

CityTwin

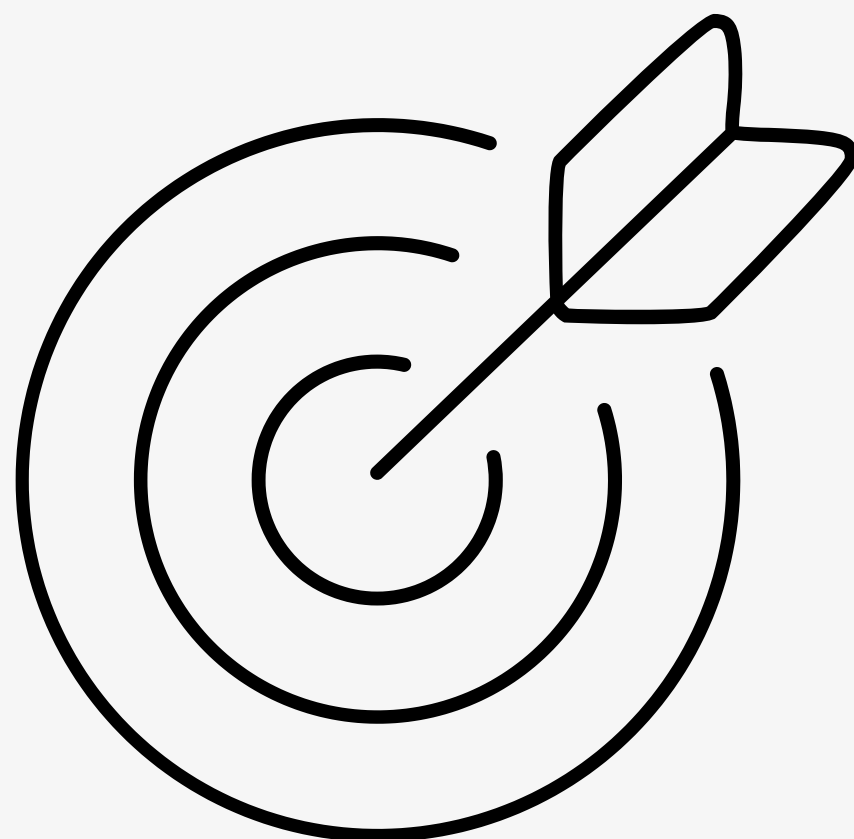


Eddie Barzi
Filippo Vissani

Scopo del software

Il progetto "CityTwin" mira a creare una **piattaforma di simulazione e monitoraggio** delle smart city.

