Corso di Sistemi Distribuiti 2023-2024
Progetto finale di laboratorio

Flavio Maria De Paoli [*] Michele Ciavotta [†]
Federica Filippini [‡] Emanuele Petriglia [§]

Aggiornato il 31 Maggio 2024

Indice

1 Introduzione **2**

2 Componenti del sistema **3**
2.1 Client Web 3
2.1.1 Esempi di interazione 4
2.2 Server Web 5
2.3 Database 5

3 Comunicazione tra i componenti **6**

4 Consegna e valutazione **7**

Calendario

2024-05-31	Pubblicazione progetto.
2024-06-06	Incontro discussione dubbi sul progetto in Aula C1, Edificio U24. Orario 11:00 - 12:00.
2024-06-14	Termine registrazione dei gruppi.
2024-06-30	Termine consegna dei progetti.

[*] flavio.depaoli@unimib.it
[†] michele.ciavotta@unimib.it
[‡] federica.filippini@unimib.it
[§] emanuele.petriglia@unimib.it

1 Introduzione

Il progetto d'esame del corso di "Sistemi Distribuiti" dell'anno 2023-2024 consiste nella progettazione e sviluppo di un'applicazione distribuita per l'acquisto e gestione di domini Internet.

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

1. **Client Web**: un'interfaccia che permetta agli utenti l'interazione con l'applicazione. Deve permettere la gestione di domini Internet, tra cui l'acquisto di un nuovo dominio. Comunica con il server Web tramite delle API REST.
2. **Server Web**: contiene la logica di gestione dei domini e del loro acquisto. 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 documentale che gestisce i dati degli acquisti e dei domini acquistati da parte degli utenti. Comunica con il server Web tramite protocollo personalizzato su socket TCP.

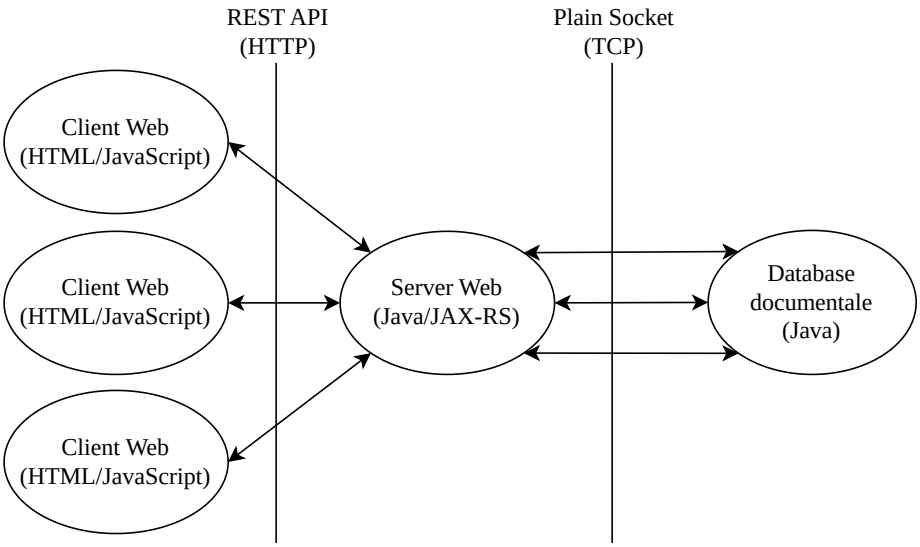

Figura 1: Schema architetturale del progetto.

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 (domini, acquisti, utenti), 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 i documenti). Le API REST e il protocollo su socket TCP devono essere documentate.

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 e Jakarta JSON Binding. Anche il database deve essere implementato in **Java**. Per entrambi, eventuali librerie all'infuori di quelle permesse 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-project.zip` fornito su e-Learning.

Ambiente di sviluppo. Si suggerisce di utilizzare la macchina virtuale usata nei laboratori per svolgere il progetto. **Attenzione!** La macchina virtuale non persiste i dati per più di qualche giorno, assicurarsi ogni volta di salvare i dati. I progetti verranno valutati utilizzando la macchina virtuale. Lo scheletro fornito è garantito che funzioni utilizzando la macchina virtuale.

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

Ogni gruppo si deve **registrare entro il 14 Giugno** su e-Learning. La registrazione deve essere **effettuata da un solo membro** del gruppo e deve indicare il **nome del progetto, matricola, nome, cognome ed email di ogni componente**.

Consegna. Il termine della consegna è il **30 Giugno**.

