Corso di Sistemi Distribuiti 2024-2025
Progetto finale di laboratorio

Flavio Maria De Paoli * Michele Ciavotta †
Emanuele Petriglia ‡

Aggiornato il 23 Maggio 2025

Indice

Calendario

2025-05-23	Pubblicazione progetto.
2025-05-23	Inizio registrazione dei gruppi.
2025-06-14	Termine registrazione dei gruppi.
2025-06-30	Termine consegna dei progetti.

1 Introduzione

Il progetto consiste nella progettazione e sviluppo di un'applicazione distribuita per la creazione e gestione della "Carta Cultura Giovani".

Architettura. L'applicazione da realizzare deve essere composta da tre componenti come mostrato in fig. 1: un client, un server e un database. Ogni componente è così specificato:

*flavio.depaoli@unimib.it
†michele.ciavotta@unimib.it
‡emanuele.petriglia@unimib.it

1. **Client Web**: un'interfaccia che permetta agli utenti l'interazione con l'applicazione. Deve permettere la gestione del contributo di 500 euro della "Carta Coltura Giovani", tramite la generazione dei buoni. Il client con il server Web tramite delle API REST.

2. **Server Web**: contiene la logica di gestione del contributo e dei buoni per utente. 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 degli utenti, del contributo e dei buoni. Comunica con il server Web tramite protocollo personalizzato su socket TCP.

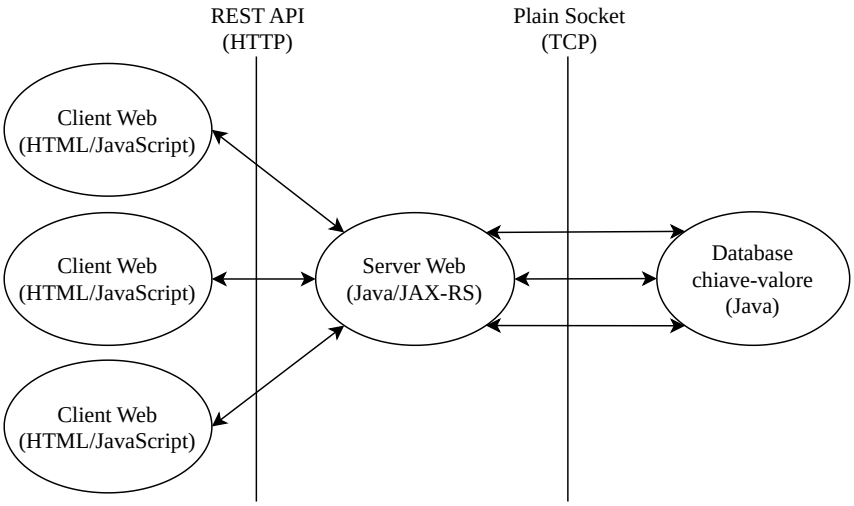

Figura 1: Schema architetturale del progetto.

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 per il database.

API REST e socket TCP. La progettazione delle API REST HTTP deve modellare il tema (utente, contributo, buoni), 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 le coppie chiave-valore). Le API REST e il protocollo su socket TCP devono essere documentati.

Implementazione. Il client Web deve essere implementato utilizzando **JavaScript** e **HTML**. Eventuale codice CSS per lo stile grafico è ammesso, ma non verrà valutato. È vietato l'uso di librerie o framework JavaScript.

Il server Web deve essere implementato in **Java** usando le specifiche Jakarta RESTful Web Services (JAX-RS) e Jakarta JSON Binding (JSON-B). Anche il database deve essere implementato in **Java**. Per entrambi, eventuali librerie

all'infuori di quelle indicate sono ammesse ma non devono essere invasive (per esempio una libreria per generare degli ID va bene, una libreria che gestisce tutta la logica o un framework no).

È obbligatorio partire dallo scheletro `skeleton.zip` fornito su e-Learning.

Ambiente di sviluppo. Utilizzare la VM usata nei laboratori per svolgere il progetto, perché i progetti verranno valutati utilizzando tale VM e lo scheletro fornito è garantito che funzioni.

Gruppi e registrazione. Il progetto deve essere svolto in gruppi da **tre** studenti. Gruppi composti da più o meno di tre persone non sono ammessi.

Ogni gruppo si deve **registrare entro il 14 Giugno** su e-Learning. Nella registrazione è necessario indicare il **nome del progetto, matricola, nome, cognome ed email di ogni componente**.

Consegna. Il termine della consegna è il **30 Giugno** tramite e-Learning. Maggiori informazioni nel section 4.

Dubbi e domande. Eventuali dubbi e domande possono essere poste nel forum e-Learning dedicato al laboratorio ("Forum Laboratorio").

2 Componenti del sistema

2.1 Client Web

Il client Web deve permettere agli utenti di svolgere le seguenti azioni:

- Controllare lo stato del contributo assegnato (inizia con 500€) e l'elenco dei buoni. In particolare, l'utente deve poter vedere lo stato riassuntivo indicando:
  - La quantità di contributo disponibile per generare dei buoni,
  - La quantità di contributo usata per generare dei buoni ma ancora non sono stati consumati,
  - La quantità di contributo usata per generare dei buoni che sono stati consumati.
- Controllare l'elenco cronologico dei buoni generati, con l'importo, il loro stato e il tipo di bene assegnato.
- Generare un nuovo buono inserendo le seguenti informazioni:
  - L'importo in euro (non può superare la quantità di contributo disponibile),
  - Il bene acquistato (una selezione tra cinema, musica, concerti, eventi culturali, libri, musei, strumenti musicali, teatro e danza).
- Visualizzare in dettaglio un buono:
  - L'importo in euro,

  - La tipologia di bene acquistato,
  - Lo stato (se è stato consumato oppure no),
  - Data di creazione e consumo (se è stato consumato). L'utente può inoltre cancellare un buono solo se non è stato consumato e può consumare il buono tramite un pulsante[1].

