



UNIVERSITÀ DEGLI STUDI DI NAPOLI
FEDERICO II

LSO 2023/2024

Gruppo TheLibrarians

Antonio Tagliafierro
N86003148

Panico Gianpietro
N86003500

Fortino Alessandro
N86003884

***REALIZZARE LA SIMULAZIONE DI UN SISTEMA CHE
MODELLA UNA LIBRERIA PER UN NUMERO NON
PRECISATO DI UTENTI***

Sommario

1. Panoramica dell'app	3
2. Server	3
2.1 Componenti principali	3
3. Docker	4
3.1 Docker compose	4
3.2 Dockerfile	5
3.3 Integrazione tra docker compose e dockerfile	6
3.4 Ambiente di sviluppo e test	6
4. Database	6
5. Client	6
5.1 Utente.....	6
5.2 Admin	8

1. Panoramica dell'App

Il sistema permette la gestione di una libreria digitale accessibile a un numero indefinito di utenti.

Gli utenti hanno la possibilità di registrarsi e accedere alla piattaforma tramite un'interfaccia mobile, progettata per garantire un'esperienza di utilizzo fluida. La libreria fornisce ai suoi utenti un catalogo di libri ognuno dei quali caratterizzato da: ISBN, genere, nome e autore.

Gli utenti possono cercare, tramite dei filtri di ricerca, e prendere in prestito libri disponibili, fino a un massimo di K libri, limite stabilito dal libraio. Gli utenti hanno la possibilità di inserire i propri libri nel carrello per poi effettuare il checkout. Nel caso in cui un altro utente abbia preso l'ultima copia disponibile prima del completamento del prestito, l'utente riceverà un messaggio di avviso.

Il sistema prevede inoltre un meccanismo di notifica, che può inviare un promemoria agli utenti che non hanno restituito i libri entro la data di scadenza, facilitando la gestione efficiente della libreria e dei prestiti.

1.1 Struttura del Sistema

- **Server:**

- o Linguaggio: C
- o Ospitato su Google cloud

- **Database:**

- o Database Management System: PostgreSQL

- **Client:**

- o Applicazione Android
- o Linguaggio: Java

2. Server

Il server sviluppato in C è progettato per gestire la comunicazione con il client tramite socket TCP, interagendo con un database PostgreSQL per gestire dati come utenti, libri e prestiti. È strutturato per supportare la registrazione e il login degli utenti, la gestione dei libri disponibili, dei carrelli e dei prestiti, il tutto con un'architettura multithreaded che permette la gestione simultanea di più connessioni.

2.1 Componenti principali

1. **Gestione delle Connessioni**

- o Il server è in ascolto su una porta specifica (la porta 8080) e accetta le connessioni dai client tramite la funzione `listen()`. Quando una connessione è stabilita, viene creato un thread separato per gestirla, mantenendo il server libero di accettare nuove connessioni.

2. **Funzione `handle_client`**

- Questa funzione gestisce la logica principale per ogni client. Riceve comandi come "REGISTER", "LOGIN", "GET_BOOKS" dal client e chiama le funzioni appropriate per interagire con il database e rispondere al client.
- 3. **Registrazione e Login**
 - **handle_register**: Gestisce la registrazione di un nuovo utente inserendo i dati nel database PostgreSQL.
 - **handle_login**: Verifica le credenziali dell'utente contro quelle archiviate nel database e risponde con un successo o un errore.
- 4. **Gestione dei Libri**
 - **send_books_from_db**: Recupera l'elenco dei libri dal database e li invia al client.
 - Supporta filtri di ricerca come per titolo o genere e disponibilità attraverso funzioni dedicate.
- 5. **Gestione del Carrello**
 - **add_to_bag e rem_to_bag**: Aggiungono o rimuovono libri dal carrello dell'utente, aggiornando il database.
 - **books_from_bag**: Restituisce l'elenco dei libri presenti nel carrello di un utente.
- 6. **Gestione dei Prestiti**
 - **ordina_book**: Gestisce l'ordine dei libri, registrando i prestiti nel database.
 - Funzioni dedicate gestiscono i prestiti attivi, la quantità di copie disponibili e il monitoraggio dei prestiti in ritardo.
- 7. **Gestione delle Date**
 - Funzioni come `get_date` e `get_delivery_date` calcolano e gestiscono le date di prestito e restituzione.
- 8. **Interazione con PostgreSQL**
 - Utilizza `PQconnectdb()` per connettersi al database PostgreSQL.
 - Le query SQL vengono eseguite tramite `PQexec()` per operazioni come l'inserimento di utenti, la gestione dei libri e la gestione dei prestiti.
- 9. **Pulizia e Chiusura**
 - Ogni volta che un client termina la comunicazione, il server chiude il socket e libera le risorse associate.