Dubbi e domande. Eventuali dubbi e domande possono essere poste nell'incontro del **6 Giugno**. Dopo tale data si può usare il forum di laboratorio su e-Learning per altre domande.

2 Componenti del sistema

2.1 Client Web

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

- Controllare se un dominio Internet è disponibile per la registrazione. Se non è disponibile, indicare nome, cognome ed email dell'utente che ha registrato il dominio, e la sua data di scadenza.

- Registrare un dominio Internet, dopo aver controllato se è disponibile. La registrazione avviene tramite un acquisto. I dati di registrazione comprendono: dominio da acquistare, durata della registrazione (minimo un anno, massimo dieci). I dati dell'acquisto comprendono: nome, cognome e indirizzo email dell'utente[1], numero di carta di credito, data di scadenza della carta di credito, CVV, nome e cognome dell'intestatario della carta.

- Visualizzare l'elenco dei domini registrati con i seguenti dati: dominio, data di registrazione e data di scadenza. Nell'elenco ci devono essere anche i domini scaduti e non rinnovati.

- Visualizzare l'elenco degli ordini. Ogni ordine contiene i seguenti dati: dominio a cui fa riferimento, data ordine, oggetto dell'ordine (registrazione o rinnovo) e quota pagata.

- Rinnovare un dominio. Il rinnovo è come la registrazione di un nuovo dominio, ma fa riferimento a un dominio che già è registrato all'utente. Si può estendere il rinnovo fino a dieci anni totali (ad esempio, se un dominio scade tra tre anni, l'utente può rinnovare per massimo altri sette anni).

- Registrare un nuovo utente, inserendo nome, cognome ed email.

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[2].

Nota bene: se, mentre un utente sta acquistando un dominio, un secondo utente ne tenta l'acquisto, il sito blocca l'operazione dicendo che l'acquisto è già in corso (vedi esempio 3 nella sezione successiva).

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 i docenti eseguono durante la valutazione.

Negli esempi, si presuppone che l'utente sia identificato da un ID numerico univoco e che il sistema permetta di registrare domini *.it*. Si presuppone inoltre che esista un utente con ID 12 che abbia registrato all'inizio i seguenti domini:

- *mariorossi.it* per un anno dal 2024-03-03 al 2025-03-02.

- *costruzionisrl.it* per due anni dal 2024-05-05 al 2026-05-04.

[1] Questi tre dati possono essere recuperati in automatico dal profilo dell'utente.

[2] Si suggerisce di far inserire all'utente un codice identificativo univoco (ad esempio l'email, se è univoca, o un ID generato dal server), così da distinguere le richieste che il client fa al server.

Gli esempi sono:

1. L'utente 6 intende registrare il dominio *costruzionisrl.it* per un anno. Inserisce il dominio nel sito ma il sito risponde dicendo che è stato già registrato e mostra nome, cognome ed email dell'utente 12 con la data di scadenza del 2026-05-04.

2. L'utente 6 intende registrare il dominio *avvocati.it* per un anno. Inserisce il dominio nel sito e il sito risponde che è disponibile. Procede alla registrazione. Inserisce un anno di scadenza e poi i dati della carta. Al termine l'ordine viene completato e il dominio viene registrato e associato all'utente 6.

3. L'utente 6 intende registrare il dominio *barchecco.it* per un anno. Inserisce il dominio nel sito e il sito risponde che è disponibile. Procede alla registrazione. Inserisce un anno di scadenza e poi i dati della carta. **Nel mentre**, l'utente 8 cerca di registrare lo stesso dominio; il sito blocca questa operazione anche se il dominio risulta disponibile, dicendo che ne è in corso l'acquisto da parte di un altro utente. L'ordine dell'utente 6 viene completato e il dominio viene registrato.

4. L'utente 12 visualizza tutti i domini che ha registrato con la relativa scadenza. Rinnova il dominio *mariorossi.it* per un altro anno.

5. Un nuovo utente si registra inserendo nome, cognome ed email. Al termine, il sistema restituisce il suo codice identificativo 21. L'utente apre la pagina con l'elenco dei domini usando il suo codice, ma questa risulta vuota perché non ha ancora registrato alcun dominio.

2.2 Server Web

Il server Web è responsabile della logica di gestione dei domini, il loro acquisto, rinnovo e controllo della disponibilità. Deve anche gestire gli acquisti e gli ordini. È responsabile dell'accesso condiviso sulle stesse risorse da parte di più utenti (es. acquisto di uno stesso dominio).