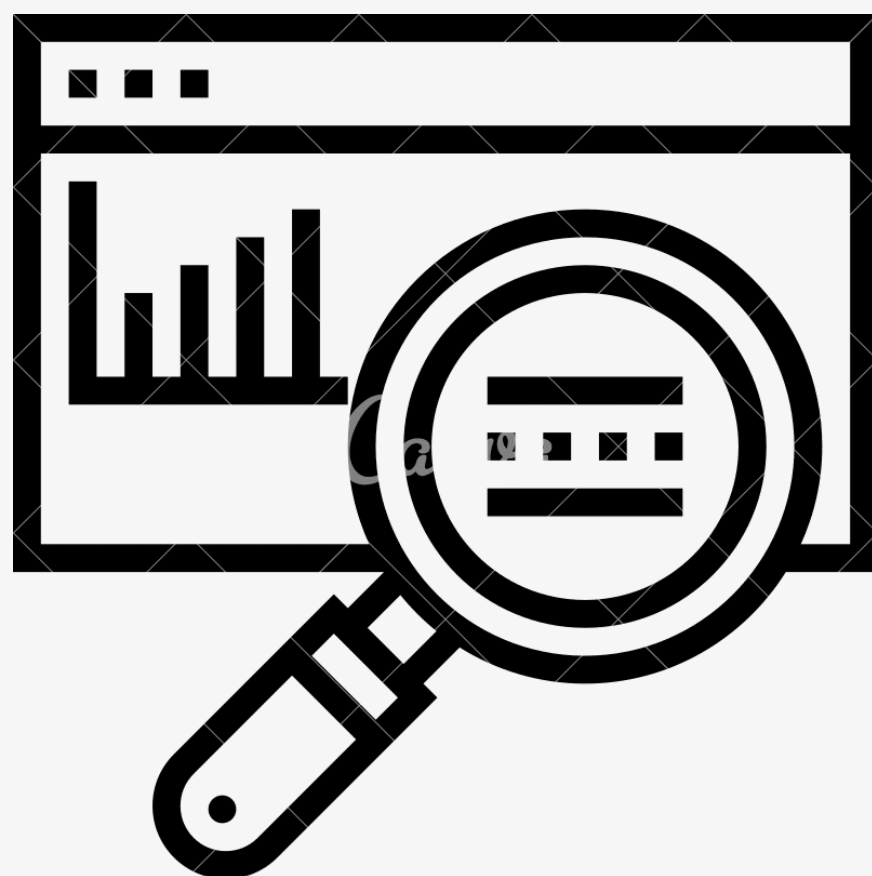
L'obiettivo è migliorare l'efficienza operativa delle città intelligenti attraverso la **rappresentazione digitale delle entità** e delle loro interazioni.



```
filippo@nixos:~/Documents/repositories/ds-project-vissani-barzi-ay2223
~/Documents/repositories/ds-project-vissani-barzi-ay2223 master !1 11:09:10
➔ ./scripts/startup.sh
```

```
filippo@nixos:~/Documents/repositories/ds-project-vissani-barzi-ay2223 <2>
~/Documents/repositories/ds-project-vissani-barzi-ay2223 master !1 11:09:56
➔ docker compose up
```

