UNIVERSITA' DEGLI STUDI DI PERUGIA

Dipartimento di Ingegneria

Corso di laurea magistrale in Ingegneria Informatica e Robotica

Curriculum Robotica

Progetto Virtual networks and Cloud computing

SVILUPPO DI UN APPLICAZIONE WEB FLASK, MYSQL CON DOCKER

Docente Gianluca Reali

Anno accademico 2022-2023

Riccardo Rossi *346626*

Introduzione

Per il progetto in questione è stata sviluppata una applicazione web che gestisce il log-in di utenti per un sito che si occupa di vendita di libri online.

Lo sviluppo di tale applicazione è stato effettuato utilizzando un framework web per Python particolarmente leggero denominato ***Flask***. La parte di storage dei dati è stata effettuata attraverso un database ***MySQL***, mentre la parte relativa alla creazione di pagine web e delle relative funzioni è stata sviluppata rispettivamente con l'utilizzo di ***HTML*** e ***Javascript***. Il tutto è stato disposto in due rispettivi container di Docker, uno dedicato all'applicazione Flask, uno dedicato allo storage dei dati.

L'utente, dopo aver aperto la pagina principale del sito, dispone di un pulsante di *log-in*, il quale, quando viene premuto, apre una rispettiva pagina dedicata all'autenticazione del lettore nel sito tramite nome, e-mail e password.

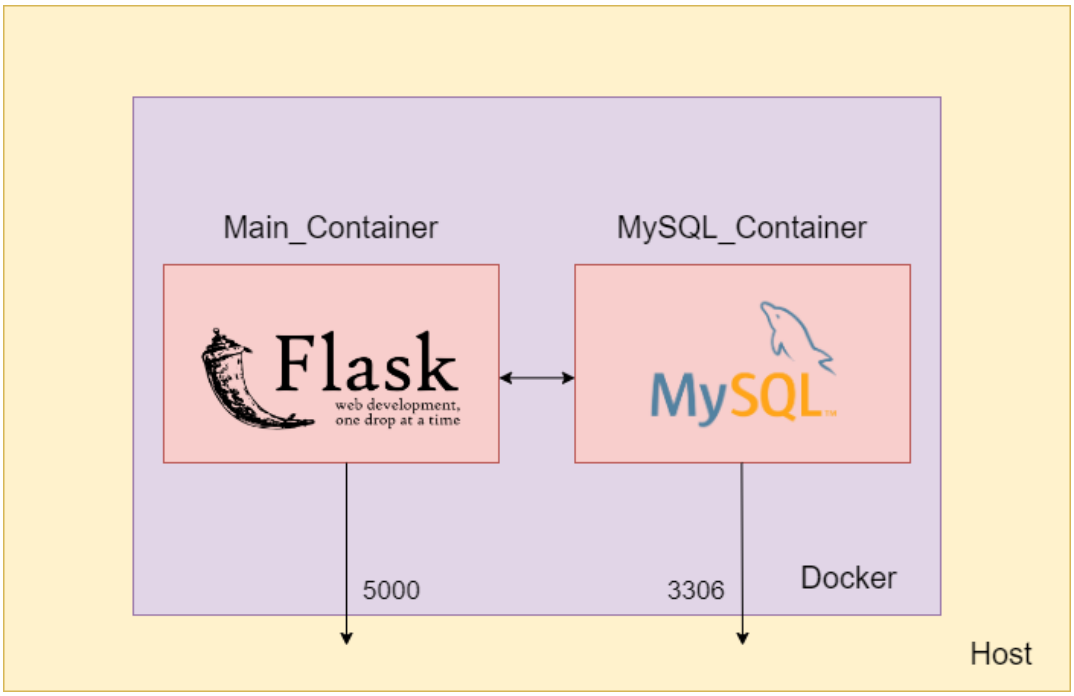

Fig.1 Schema applicazione.

Ora analizziamo da più vicino le varie componenti utilizzate per la realizzazione dell'applicazione.

App Flask

Il compito di Flask è quello di definire la struttura e i collegamenti tra l'interfaccia web realizzata in HTML e il database MySQL. Nel file '*app.py*' si può vedere come tramite la libreria '*flask-mysqldb*', l'app si metta in contatto con il database creando una vera e propria connessione attraverso la dichiarazione di parametri di configurazione:

```
7   # MySQL configurations
8   app.config['MYSQL_USER'] = 'ric'
9   app.config['MYSQL_PASSWORD'] = '123456'
10  app.config['MYSQL_DB'] = 'bucketlist'
11  app.config['MYSQL_HOST'] = 'db'
12  app.config['MySQL_PORT'] = 3306
```

Fig.2 Parametri di configurazione della connessione con il database.

