Relazione Fase 1 ISS25

Nella prima parte del corso si è preso come modello di partenza **Conway Life**, gioco sviluppato dal matematico John Conway verso la fine degli anni sessanta, in cui il proseguimento dello stesso è determinato esclusivamente dal suo stato iniziale, senza necessità di alcun input da parte di giocatori umani, .

La prima implementazione ha visto l'utilizzo di Javascript, linguaggio più moderno e avanzato che permette un codice semplice e funzionale senza l'utilizzo di alcun server per la visualizzazione e il controllo della griglia.

Successivamente si è passati ad un'implementazione su Java:

Il lavoro è stato suddiviso in diverse fasi:

- **Individuazione dei componenti** per creare un modello di gioco.
- **Sviluppo e introduzione ai test** per verificare il corretto funzionamento del modello.
- **Organizzazione in packages**, che ha reso necessaria l'introduzione di un dispositivo di output.

A questo proposito, è stata definita l'interfaccia **IOutDev**. Questa interfaccia funge da contratto che ogni dispositivo di output deve rispettare, implementando un metodo per visualizzare una cella in modo appropriato.

Indipendentemente dall'effettivo dispositivo di output scelto, l'interfaccia garantisce un'astrazione utile e facilita la gestione della visualizzazione. Attualmente, l'implementazione dell'interfaccia mostra l'output sul terminale.

Successivamente è stato introdotto **SpringBoot** con l'obiettivo della creazione di un servizio che permettesse a più utenti di connettersi alla pagina HTML, senza obbligatoriamente avere permessi owner e rendere visibili eventuali progressi nel gioco.

E' stato quindi effettuato un refactoring del codice utilizzato affinché il modello prevedesse le celle come degli oggetti Java e non come semplici numeri e infine implementata la logica Spring per creare la base e utilizzato Gradle per gestire le dipendenze.

Si è arrivati quindi alle considerazioni per cui Il codice Java inserito come parte integrante del servizio:

- ha la stessa struttura introdotta in ConwayLife25 in Java ma con un diverso LifeController
- la classe Life non ha il compito di visualizzare le celle, compito assunto dal ConwayGuiControllerLifeLocal
- elimina ogni dispositivo di input, in quanto i comandi-utente vengono inviati tramite WebSocket
- definisce un dispositivo di output WSIoDev che implementa IOutDev interface inviando messaggi alla pagina HTML tramite WebSocket.

L’approccio adottato è simile a un’**armatura di Iron-Man**, in cui il progetto viene incapsulato in una struttura preesistente, modificandolo il meno possibile per garantire stabilità e funzionalità. Alla base dell’approccio c’è l’idea che se un sistema già funziona, non ha senso rivoluzionarlo radicalmente.

Spring Boot è stato scelto per la sua semplicità e capacità di nascondere complessità dietro annotazioni, riducendo le righe di codice necessarie per ottenere un servizio funzionante.

La combinazione di Java con Spring Boot, WebSocket e JavaScript è risultata efficace per rapidità di sviluppo e semplicità rispetto ad altre soluzioni come JavaFX o Swing. Inoltre, una pagina HTML rappresenta uno strumento universale indipendente dalla piattaforma hardware su cui viene visualizzata.

Sono stati poi contrapposti due diversi approcci:

- L’utilizzo di funzioni standard della libreria Websocket
- L’utilizzo di funzioni di una libreria custom.

Le librerie custom rivestono un ruolo fondamentale nello sviluppo software, poiché permettono di:

- **Standardizzare i processi**, nascondendo funzioni complesse dietro interfacce intuitive.
- **Centralizzare la logica comune**, riducendo la ridondanza e migliorando la manutenibilità.
- **Aumentare la riusabilità**, facilitando l’integrazione in progetti diversi.

La scelta dell’utilizzo di librerie “personalizzate” ha infatti garantito un'architettura più modulare e flessibile, facilitando eventuali miglioramenti futuri.

Software Utilizzati:

Docker

Docker è una piattaforma che semplifica il deployment delle applicazioni, permettendo di distribuirle in modo efficiente. Il processo avviene attraverso la creazione di immagini immutabili basate su un Dockerfile, che definisce l’ambiente e le dipendenze necessarie. Queste immagini possono essere eseguite in container, ambienti isolati che garantiscono una maggiore portabilità e riproducibilità del software.

Interazioni M2M

Il sistema prevede l’uso di conwayCallerWs, un servizio che si connette via WebSocket per inviare e ricevere messaggi. Per garantire la sicurezza, abbiamo introdotto la protezione ownerOff, che consente solo all’owner di selezionare celle, avviare e arrestare il gioco, prevenendo accessi non autorizzati.

Nel modulo conwayCallerWsInteraction, le funzioni di comunicazione sono implementate attraverso la libreria personalizzata unibo.basicomm23. Questo permette di astrarre i dettagli dell'uso di javax.websocket, semplificando l'integrazione del protocollo WebSocket.

MQTT

Per lo scambio di informazioni tra i micro-servizi, utilizziamo il protocollo MQTT. In particolare, ConwayGui opera come micro-servizio indipendente, connesso al resto del sistema tramite un broker MQTT. La comunicazione segue il modello Publisher/Subscriber, dove i componenti pubblicano e ricevono messaggi attraverso topic specifici.

Conclusioni

L'obiettivo iniziale del corso è stato comprendere come sviluppare un programma partendo da una consegna, adattandolo ed evolvendolo in base alle necessità emergenti.

Il percorso di sviluppo ha seguito queste fasi principali:

1. **Implementazione iniziale in JavaScript** una soluzione semplice e immediata senza dipendenze esterne.

2. **Trasposizione in Java** per una gestione più strutturata e scalabile.

3. **Refactoring per l'integrazione con Spring Boot** per trasformare il progetto in un servizio web.

4. **Introduzione di librerie custom e WebSocket** per standardizzare la comunicazione e migliorare l'interazione tra componenti.

5. **Adozione di Docker e MQTT** per sperimentare tecnologie di containerizzazione e protocolli di comunicazione asincrona.

Inoltre durante questa fase, sono stati sperimentati diversi protocolli di comunicazione analizzandone vantaggi e svantaggi, e implementato un'interazione M2M per testare l'automazione tra microservizi.