Moduli monitor

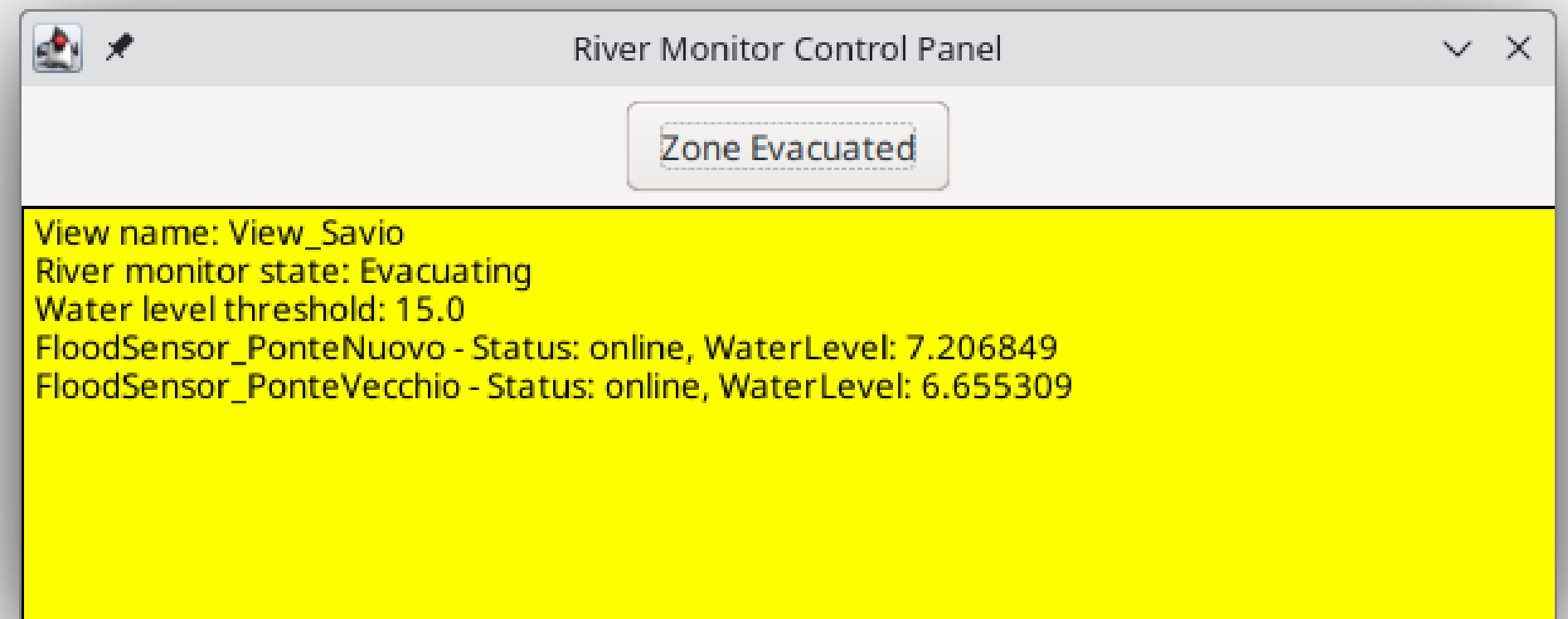
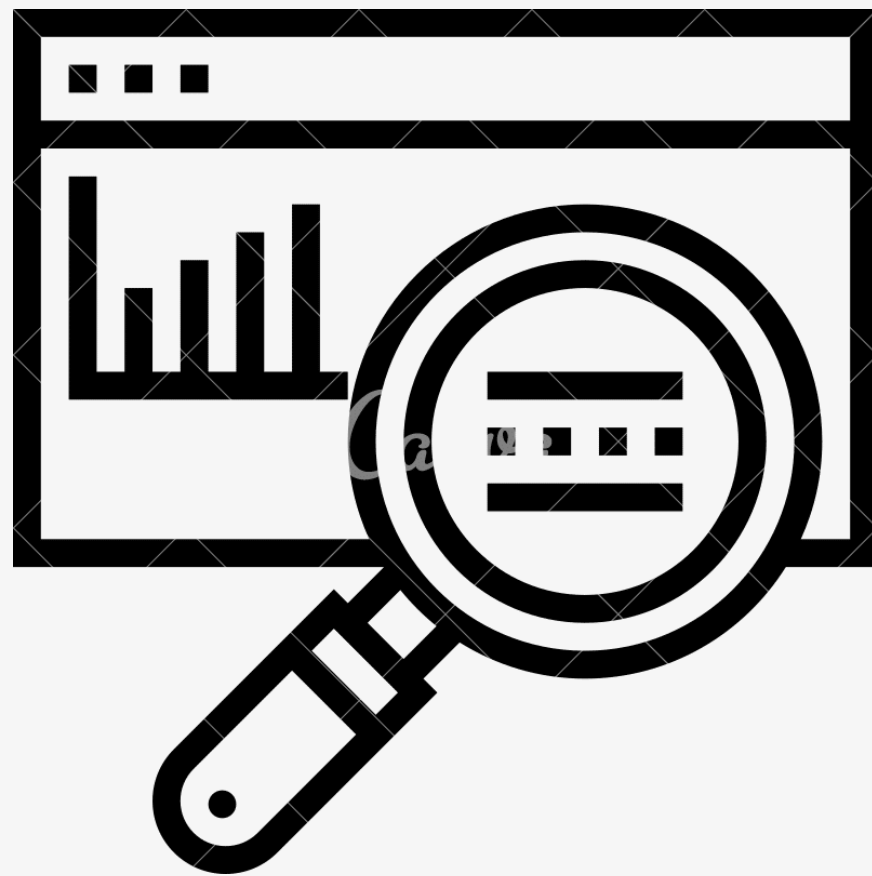


Sono stati implementati i seguenti moduli di monitoraggio all'interno del sistema:

- Sensore per l'***analisi della qualità dell'aria***.
- Sensore per la rilevazione di ***piogge acide***.
- Sensore per la rilevazione dell'***inquinamento acustico***.
- Sistema di ***monitoraggio di un fiume***, con possibilità di intervento in caso di allagamento.

