DS - PRACTICE COURT

Sistema distribuito per la prenotazione di impianti sportivi

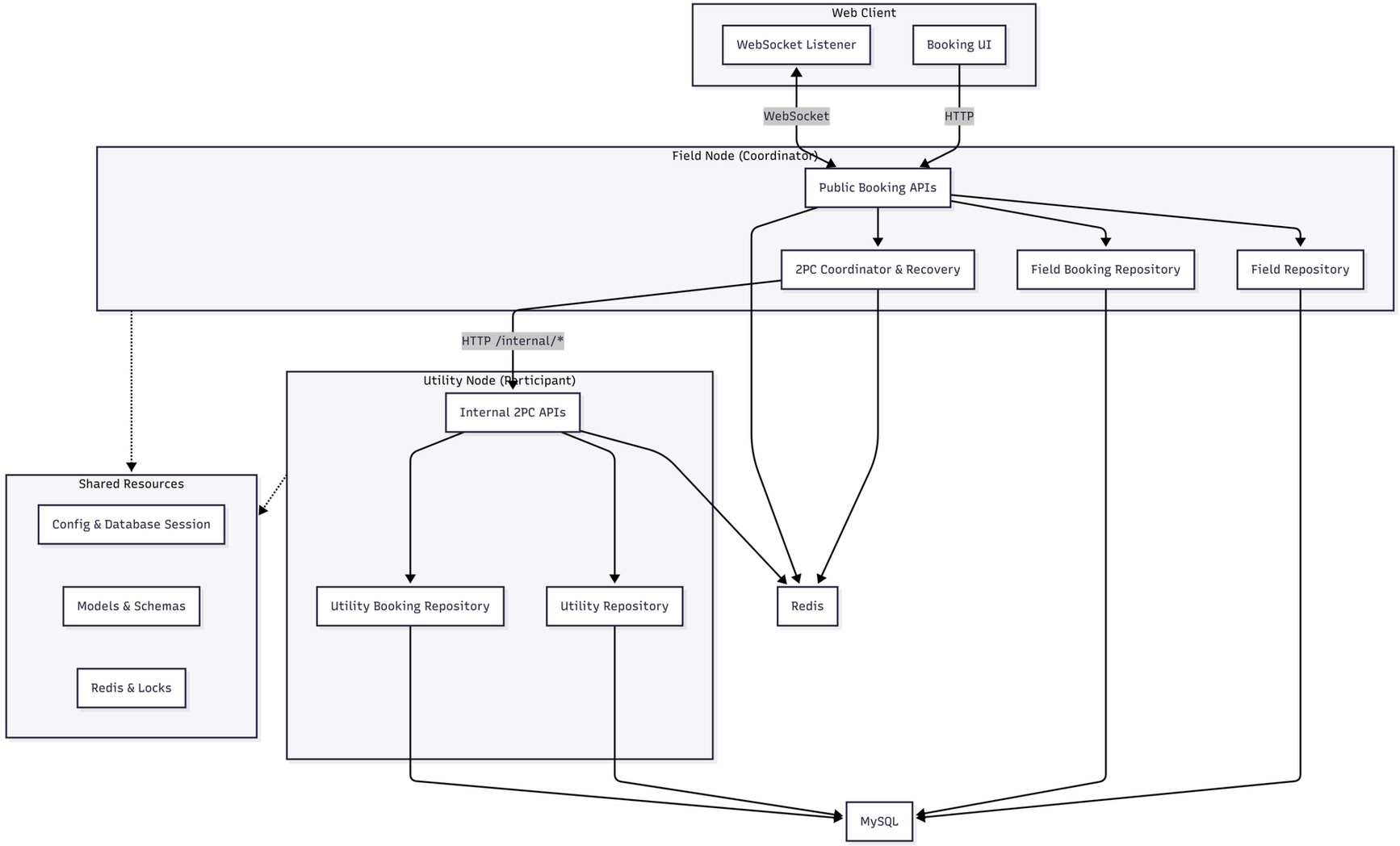
Stack: Python 3.12 - FastAPI - Redis - MySQL - Docker

Realizzato da Brighi Federico

Perchè è un sistema distribuito?

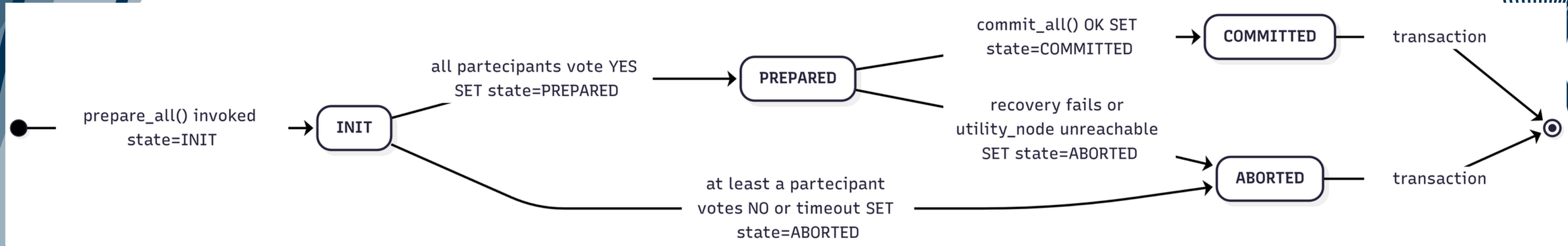
- **Separazione delle responsabilità:** 2 nodi Docker: Field Node e Utility Node rappresentano domini distinti (impianti vs servizi extra) gestiti spesso da provider diversi nella realtà
- **Concorrenza sulle risorse:** negli orari di punta più utenti competono per lo stesso slot: serve un meccanismo di lock distribuito per evitare il double booking
- **Fault Tolerance:** guasti di rete o nodi non raggiungibili sono inevitabili, il sistema deve non finire mai in uno stato non consistente

