Relazione Fase 1 - ISS25

Informazioni Generali

Nome: Nicolò Venieri **Repository GIT:** Conway_nicolovenieri

Finalità della Prima Fase e Aspettative per le Fasi Successive

Durante la prima fase, sono stati sviluppati e testati diversi sistemi, tra cui il **Gioco della Vita di Conway**, implementato utilizzando **Java** e **Spring Boot**. Questo progetto ha permesso di comprendere il funzionamento di un sistema distribuito e la sua interazione tramite protocolli di comunicazione. Per le fasi successive, ci aspettiamo di raggiungere un sistema distribuito su ogni "edge device", dove l'elaborazione del servizio non sarà più centralizzata.

Sistemi Realizzati e Sperimentati

Durante queste settimane di lezione, ho testato e implementato il codice fornito dal docente, approfondendo le logiche implementative e il loro ruolo funzionale nel sistema in sviluppo. Ho sperimentato in modo efficace la comunicazione tramite **MQTT**, valutandola come un concetto chiave per rendere l'architettura più indipendente dalle **WebSocket**. Ritengo che questa comprensione sia essenziale per affrontare la seconda fase dello sviluppo dell'applicazione.

Abilità e Competenze Acquisite

- **Utilizzo di Java con i principi di ingegneria del software**: Ho appreso l'uso di **Java** applicando concetti fondamentali dell'ingegneria del software, come il **Factory Pattern** per una gestione più modulare della creazione degli oggetti e il **Pattern Observer** per implementare una comunicazione reattiva tra componenti.
- **Gestione delle dipendenze con Gradle**: Ho imparato a utilizzare **Gradle** per la gestione delle dipendenze e l'automazione del sistema di build, semplificando l'integrazione di librerie e framework nel progetto.
- **Introduzione a Spring Boot**: Ho acquisito familiarità con **Spring Boot**, comprendendo come facilita la creazione di applicazioni enterprise grazie alla sua configurazione automatica e alla gestione dei componenti.
- **Comunicazione tramite MQTT**: Ho sperimentato l'uso del protocollo **MQTT** per la comunicazione tra componenti, scegliendolo come alternativa alle **WebSocket** per garantire un'architettura più scalabile, indipendente e adatta a scenari di comunicazione asincrona.

Motivazione della Scelta del "Gioco della Vita di Conway"

A mio parere, il **Gioco della Vita di Conway** è stato scelto perché presenta regole semplici, riducendo la complessità della fase di analisi del problema e permettendo di concentrarsi direttamente sull'implementazione.

Inoltre, lo sviluppo del gioco basato su celle si presta bene al concetto di **sistema distribuito** e alla **divisione dei compiti**. Ritengo che in futuro si potrebbe arrivare a delegare a dispositivi specifici la gestione di una singola cella o di piccoli gruppi di celle all'interno di una grande griglia, riflettendo un'architettura distribuita su larga scala.

Scelta di Java e Spring Boot

Spring Boot è ideale per lo sviluppo di sistemi distribuiti grazie alla sua capacità di gestire microservizi in modo scalabile e modulare. Offre configurazioni predefinite, supporto nativo per MQTT e integrazione con strumenti come Docker e Kubernetes. La Dependency Injection (DI) migliora la manutenibilità del codice. Infine, essendo basato su Java, garantisce stabilità, compatibilità e un ampio ecosistema di librerie.

Aspetti di Ingegneria del Software

Durante lo sviluppo del progetto, sono stati applicati concetti chiave dell'ingegneria del software. Il principio della **Single Responsibility** è stato utilizzato per assicurare che ogni classe avesse una sola responsabilità. Inoltre, il **Pattern Observer** è stato impiegato per gestire la comunicazione tra oggetti in modo reattivo, mentre il **Factory Method** ha permesso di creare oggetti in modo flessibile senza dipendere direttamente dalle classi concrete. Infine, l'**Inversione delle dipendenze** è stata implementata per separare le dipendenze tra i componenti.

Il Sistema sviluppato è un sistema distribuito basato su microservizi?

Il sistema sviluppato adotta un'architettura a microservizi, ma attualmente non può essere considerato un sistema distribuito completo. Siamo in una fase di transizione in cui i microservizi gestiscono la GUI, la comunicazione e la logica di gioco. Per definirlo un sistema distribuito, ogni singolo "nodo" dovrebbe essere parte di una rete funzionale, operando in modo indipendente dagli altri. Inoltre, il sistema dovrebbe garantire il funzionamento del servizio anche in caso di malfunzionamento di uno o più nodi.

Ruolo delle Librerie "Custom"

L'adozione di librerie personalizzate, come `unibo.basicomm23-1.0.jar`, ha facilitato l'astrazione nella gestione della comunicazione tramite MQTT. Questo approccio ha garantito una maggiore flessibilità e riusabilità del codice, permettendo di elevare il livello di astrazione e semplificando l'integrazione di nuove funzionalità. L'uso di tali librerie ha contribuito a rendere il sistema più modulare e manutenibile.

Ruolo del Concetto di Linguaggio

Il concetto di linguaggio rispetto all'utilizzo di librerie rappresenta un salto evolutivo, garantendo vantaggi significativi in vari aspetti:

- **Astrazione**: A differenza delle librerie, un linguaggio personalizzato permette di avere un controllo maggiore e di creare nuove semantiche e sintassi più vicine

alla soluzione del problema.

- **Comportamento e Semantica**: Consente di definire nuove regole, comportamenti e modelli di esecuzione in base alle necessità del progetto.
- **Specializzazione**: Un linguaggio permette di creare soluzioni specifiche e, allo stesso tempo, facilmente manutenibili, a differenza delle librerie che offrono strumenti generici.
- **Evoluzione**: Risolvere un problema tramite librerie implica l'uso di strumenti già esistenti, mentre con un linguaggio personalizzato si può creare una soluzione su misura senza dover necessariamente avvalersi di strumenti preesistenti.