3. Docker

Il server viene gestito tramite Docker e Docker Compose, ed è stato deciso di fare il deployment dell'immagine Docker del server nel servizio di cloud hosting Google Cloud, che offre scalabilità dinamica, elevate prestazioni e sicurezza avanzata, con infrastruttura globale e supporto per container.

3.1 Docker compose

Il file `docker-compose.yml` è uno strumento fondamentale per orchestrare i container Docker. Questo file specifica come il server C, insieme al database PostgreSQL, viene eseguito all'interno di container Docker, consentendo una gestione semplice e replicabile dell'ambiente.

- **Servizio per il server C**: nella definizione del servizio, Docker Compose utilizza un'immagine basata su `gcc:latest` come indicato nel Dockerfile. La build context specifica dove si trovano i file sorgente. Una volta costruita l'immagine, il server viene avviato eseguendo il file binario creato (`./server`).

- **Servizio per PostgreSQL:** il database PostgreSQL è un altro servizio specificato nel file. Docker Compose scarica l'immagine ufficiale di PostgreSQL, configurando automaticamente il database al primo avvio tramite le variabili POSTGRES_USER, POSTGRES_PASSWORD e POSTGRES_DB. Viene usata la sezione volumes per garantire la persistenza dei dati.
- **Networking tra i Container:** Docker Compose crea una rete virtuale per permettere la comunicazione tra il server e il database PostgreSQL. Il server C si connette al container PostgreSQL utilizzando il nome del servizio come hostname, garantendo un'interazione sicura gestita tramite libpq.
- **Volumi e persistenza:** il file docker-compose.yml include anche la definizione di volumi per garantire la persistenza dei dati nel container PostgreSQL, essenziale per mantenere i dati anche dopo l'arresto o la ricreazione dei container.

3.2 Dockerfile

Il Dockerfile definisce come viene costruito il container per eseguire il server C e include la configurazione di base, l'installazione delle dipendenze necessarie, la compilazione del server e l'esposizione delle porte per la comunicazione con i client.

- **Base Image:** il Dockerfile utilizza ubuntu:latest come immagine di base, fornendo un ambiente flessibile per installare le dipendenze necessarie come gcc e libpq. Si parte quindi da un sistema operativo minimale, configurato per eseguire il server.
- **Impostazioni di ambiente:** le variabili DEBIAN_FRONTEND=noninteractive e TZ=Europe/Rome vengono utilizzate per disabilitare i prompt interattivi durante l'installazione dei pacchetti e per impostare il fuso orario corretto
- **Installazione delle dipendenze:** il comando RUN apt-get update && apt-get install -y ... installa le dipendenze necessarie, tra cui gcc per compilare il server, make per automatizzare processi di build, libpq-dev per interfacciarsi con PostgreSQL, e postgresql-all per test locali del database.
- **Compilazione del server:** la directory di lavoro è impostata su /app e il codice sorgente server.c viene compilato con gcc. Il comando -I/usr/include/postgresql specifica il percorso delle librerie header di PostgreSQL, mentre -lpq serve per collegare la libreria PostgreSQL.

- **Esposizione della porta:** il Dockerfile espone la porta 8080, che consente ai client di connettersi al server attraverso questa porta.
- **Avvio del Server:** il Dockerfile avvia il container con una shell Bash (CMD ["/bin/bash"]), ma può essere configurato per eseguire direttamente il server (CMD ["/server"]) o uno script personalizzato come start.sh.

3.3 Integrazione tra docker compose e dockerfile

Docker Compose gestisce i servizi, mentre il Dockerfile definisce il build e l'esecuzione del server. Docker Compose utilizza il Dockerfile per costruire l'immagine del server C, permettendo al server di connettersi al database PostgreSQL tramite la rete Docker creata.

3.4 Ambiente di sviluppo e test

Docker Compose rende l'ambiente di sviluppo coerente e replicabile, con la possibilità di riavviare facilmente i container e mantenere la persistenza dei dati attraverso i volumi.

4. Database

TheLibrariansApp è progettata per gestire una libreria digitale, permettendo la registrazione di utenti, la gestione dei libri e il monitoraggio dei prestiti. Include:

1. **users:** memorizza i dettagli degli utenti, garantendo unicità dei nomi utente.
2. **books:** gestisce i libri con dettagli come ISBN, genere (ENUM personalizzato), quantità e copie prestate.
3. **bag:** traccia i libri aggiunti dagli utenti al carrello.
4. **loan:** registra i prestiti con informazioni su date e stato (attivo/consegnato).