Architettura Generale:



Field Node (coordinatore, API pubbliche), Utility Node (partecipante 2PC, API private)
Redis (stato transiente, lock/hold/transizioni), MySQL (persistenza)

Protocollo Two-Phase Commit



Field Node coordinatore, Utility Node partecipante

Fase 1 (Prepare): richiesta di voto per ogni utility richiesta

Fase 2: se tutti i voti sono YES → commit, altrimenti → rollback

Questo meccanismo garantisce che un booking non sia mai preparato “a metà” (field confermato senza utility, o viceversa)

Fault Tolerance & CAP Theorem

Recovery loop in background (ogni 60sec) scansiona Redis per transazioni bloccate in stato PREPARED. Per ognuna tenta di completare il commit con l'Utility Node, altrimenti fa rollback.

<u>Limite Noto:</u>

TTL transazione su Redis 300s → se il Field Node crasha per un tempo maggiore, la transazione decade e il booking resta PENDING per sempre!

Il sistema è un CP System (Consistency + Partition Tolerance):
In caso di Network Partition tra Field e Utility Node, il booking distribuito viene rifiutato piuttosto che rischiare uno stato inconsistente

<u>Motivazione:</u>

Una prenotazione errata è peggio di un temporaneo rifiuto del servizio!

Concorrenza: Distributed Locking

- **Ogni time slot è protetto da un lock Redis con il comando SET NX PX:**
- **Ogni lock ha un token univoco (uuid4) che ne certifica il proprietario**
- **Il rilascio non è banale: un semplice GET+DELETE non è atomico, un altro client potrebbe intromettersi nella finestra tra le due operazioni**
- **<u>Soluzione adottata</u>: script Lua eseguito server-side su Redis, che controlla il token e cancella in un'unica operazione indivisibile.**

```
_LUA_RELEASE_SCRIPT = """
if redis.call('get', KEYS[1]) == ARGV[1] then
    return redis.call('del', KEYS[1])
else
    return 0
end
"""
```

Demo e User Interface

Griglia di disponibilità (available/booked/held) e live feed con possibilità di cancellazione

Book a Field

WHO ARE YOU?

federico brighi

SPORT

Field A — €10/h

DATE

29/05/2026

AVAILABLE SLOTS (Click first and last to select a range of hours)

08:00	08:30	09:00 H	09:30 H	10:00	10:30
11:00	11:30	12:00	12:30	13:00	13:30
14:00	14:30	15:00	15:30	16:00	16:30
17:00	17:30	18:00	18:30	19:00	19:30
20:00	20:30	21:00	21:30		

UTILITIES (Optional)

Changing Room €9 | Indoor Field €3 | Field Heating €10

SPORT

Field A — €10/h

DATE

29/05/2026

AVAILABLE SLOTS (Click first and last to select a range of hours)

08:00	08:30	09:00 H	09:30 H	10:00	10:30
11:00	11:30	12:00	12:30	13:00	13:30
14:00	14:30	15:00	15:30	16:00	16:30
17:00	17:30	18:00	18:30	19:00	19:30
20:00	20:30	21:00	21:30		

UTILITIES (Optional)

Changing Room €9 | Indoor Field €3 | Field Heating €10 | Nocturnal Lighting €7 | Tennis Racket Rental €8 | Training Bib Rental €8 | Ball Rental €2 | Lights 1777216887 €5 | Lights 1777217636 €5 | Lights 1777217831 €5 | Luci E2E 1777395111 €5 | Test Lighting 2PC €5 | Lights 1777395111 €5 | Luci E2E 1779280255 €5 | Test Lighting 2PC €5 | Lights 1779280255 €5

€36.50

Confirm Booking

POST /bookings/2pc prepare -> commit

Live Feed

LIVE

ws://localhost:8001/ws/availability

Booking Confirmed #51
Field 1 · federico
29 mag 11:00 - 12:30

Booking Confirmed #50
Field 1 · fede
29 mag 10:00 - 11:00

Booking Confirmed #49
Field 1 · fede
29 mag 08:00 - 09:00

Booking Failed #48
Field 35 · test_ws_fail177928...
01 gen 11:00 - 12:00

Booking Confirmed #47
Field 35 · test_ws_multi_177...
01 gen 12:00 - 13:00

Testing e Validazione del sistema

Il sistema è stato opportunamente validato tramite diverse tipologie di test:

- **Unit Test: logica interna del coordinatore 2PC (Field Node)**
- **Integration Test: comunicazione tra Field Node e Utility Node**
- **End-To-End Test: flussi di booking completi**
- **Concurrency Test: prevenzione dei double booking**
- **Fault Tolerance Test: recovery da stati intermedi**
- **WebSocket Test: broadcast corretto degli eventi**
- **Manual Test: prove manuali delle varie funzioni della UI**

Self Evaluation & Miglioramenti Futuri

- **<u>Punti di Forza:</u> separazione delle responsabilità, gestione della concorrenza e aggiornamenti real-time**

- **<u>Limiti:</u> fault tolerance non completa in tutti gli scenari possibili (esempio: crash prolungato del coordinatore)**

- **<u>Sviluppi Futuri:</u> aggiunta dell'autenticazione utenti e una recovery più robusta**

GRAZIE PER L'ATTENZIONE

Realizzato da Brighi Federico