Relazione fine Fase2-ISS25

Jacopo Chergui

---

1. Finalità e attese future

Durante questa seconda fase, ho compreso l'importanza della modellazione ad attori tramite Qak per costruire sistemi software distribuiti e scalabili.

Mi aspetto che le prossime fasi approfondiscano il passaggio da sistemi prototipali a veri e propri microservizi cloud-native, integrando CI/CD, orchestrazione (es. Kubernetes) e gestione del ciclo di vita del software.

---

2. Sistemi costruiti e attività

Abbiamo lavorato su attori che comunicano tramite MQTT per simulare celle di Conway distribuite.

Ogni cella è un attore con logica locale, che osserva i vicini tramite topic MQTT, e prende decisioni sul proprio stato.

Ho realizzato, eseguito e testato localmente un sistema distribuito multi-attore sfruttando anche un broker MQTT (mosquitto).

---

3. Nuove competenze apprese

Ho consolidato competenze su:

- progettazione ad attori
- uso di MQTT per sistemi distribuiti
- gestione della concorrenza e sincronizzazione
- progettazione bottom-up di un sistema software interattivo e scalabile

Inoltre, ho imparato a tenere separati modello (Qak) e implementazione generata (Kotlin).

---

4. Gioco della vita come esempio di sistema

Il gioco della vita è stato un primo esempio efficace: ogni cella opera con logica locale ma interagisce in modo distribuito.

Il sistema ha richiesto sincronizzazione tra attori, scambio messaggi e tolleranza a eventuali ritardi.

Questo approccio ha reso chiaro il potenziale e le sfide della distribuzione.

---

5. Scelte di sviluppo e tecnologie

L'uso di Java è stato positivo per interoperabilità con librerie MQTT, ma in un contesto moderno valuterei alternative più reattive e leggere, come Kotlin puro o Go.

Inoltre, l'integrazione con Docker semplificherebbe il testing e la distribuzione.

---

6. Aspetti di Ingegneria del Software

Tra i key points:

- Separazione dei ruoli tra attori
- Modellazione Qak per ridurre complessità
- Uso di broker per disaccoppiare componenti
- Progettazione incrementale test-driven

Tutti questi elementi riflettono buone pratiche di ingegneria del software.

---

7. Librerie custom e gestione MQTT

Durante la fase ho riutilizzato classi Java per la gestione MQTT (OutInMqttForActor.java) e la logica del gioco (Life.java).

Queste classi sono state incapsulate nella libreria unibo.basicomm23-1.0.jar per favorire riuso e manutenibilità.

---

8. Linguaggio come salto evolutivo

Qak non è solo un linguaggio tecnico, ma un modo diverso di pensare i sistemi: permette di modellare entità autonome, reattive e intelligenti.

È un vero salto rispetto ai linguaggi imperativi, ponendo il focus sulla cooperazione tra attori piuttosto che sul controllo centralizzato.

---

9. Oggetto di auto-valutazione

Ritengo questa fase molto formativa: ho imparato concetti chiave della progettazione distribuita, anche se a volte il debugging e la sincronizzazione hanno richiesto più tempo del previsto.

Mi sento più sicuro nella gestione di architetture a eventi e voglio approfondire il deployment con Docker e strumenti di orchestrazione.