Database

Come RDBMS è stato utilizzato MySQL, anche se, oggi giorno con l'avvento dei big data si è portati a pensare che esistano solo quelli non relazionali, i database relazionali rimangono comunque un pilastro per tutte quelle applicazioni che vogliono implementare una soluzione facile e intuitiva e che, soprattutto, facciano utilizzo di dati ben strutturati.

Il database creato contiene una tabella con tutte le informazioni dell'utente registrato al sito in questione, ovvero nome, e-mail e password:

```
3   CREATE TABLE bucketlist.tbl_user (
4     user_id BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
5     user_name LONGTEXT NULL,
6     user_username LONGTEXT NULL,
7     user_password LONGTEXT NULL,
8     PRIMARY KEY (user_id)
9   );
```

Fig.3 tabella degli utenti.

Inoltre, per mantenere la consistenza degli inserimenti all'interno del database, è stata definita una stored procedure che garantisca che un utente non venga inserito più di una volta:

```
DELIMITER //
CREATE DEFINER='root'@'localhost' PROCEDURE sp_createUser(IN p_name LONGTEXT, IN p_username LONGTEXT, IN p_password LONGTEXT)
BEGIN
  IF (select exists(select 1 from tbl_user where user_username = p_username)) THEN
    select 'Username Exists !!';
  ELSE
    INSERT INTO tbl_user(user_name, user_username, user_password) VALUES(p_name, p_username, p_password);
  END IF;
END//
DELIMITER ;
```

Fig.4 Stored procedure.

Docker

Per quanto riguarda il vero protagonista del progetto, dapprima è stato creato un dockerfile, il quale con una semplice sintassi, definisce i passaggi necessari per creare l'immagine ed eseguirla. In particolare, questo dockerfile è relativo all'applicazione Flask, esso scarica un'immagine *python:3.11.3* dal docker-hub, imposta la directory di lavoro */app* all'interno del container e vi ci copia al suo interno tutti i file presenti nella cartella corrente dell'host. Con il comando RUN, vengono installate le dipendenze, ovvero quelle librerie che sono state utilizzate nello sviluppo del codice. Infine, con CMD viene eseguita l'applicazione all'interno del container.

```
FROM python:3.11.3

# Set the working directory to /app
WORKDIR /app

# Copy the current directory contents into the container at /app
COPY . /app
COPY requirements.txt /app

# Install any needed packages specified in requirements.txt
RUN pip install -r requirements.txt

# Make port 80 available to the world outside this container
CMD ["python", "app.py", "flask", "run", "--host=0.0.0.0", "--port=5000"]
```

Fig.5 Docker-file.

Per far sì che l'user che utilizzerà questa applicazione possa eseguirla e renderla completamente funzionante, lanciando una singola istruzione da riga di comando, viene di seguito riportato il file *docker-compose.yaml*:

```yaml
version: '3.8'

services:

  db:
    image: mysql
    container_name: MySQL_Container
    ports:
      - "3306:3306"
    environment:
      MYSQL_ROOT_PASSWORD: 123456
      MYSQL_DATABASE: bucketlist
      MYSQL_USER: ric
      MYSQL_PASSWORD: 123456
    volumes:
      - ./db:/docker-entrypoint-initdb.d

  app:
    build: .
    container_name: Main_Container
    depends_on:
      - db
    environment:
      DB_HOST: db
      DB_PORT: 3306
      DB_DATABASE: bucketlist
      DB_USER: ric
      DB_PASSWORD: 123456
    ports:
      - 5000:5000
    links:
    - db
    command: python app.py
```

Fig.6 Docker-compose.

Diciamo che il senso è prettamente simile a quello del docker-file ma il *compose* risulta particolarmente utile nei casi in cui i servizi (e quindi container) che devono essere lanciati in un'applicazione siano molteplici. Infatti, con il comando '*docker-compose up'*, viene lanciato il docker deamon, che crea ed esegue le immagini inserendole nei container. Dopodichè l'applicazione è utilizzabile.

