# Corso di Sistemi Distribuiti 2022-2023
# Progetto finale di laboratorio

Flavio Maria De Paoli [*] Michele Ciavotta [†]
Claudia Raibulet [‡] Emanuele Petriglia [§]

Aggiornato il 09 Giugno 2023

## Indice



## Calendario

| | |
|---|---|
| 2023-05-31 | Pubblicazione progetto. |
| 2023-06-07 | Incontro discussione dubbi sul progetto alle ore 10:30 in aula T024 (edificio U14). |
| 2023-06-11 | Termine registrazione dei gruppi. |
| 2023-06-30 | Termine consegna dei progetti. |

## 1 Introduzione

Il progetto d'esame del corso di "Sistemi Distribuiti" dell'anno 2022-2023 consiste nella progettazione e sviluppo di un'applicazione distribuita per la prenotazione di posti di proiezioni cinematografiche in un cinema.

[*]email `flavio.depaoli@unimib.it`
[†]email `michele.ciavotta@unimib.it`
[‡]email `claudia.raibulet@unimib.it`
[§]email `emanuele.petriglia@unimib.it`

**Architettura** L'applicazione deve essere composta da un'architettura client-server con due componenti server e un client, come mostrato nella figura 1 a pagina 2:

1. **Client Web**: un'interfaccia per la prenotazione e gestione di posti in un cinema. Comunica con il server Web tramite delle API REST.
2. **Server Web**: implementa la logica di gestione delle prenotazioni e delle proiezioni del cinema. Comunica con il client Web tramite delle API REST che espone, mentre utilizza un protocollo personalizzato su socket TCP per comunicare con il database.
3. **Database**: un database chiave-valore in-memory che gestisce i dati delle prenotazioni, sale e proiezioni del cinema. Comunica con il server Web tramite protocollo personalizzato su socket TCP.

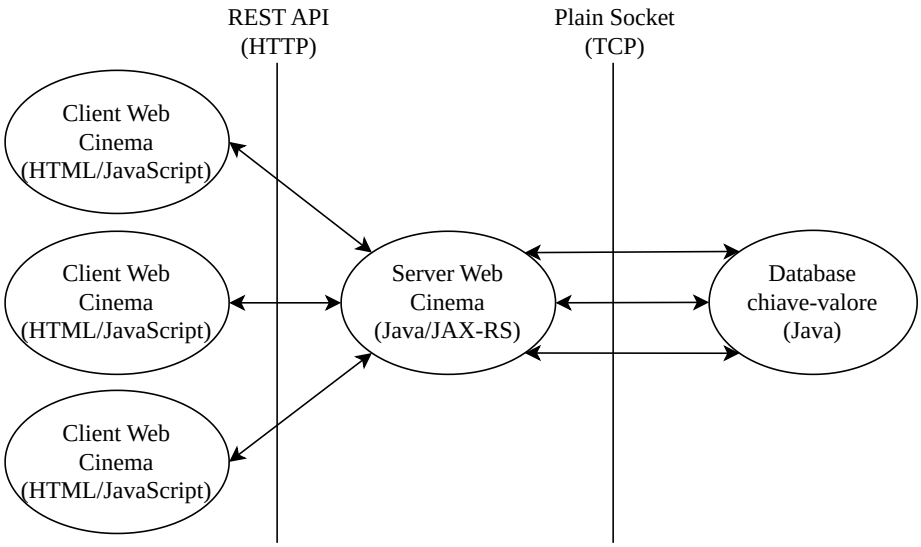


Figura 1: Schema architetturale del progetto.

Il sistema deve prevedere l'esistenza di zero o più istanze in esecuzione del client Web, mentre una sola istanza per il server Web e il database.

**API REST e socket TCP** La progettazione delle API REST HTTP deve modellare il il tema (prenotazioni, posti, proiezioni), mentre la progettazione del protocollo su socket TCP deve modellare l'interazione tra il database e il server Web (comandi per salvare, eliminare o recuperare delle chiavi e i valori associati). Le API REST e il protocollo su socket TCP devono essere documentate.