River monitor

L'utente sarà in grado di interagire con eventuali moduli di monitoraggio.



Interazione tra moduli monitor



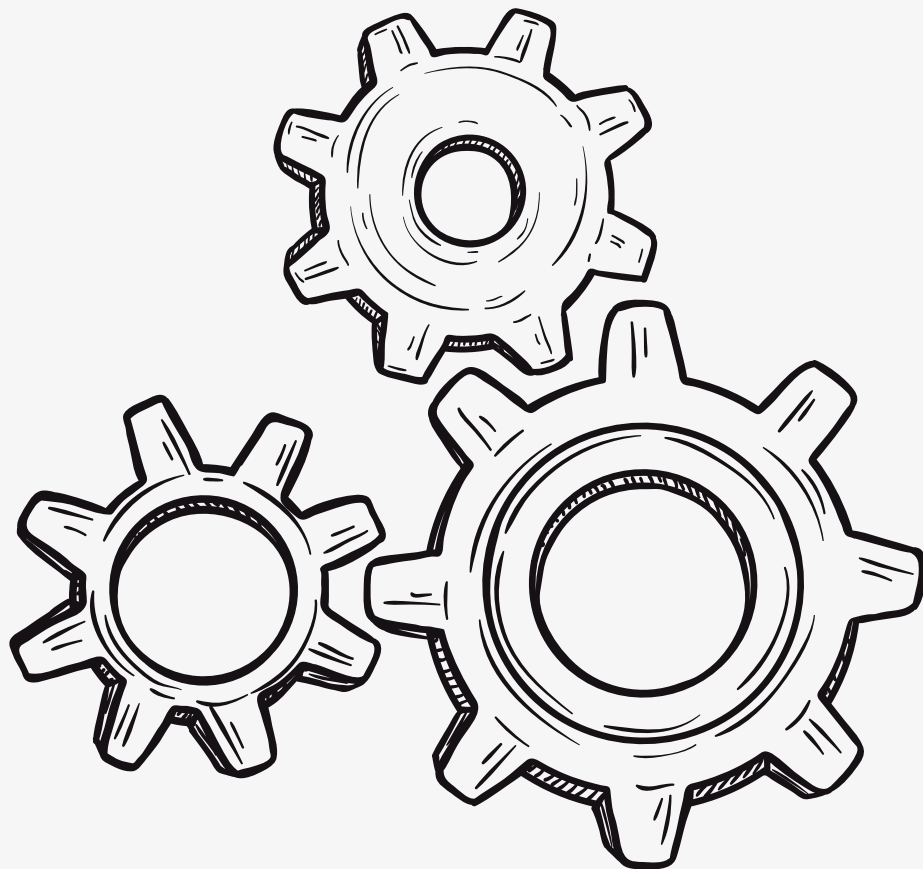
Lo scambio di dati tra entità è fondamentale per il corretto funzionamento di un sistema distribuito.

- Ogni entità può **generare** e **leggere** dati prodotti da altre entità.
- Per consentire questa comunicazione, i dati vengono serializzati in formato **JSON** e successivamente deserializzati per ricreare l'istanza dei dati.

```
case class RiverMonitorData(  
  riverMonitorState: RiverMonitorState,  
  threshold: Float,  
  monitoredSensors: Option[Map[String, Map[String, String]]]  
) derives ReadWriter
```

