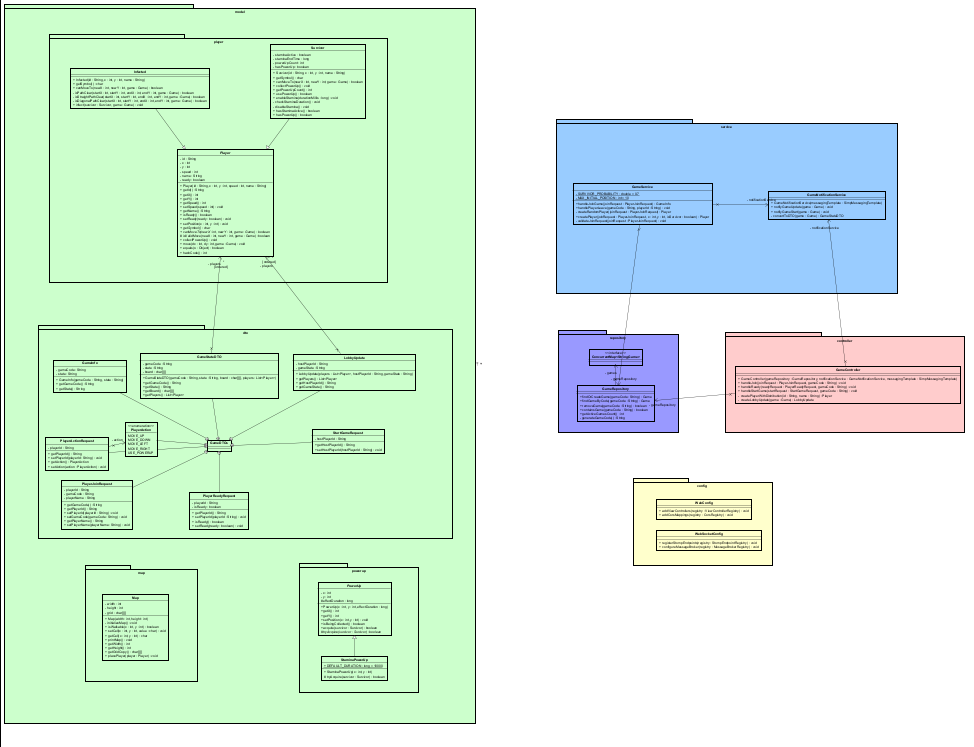
Controller: Handles HTTP requests and WebSocket connections.

Service: Contains the business logic of the application.

Repository: Manages data access and storage.

Model: Represents the data structures used in the application.

WebSocket: Manages WebSocket communication.



Controller classes depend on Service classes to process requests.

Service classes depend on Repository classes to access and manipulate data.

Model classes are used across Controller, Service, and Repository classes to represent the data.