

Relazione PCD-Assignment-3

Studente: Manuel Buizo

Corso: Programmazione Concorrente e Distribuita

Anno Accademico: 2023/2024

Indice

1. [Introduzione](#)
2. [Progetto 1 – Cars Simulation](#)
3. [Progetto 2 – Sudoku MOM](#)
4. [Progetto 3 – Sudoku Java-RMI](#)
5. [Progetto 4 – Guess The Number](#)
6. [Conclusioni](#)
7. [Riferimenti](#)

Introduzione

La presente relazione descrive quattro progetti sviluppati nell'ambito dell'esame di

[Programmazione Concorrente e Distribuita].

Ciascun progetto è stato sviluppato con l'obiettivo di analizzare un problema specifico e progettare una soluzione software appropriata, sfruttando le tecnologie trattate durante le lezioni.

Breve panoramica dei progetti:

1. **[Cars Simulation]** – Simulazione multi-agente di automobili basata sul paradigma ad attori con Akka.
2. **[Sudoku MOM]** – Implementazione distribuita di Sudoku tramite Message-Oriented Middleware (MOM) e scambio asincrono di messaggi.
3. **[Sudoku Java-RMI]** – Sviluppo di Sudoku con Distributed Object Computing e comunicazione remota tramite Java RMI.
4. **[Guess The Number]** – Gioco Guess the Number basato su scambio sincrono di messaggi, implementato in Go.

Progetto 1 – Cars Simulation

Analisi del problema

Partendo

dall'[Assignment 1](#)

relativo alla simulazione delle macchine, il progetto estende la gestione del sistema

adottando il paradigma ad attori tramite il framework Akka in Scala. In tale modello, ogni automobile è rappresentata da un attore autonomo, responsabile della gestione del proprio stato, del comportamento e delle interazioni con l'ambiente e con gli altri attori. La simulazione complessiva è coordinata attraverso la comunicazione asincrona tra attori, consentendo una gestione concorrente e scalabile del sistema.

Punti critici:

- Gestione del muro per la sincronizzazione e il coordinamento delle automobili durante la simulazione.
- Gestione degli step temporali e dello stato di ciascuna automobile a ogni iterazione, mediante l'utilizzo di attori Akka.
- Gestione della comunicazione tramite messaggi per l'integrazione e l'interazione con l'interfaccia grafica (GUI).

Architettura proposta

La simulazione viene creata come un singolo attore che orchestra le automobili ovvero gli attori figli. Gestisce al suo interno il tempo della simulazione facendo passi per un tempo discreto ovvero un delta t e un opzionale temporizzazione tramite un muro per sincronizzare (es: come fps nei giochi) a un tempo costante a simulaizone, ovvero il comportamento della macchine. Ogni macchina viene modellata tramite un singolo attore che gestirà il proprio stanto, non che eseguirà le fasi del suo comportamento, ossia sense, decide e act. Per ogni step della simulazione le macchine eseguiranno tutte le sue fasi in modo non deterministico e dopo che ogni attore della macchine finisce notifica alla simulazione tramite messaggio, alla fine dell'esecuzione di tutte le macchine la simulazione fa partire il prossimo step al tempo t + dt. Oltre alla fase di sincronozzazione per l'esecuzione degli step, l'attore della simulazione si sincronizza con le macchine eseguendo una fase si setup per inizializzare i dati.



Syntax error in graph
mermaid version 9.1.7

Spiegare la struttura del sistema e le componenti principali.
Si possono inserire diagrammi UML, schemi architetturali o diagrammi dei flussi.

Esempio di punti chiave:

- Architettura a 3 livelli (frontend, backend, database)
- API REST per la comunicazione
- Gestione autenticazione via JWT

Diagramma Architettura


Tecnologie utilizzate

Componente	Tecnologia	Ruolo
Frontend	React.js	Interfaccia utente
Backend	Node.js + Express	API REST e logica applicativa
Database	MongoDB	Gestione dati
Autenticazione	JWT	Sicurezza e session management

Sviluppo

Breve descrizione del processo di sviluppo:

- Struttura del progetto e principali moduli
- Funzionalità implementate
- Difficoltà tecniche e soluzioni adottate

Diagramma di flusso o sequenza

Risultati e considerazioni

Sintesi dei risultati ottenuti, test effettuati e possibili miglioramenti futuri.

Esempio:
L'obiettivo del progetto è sviluppare una web app per la gestione delle attività giornaliere, accessibile da desktop e mobile.

Progetto 2 – Sudoku MOM

(stessa struttura compatta del Progetto 1)

Progetto 3 – Java-RMI

(stessa struttura compatta del Progetto 1)

Progetto 4 – Guess The Number

(stessa struttura compatta del Progetto 1)

Conclusioni

Riflessione generale sull'intero percorso:

- Competenze tecniche acquisite
- Tecnologie più efficaci o interessanti
- Difficoltà comuni e soluzioni riutilizzabili
- Possibili sviluppi futuri dei progetti

Riferimenti

- [1] Documentazione ufficiale delle tecnologie utilizzate
- [2] Tutorial o articoli di riferimento
- [3] Altri materiali consultati