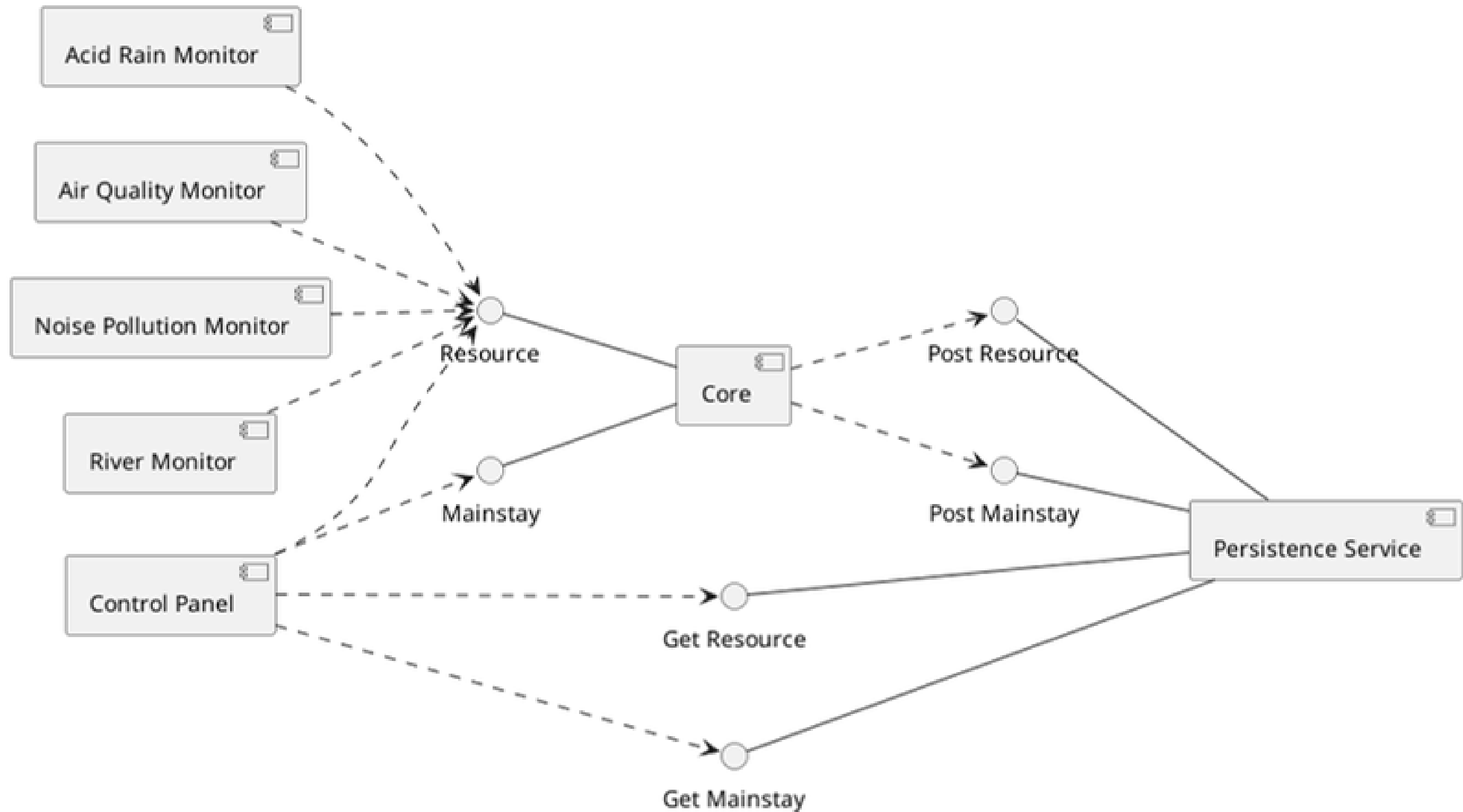
```
{"riverMonitorState": "Safe",  
  "threshold": 15,  
  "monitoredSensors": [{"FloodSensor_PonteVecchio": {  
    "Status": "online",  
    "WaterLevel": "18.988287"}}]}
```