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 orientato ai documenti. I documenti possono essere salvati a scelta in memoria principale (RAM, in-memory) oppure in memoria secondaria (un file su disco liberamente strutturato). L'unico requisito è che il database contenga alcuni dati relativi a domini, utenti e ordini preesistenti, affinché i docenti possano effettuare le

prove e valutare il funzionamento del sistema[3]. Nel database bisogna gestire la concorrenza in modo esplicito nelle operazioni sugli stessi documenti. Il database deve essere predisposto ad accettare più connessioni socket dal server Web.

Cos'è un database documentale? Un database orientato ai documenti è un sistema che permette di archiviare, gestire e recuperare dati sotto forma di documenti. Un documento è una informazione completa a sé stante (per esempio, una ricevuta o un utente con i suoi dati). Un documento è codificato in un formato conosciuto dal database (per esempio XML o JSON). Uno o più documenti possono essere racchiusi all'interno di una stessa collezione (per esempio una collezione di ricevute che comprende un documento per ogni ricevuta). L'identificazione dei documenti avviene tramite una chiave, che può essere un ID o un percorso (per esempio "/ricevute/2024-05-15"). Il database espone dei comandi per manipolare il documento e per interrogare un singolo documento o tutti i documenti di una collezione.

A differenza di un database relazionale, i documenti non devono seguire uno schema predefinito, ma devono usare la stessa codifica (JSON, XML...). A differenza di un database chiave-valore, il database documentale espone sempre dei comandi per manipolare o interrogare il contenuto di un documento.

Cosa si deve implementare? È richiesta una implementazione minimale di un database documentale. I client devono poter creare ed eliminare singoli documenti, organizzati in più collezioni. Ci deve essere **almeno un comando** che permetta al client di interrogare il contenuto dei documenti di una collezione[4]. La scelta se salvare i documenti in memoria principale o in memoria secondaria è libera, come anche il formato dei documenti.

3 Comunicazione tra i componenti

Client Web ↔ Server Web (API REST) L'API REST, come già descritto, deve modellare la gestione e acquisto dei domini Internet. 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.

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 dal protocollo di Redis, che è

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

[4] Una sorta di query SQL con il `WHERE`. Per esempio, se i documenti sono JSON, tutti i documenti che hanno un campo booleano vero.

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 (i comandi, il formato dei documenti...) deve essere generico e non specifico per la gestione dei domini Internet. Il database deve essere riutilizzabile anche in progetti di natura completamente diversa.

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

Il protocollo deve essere documentato in modo simile a come mostrato nello scheletro fornito. Si deve documentare come è fatta una richiesta e la risposta, quali sono i comandi del database e come si possono formare i documenti.

Poiché verrà valutata sia la documentazione sia 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. **Solo un membro del gruppo** deve consegnare il progetto. La consegna consiste in **un file testuale** che deve contenere:

1. Nome del progetto;
2. Nome, cognome, matricola ed email di ogni componente del gruppo;
3. Link alla repository su GitHub.

La repository su GitHub deve essere privata. Una volta consegnato il file, il proprietario della repository deve inviare un'email a Emanuele Petriglia[5] scrivendo il nome del gruppo e il link alla repository GitHub. Il docente risponde allegando una chiave pubblica SSH. Il proprietario della repository deve quindi accedere alla propria repository su GitHub e navigare in `Settings` → `Deploy keys` → `Add deploy key`. Alla schermata lo studente deve inserire un titolo qualsiasi e poi incollare il contenuto della chiave fornita via email. Al termine si deve selezionare il pulsante `Add key`.

Struttura repository La cartella principale della repository 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.

[5] emanuele.petriglia@unimib.it

- `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 per ogni gruppo da svolgere in presenza, dalla durata di circa 15-20 minuti. Durante il colloquio i docenti possono fare domande sul progetto, sulle scelte di progettazione e di implementazione. La presenza è obbligatoria per tutti i membri del gruppo. È possibile svolgere il colloquio a distanza (anche in modalità mista, es. due studenti in presenza e uno a distanza) solo se richiesto via email per motivi esplicitamente espressi.

Eventuali commit effettuati dopo il termine di consegna sono ignorati. 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. Ogni punto può essere assegnato pieno (1), nullo (0) o un intervallo tra i due. 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 macchina virtuale fornita durante il laboratorio. È importante quindi che il gruppo verifichi che il progetto funzioni all'interno di essa, si suggerisce per questo di sviluppare il progetto direttamente all'interno della macchina virtuale.