**Implementazione** È richiesto l'uso di **JavaScript** e **HTML** per realizzare il client Web, con le tecnologie e standard mostrati nei laboratori (API fetch, selettori DOM...). Mentre per il server Web e il database è richiesto **Java**, almeno la versione 11, ed è richiesto l'uso delle librerie e standard mostrati nei laboratori (API Jackarta come JAX-RS, Jersey, Jackson e Grizzly). Ulteriori librerie esterne sono permesse ma non devono essere invasive (es. una libreria per generare degli ID va bene, una libreria che gestisce tutta la logica o un framework non va bene). Per il frontend si possono usare librerie per lo stile grafico (CSS), anche se non verrà valutato, mentre non si possono usare librerie esterne per JavaScript. Verrà fornito uno scheletro di partenza del progetto.

**Gruppi e consegna** Il progetto è tarato per gruppi di tre studenti. Sono sconsigliati, ma ammessi, gruppi di due studenti. Ogni gruppo si deve registrare, indicando matricola, nome, cognome ed email di ogni componente, entro il 12 Giugno 2023 tramite il modulo apposito su e-Learning. Il termine per la consegna è il 30 Giugno 2023. Si può consegnare il progetto **solo una volta**.

**Dubbi e domande** Eventuali dubbi e domande possono essere poste nell'incontro del 7 Giugno 2023. Dopo tale data si possono porre domande pubbliche nel forum di laboratorio su e-Learning.

## 2 Componenti del sistema

**Client Web** Il sistema gestisce le proiezioni cinematografiche di un cinema multi sala, per cui il client Web deve permettere agli utenti le seguenti azioni:

- Visualizzare un elenco delle prossime proiezioni, con i dati del film, data, orario e sala (in un giorno ci possono essere più proiezioni con orari ovviamente non sovrapposti per le stesse sale).
- Effettuare prenotazioni di uno o più posti per una proiezione, potendo selezionare esattamente quali posti tra i disponibili, tramite una rappresentazione della sala (si suggerisce di usare una tabella).
- Gestire la prenotazione effettuata rimuovendo dei posti o rimuovendo l'intera prenotazione.

Non è richiesto alcun tipo di autenticazione degli utenti. Per permettere quindi la modifica di una prenotazione, si suggerisce di fornire all'utente un codice identificativo univoco per una prenotazione effettuata, codice che poi l'utente può usare per selezionarla e modificarla.

Il client Web non deve essere fornito come risorsa statica dal server Web, bensì come file HTML in una cartella a parte, come mostrato nello scheletro.

**Server Web** Il server Web deve implementare tutta la logica di gestione delle prenotazioni e proiezioni. Ogni richiesta del client deve essere gestita aprendo una connessione socket verso il database per la persistenza dei dati. La concorrenza è gestita in modo implicito da Jersey.

Lo scheletro fornito implementa la parte necessaria per l'avvio, molto simile a quella degli esercizi di laboratorio.

**Database** Il database deve essere implementato come chiave-valore e in-memory.

Un database chiave-valore è un paradigma che consiste nel gestire, salvare e recuperare i dati attraverso una tabella hash (o anche chiamato dizionario). In generale le chiavi e i valori possono essere tipi di dati arbitrari o binari, tuttavia si suggerisce di limitare i tipi alle sole stringhe. Si può prendere ispirazione dai tipi di dati `strings` e `lists` del database Redis. Si suggerisce di sfruttare l'uso delle chiavi, in modo da avere tante chiavi con valori contenuti (stringhe brevi) rispetto a poche chiavi con valori corposi (es. serializzazione di un intero oggetti con sotto-liste).

Un database in-memory è un database che gestisce i dati principalmente nella memoria centrale (RAM). Per il progetto non è richiesto di salvare i dati sulla memoria secondaria.

È richiesto che ci siano dei dati sulle proiezioni cinematografiche precaricati, per poter effettuare le prove e poter valutare il funzionamento del sistema. Per fare ciò si suggerisce di leggere un file dalla memoria secondaria all'avvio del database, così da caricare i dati letti (coppie di chiavi con valore) in memoria principale. Si può decidere liberamente come strutturare il file (ad esempio un file testuale serializzato in JSON, serializzato in binario o un CSV).

A differenza del server Web, per il database è necessario gestire la concorrenza in modo esplicito, per poter accettare più socket dal server Web e gestire correttamente la concorrenza delle operazioni sulle chiavi.