Interfaccia

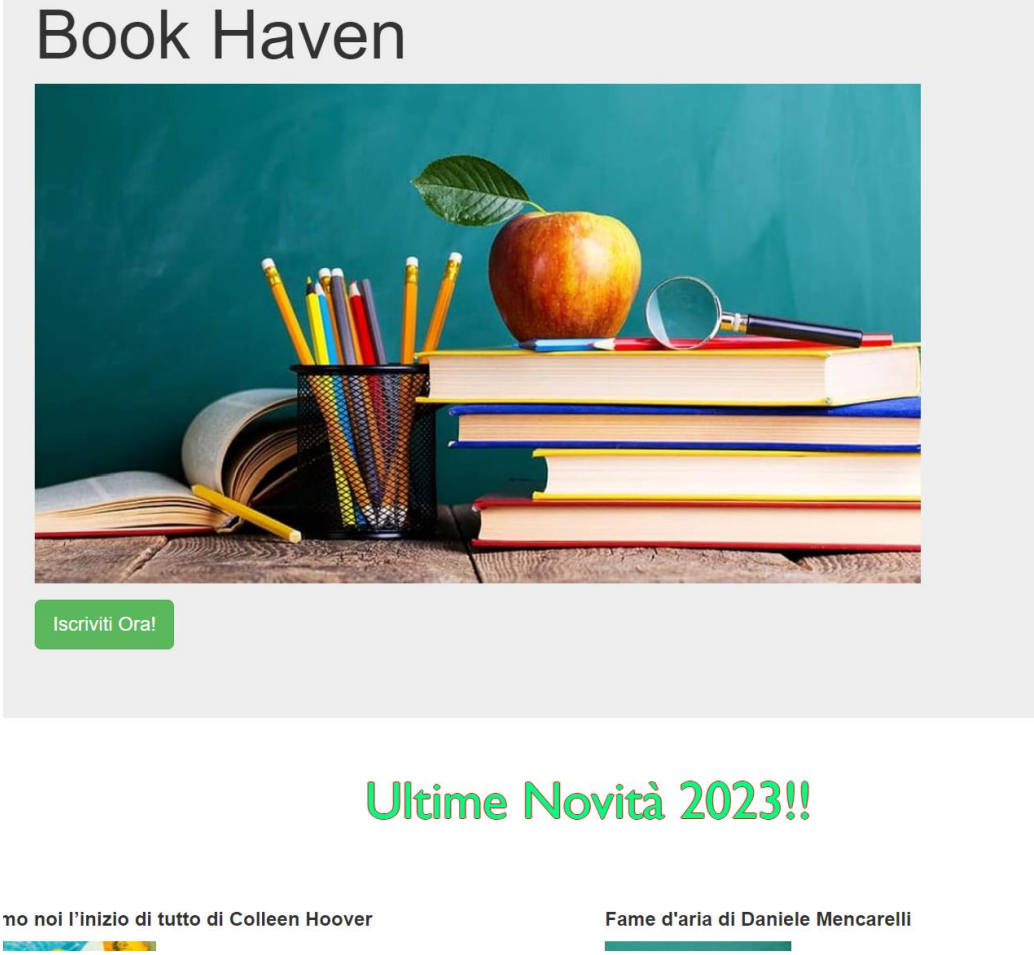

Fig.7 Interfaccia iniziale.

Attraverso il pulsante '*Iscriviti ora'*, l'utente accede alla pagina di log-in, dove inserendo le proprie credenziali e pigiando il tasto '*iscriviti*', potrà registrarsi al sito web. La verifica di quanto detto può essere fatta tramite il pulsante '*Mostra database'*.

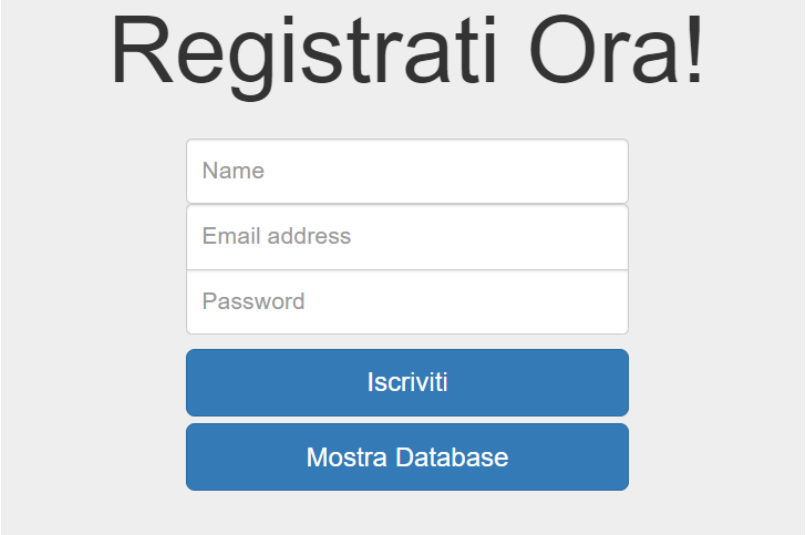

Fig.7 Interfaccia di log-in.