Proprietà del sistema

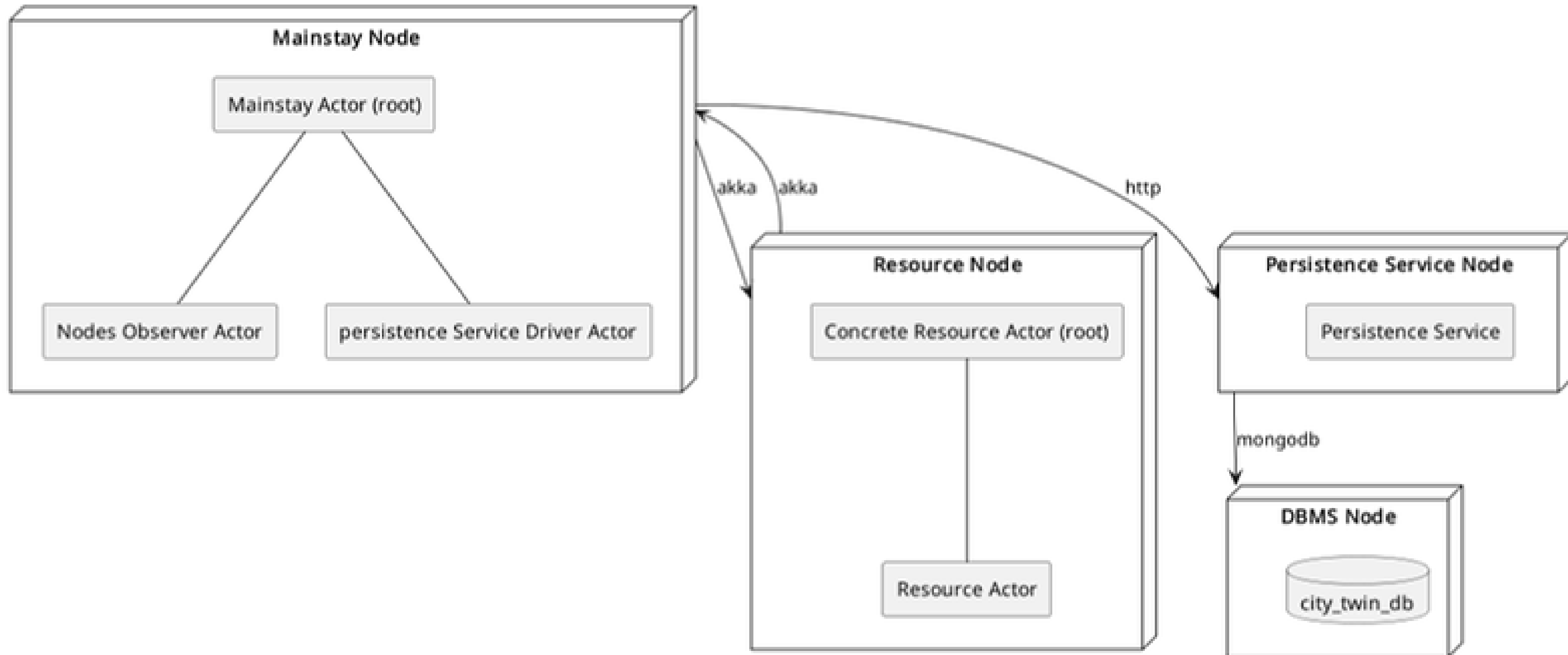


- **Consistenza:** *Nel caso in cui si avvia solo un nodo Mainstay, a discapito della disponibilità e dell'affidabilità. In caso di guasti il sistema fallisce.*
- **Disponibilità:** *L'esecuzione di più nodi Mainstay garantisce la resistenza ad eventuali malfunzionamenti.*
- **Affidabilità:** *le Informazioni sono ridondate in tutti i nodi Mainstay.*
- **Modularità:** Il sistema è stato realizzato in moduli separati, in modo da poter minimizzare le dipendenze.
- **Estendibilità:** l'architettura modulare permette di introdurre nuovi moduli, anche in fase di esecuzione.