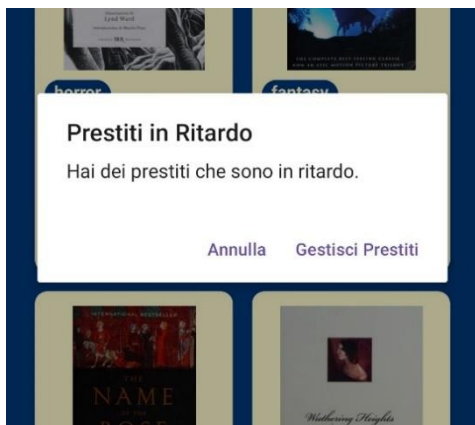
Il database include anche controlli per garantire che non vengano ricreate tabelle o tipi esistenti, garantendo una gestione robusta e dinamica della libreria.

5. Client

Il client è stato realizzato su android studio in java, ed è suddiviso in una sezione per gli utenti ed una esclusivamente per l'admin.

5.1 Utente

All'avvio dell'applicazione l'utente visualizzerà la schermata per accedere, o nel caso non si sia ancora registrato potrà farlo attraverso l'apposita schermata.

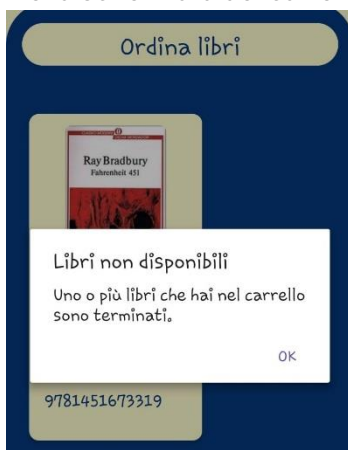


Una volta effettuato l'accesso si ritroverà nella home page dell'applicazione e verremo avvisati se abbiamo prestiti in ritardo di consegna attraverso una dialog.



Nella Home sono visualizzabili tutti i libri presenti nel server. L'utente potrà cercare qualsiasi libro con appositi filtri di ricerca che gli permettono di cercare un libro per parole chiave, quindi anche per titolo, mostrare solo i libri attualmente disponibili, e cercare un libro per genere. E' permessa anche la ricerca combinata di questi filtri.

Una volta selezionato un libro è possibile visualizzare tutti i suoi dettagli e aggiungerlo al carrello. Nella schermata del carrello, l'applicativo ci avviserà se un libro è terminato attraverso una dialog e attraverso il tasto "Ordina libri" è possibile creare il prestito dei libri all'interno del carrello.

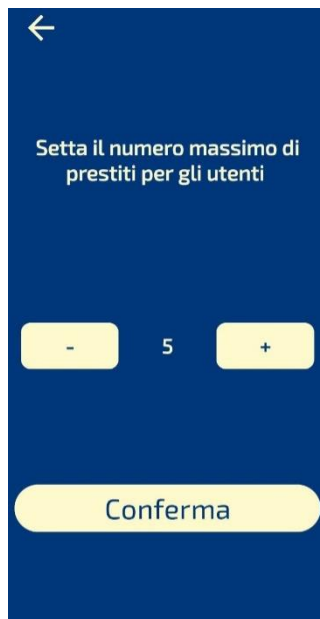


Nella schermata del profilo è possibile visualizzare il numero di prestiti attivi dell'utente e il numero massimo di prestiti attivi contemporaneamente. Invece, cliccando sul tasto "Storico prestiti" sarà possibile avere una visione più dettagliata dei singoli prestiti e sarà anche possibile riconsegnare il libro grazie all'apposito tasto.

5.2 Admin

All'avvio dell'applicazione, dalla schermata di accesso/registrazione, l'admin potrà accedere alla sezione a lui dedicata solo inserendo delle credenziali uniche, in suo possesso, che non sono accessibili agli altri utenti.

Una volta effettuato l'accesso l'admin si ritroverà nella home page in cui è visualizzabile la lista di tutti i libri presenti nel server.



L'admin, cliccando sul bottone "Prestiti", presente nella home page può settare il numero massimo di prestiti effettuabili per ciascun utente.



L'admin, cliccando sulla scheda di un libro o cercando l'ISBN nell'apposita barra di ricerca presente nella home page, accederà a una pagina dedicata, dove troverà informazioni dettagliate sul libro, come il titolo, il genere, ecc., insieme ai dettagli relativi ai prestiti. In particolare, in questa pagina, l'admin potrà visualizzare il numero di copie totali del libro, il numero di copie in prestito, la lista dei prestiti attivi e quella dei prestiti in ritardo.