# 3 Comunicazione tra i componenti

**Client Web ↔ Server Web (API REST)** L'API REST, come già descritto, deve modellare la gestione delle prenotazioni dei posti, le sale e le proiezioni. L'API deve essere documentata in modo simile a come mostrato nello scheletro fornito.

Poiché verrà valutata sia la documentazione che la progettazione delle API, si suggerisce di rivedere il materiale legato al laboratorio 6 e argomento 5 di teoria su REST.

**Server Web ↔ Database (socket TCP)** Il protocollo di comunicazione tra database e server Web deve essere costruito considerando le richieste del sistema. Si suggerisce di implementare un insieme limitato di comandi e di trasmettere dati in forma testuale. Si può prendere ispirazione al protocollo di Redis, che è un protocollo testuale. Si sconsiglia di realizzare un protocollo binario, perché, anche se più efficiente, è più difficile da gestire e da implementare.

Il protocollo progettato deve essere generico, come anche i comandi, il formato delle chiavi e dei valori. In linea teorica si deve poter riutilizzare il componente anche in progetti di natura completamente diversa.

Si suggerisce inoltre di fissare una porta conosciuta (come 80/8080 per HTTP) sia dal database che dal server Web.

Il protocollo deve essere documentato in modo simile a come mostrato nello scheletro fornito. In caso di protocollo testuale, si deve documentare come è fatta una richiesta e la risposta, quali sono i comandi del database e come si possono formare le chiavi e cosa si può scrivere come valore.

Poiché verrà valutata sia la documentazione che la progettazione del protocollo, si suggerisce di rivedere il materiale legato al laboratorio 1 e 2 e argomento 2 su socket.

# 4 Consegna e valutazione

**Modalità di consegna** Ogni gruppo deve consegnare il progetto tramite l'apposito modulo su e-Learning (lo stesso della registrazione dei gruppi), e la consegna consiste in un file testuale che deve contenere:

1. Nome, cognome, matricola ed email di ogni componente del gruppo.
2. Link alla repository su GitLab o GitHub.

I repository devono essere privati. Una volta consegnato sarà necessario fornire l'accesso, verranno date maggiori informazioni su come fare.

**Struttura repository** La repository deve seguire, nella cartella principale di root, la seguente struttura:

- `database`: un cartella che contiene il codice relativo al database,
- `server-web`: un cartella che contiene il codice relativo al server web,
- `client-web`: un cartella che contiene il codice relativo al client web.
- `README.md`: un file testuale con scritto il nome, cognome, matricola ed email di ogni componente del gruppo. Deve contenere una descrizione del lavoro svolto, nonché le istruzioni per la compilazione ed esecuzione.
- `REST.md`: un file testuale contente la descrizione dell'API REST progettata.
- `TCP.md`: un file testuale con la descrizione del protocollo progettato e implementato su socket TCP.

In aggiunta devono essere presenti dei screenshot del client Web. I tre file testuali devono essere scritti in formato Markdown. Lo scheletro fornito è già strutturato nell'elenco mostrato e contiene delle indicazioni per ogni cartella e file, oltre alle istruzioni di compilazione ed esecuzione.

**Esclusione del progetto** Il progetto verrà escluso dalla valutazione in caso di plagio o difformità dalla struttura di consegna.

**Valutazione** In base al numero dei gruppi partecipanti, dopo il termine di registrazione dei gruppi, sarà indicato se la restituzione della valutazione avverrà con una discussione dei progetti in presenza o tramite e-Learning.

È possibile espandere le funzionalità del progetto, tuttavia il progetto verrà valutato in base ai requisiti richiesti, ed eventuali commit sulla repository realizzati dopo la data di consegna verranno ignorati.

La valutazione consiste in quattro punti, distribuiti uno per ogni componente (client Web, server Web e database) e uno per la documentazione. Per i tre componenti, è necessario che si possano compilare ed eseguire, che le funzionalità siano state implementate e che non ci siano problemi di concorrenza. Per la documentazione, deve essere chiara e completa, e soprattutto rispecchiare l'effettivo comportamento del sistema implementato. Solo errori gravi verranno penalizzati.