- Modificare un buono generato ma non ancora speso, permettendo il cambio di tipologia di bene acquistato.
- Registrare un nuovo utente, inserendo nome, cognome, email e codice fiscale.
- Visualizzare lo stato globale del sistema[2]:
  - Numero di utenti registrati,
  - Somma dei contributi di tutti gli utenti, differenziando la porzione disponibile, la porzione assegnata ai buoni ma non ancora spesa e la porzione spesa.
  - Numero dei buoni generati da tutti gli utenti, differenziando i buoni generati ma non consumati e i buoni consumati.

Gli utenti non sono autenticati e pertanto non è richiesta la progettazione e l'implementazione di un sistema di autenticazione. È però necessario tenere traccia dell'utente durante le fasi di acquisto e gestione[3].

Nota bene: se uno stesso utente genera in contemporanea più di un buono, il sistema blocca l'operazione se la somma degli importi supera quella disponibile del contributo (non può andare in negativo!).

2.1.1 Esempi di interazione

Di seguito riportiamo alcuni esempi di interazione tra un utente e l'applicazione tramite il client Web. Non sono molto diversi dalle prove che il docente esegue durante la valutazione:

1. L'utente ha disponibile 500€ di contributo e genera un buono da 140€ con la tipologia "cinema". Il sistema valida la richiesta e genera il buono.
2. L'utente ha disponibile 100€ di contributo e genera un buono da 110€ con la tipologia "musei". Il sistema valida la richiesta e rifiuta la generazione del buono.
3. L'utente richiede l'eliminazione di un buono non ancora speso. Il sistema accoglie la richiesta e cancella il buono. A seguire l'utente richiede l'eliminazione di un buono speso e questa volta il sistema rifiuta la richiesta.
4. L'utente ha disponibile 100€ di contributo e prova a generare in contemporanea due buoni da 60€. Il sistema rifiuta la generazione di uno dei due buoni.

[1] Questo per "simulare" il consumo effettivo del buono, rispetto alla realtà in cui è il venditore a validare il buono.

[2] È una sorta di "visualizzazione admin".

[3] Si suggerisce di usare il codice fiscale.

5. L'utente segna come speso un buono da 100€. Una volta speso prova a modificare il buono ma il sistema rifiuta l'operazione.

2.2 Server Web

Il server Web è responsabile della logica di gestione del contributo e dei buoni per ogni utente. È responsabile dell'accesso condiviso sulla stessa risorsa (il contributo) da parte dello stesso utente che genera più buoni. Implementa l'API REST esposta ai client.

Il server Web non gestisce la persistenza dei dati: ogni richiesta ricevuta dal client comporta l'apertura di una connessione socket verso il database per ottenere o salvare i dati.

2.3 Database

Deve essere implementata una versione minimale di un database chiave-valore 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 oggetto 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. L'unico requisito è che il database contenga alcuni dati relativi ai buoni e utenti preesistenti, affinché il docente possa effettuare le prove e valutare il funzionamento del sistema[4]. Nel database bisogna gestire la concorrenza in modo esplicito nelle operazioni sulle stesse chiavi. Il database deve essere predisposto ad accettare più connessioni socket dal server Web.

3 Comunicazione tra i componenti

Client Web ↔ Server Web (API REST) L'API REST, come già descritto, deve modellare la gestione dei buoni e del contributo per ogni utente. L'API deve essere documentata in modo simile a come mostrato nello scheletro fornito.

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

Nota bene: la progettazione dell'API deve seguire lo stile architetturale REST e considerare JSON come formato di scambio dei dati.

[4]Per fare ciò, si suggerisce di leggere un file dalla memoria secondaria all'avvio del database. L'eventuale inserimento di coppie chiave-valore hard-coded nel codice del database viene penalizzato!

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 ai laboratori 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. **Solo un membro del gruppo** deve consegnare il progetto. La consegna consiste in archivio compresso ZIP che deve il codice del progetto e deve seguire lo scheletro fornito.

Struttura progetto La cartella principale del progetto deve seguire 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 del progetto, 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 degli **screenshot del client Web**. I tre file testuali devono essere scritti in formato Markdown. Lo scheletro fornito è già strutturato come richiesto e contiene indicazioni di partenza per ogni file e cartella.

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

Valutazione La restituzione della valutazione avviene tramite un breve colloquio nelle prime settimane di Luglio per ogni gruppo da svolgere in presenza, dalla durata di circa 15-20 minuti. Durante il colloquio il docente può fare domande sul progetto, sulle scelte di progettazione e di implementazione. La presenza è obbligatoria per tutti i membri del gruppo.

È possibile espandere le funzionalità del progetto, ma ciò non influenza la valutazione.

La valutazione consiste in quattro punti, distribuiti uno per ogni componente (client Web, server Web e database) e uno per la documentazione. Possono essere assegnati valori intermedi per punto. Viene valutata la correttezza e implementazione dei tre componenti compilandoli ed eseguendoli. Sono controllate se le funzionalità richieste sono presenti e se ci sono problemi di concorrenza. Per la documentazione, viene valutata la chiarezza e completezza, e soprattutto se rispecchia l'effettivo comportamento del sistema implementato.

La valutazione avviene utilizzando la VM fornita durante il laboratorio. È importante quindi che il gruppo verifichi che il progetto funzioni all'interno di essa.