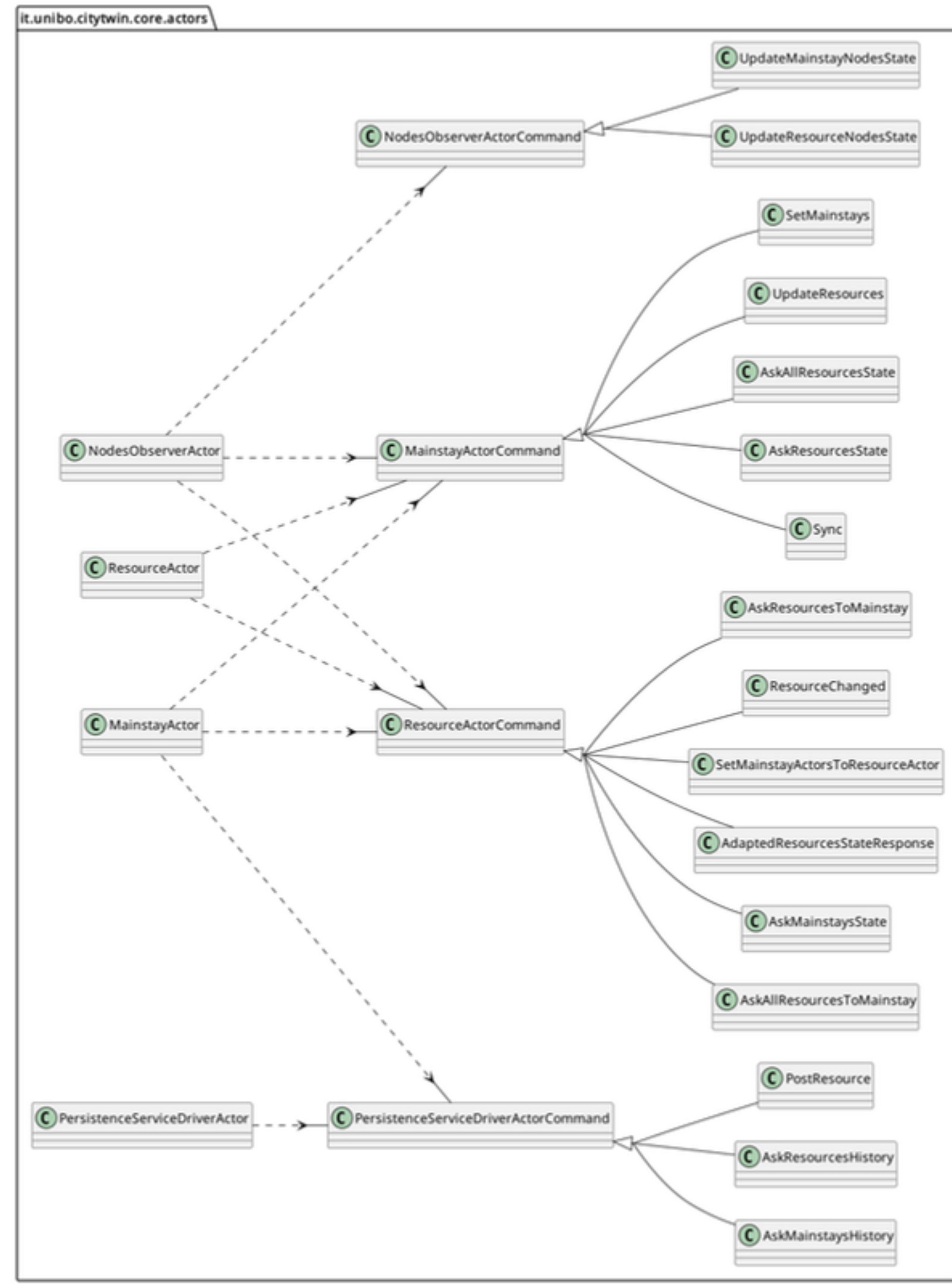
Design architetturale 1



Design architetturale 2



Design: Core



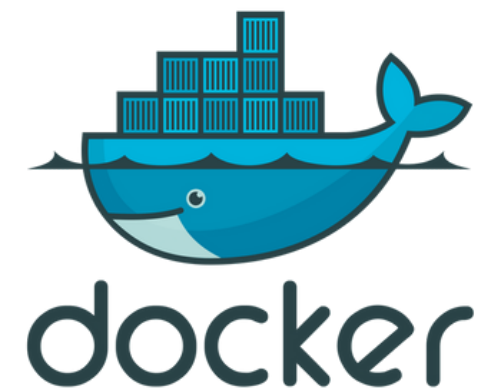
Tecnologie utilizzate



sbt



>ScalaTestTM
simply productive



Prestazioni

SPECIFICHE DEL SISTEMA

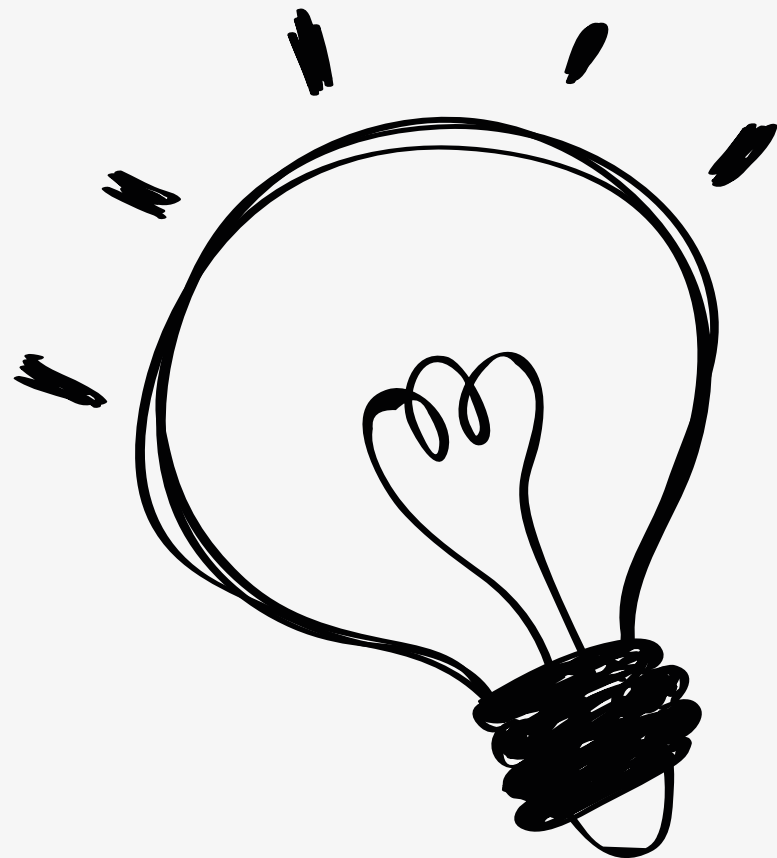
- ***Sistema operativo:*** NixOS 23.05
- ***Kernel Linux:*** 6.1.47
- ***CPU:*** AMD Ryzen 7 5700U
- ***Core:*** 8
- ***Thread:*** 16
- ***Clock base:*** 1.8 GHz
- ***Clock massimo:*** 4.3 GHz
- ***Cache L2:*** 4 MB
- ***Cache L3:*** 8 MB
- ***Memoria:*** 15325 MiB



RISULTATI DELL'ANALISI EFFETTUATA

- ***Numero di nodi Mainstay:*** 3
- ***Numero di nodi Resource:*** 31
- ***Numero di richieste effettuate:*** 101
- ***Numero di risposte ottenute:*** 101
- ***Tempo di risposta massimo:*** 3.228 s
- ***Tempo di risposta minimo:*** 0.002 s
- ***Tempo di risposta medio:*** 0.183 s

Sviluppi futuri



- *Sostituzione dei sensori virtualizzati con sensori reali.*
- *Implementazione di un sistema di autenticazione e autorizzazione.*
- *Integrazione di Google Maps nel pannello di controllo.*