SVILUPPO DI UNA APPLICAZIONE CLIENT-SERVER CON DOCKER

Anno accademico 2021/22

Folco Giorgetti

Gianluca Coletta

Virtual Networks and Cloud Computing

Introduzione

In questo progetto si è realizzata una semplice applicazione client server per gestire la ricerca di insegnanti per le ripetizioni. L'applicazione è composta da tre componenti principali, un database non relazionale realizzato con MongoDB, la relativa interfaccia grafica ottenuta grazie a Mongo-Express e Node.js che si occupa dell'interazione con il database e della visualizzazione dei contenuti dinamici presenti nella pagina html. Le componenti sono state incapsulate all'interno di container per permetterne l'utilizzo e la distribuzione attraverso Docker.

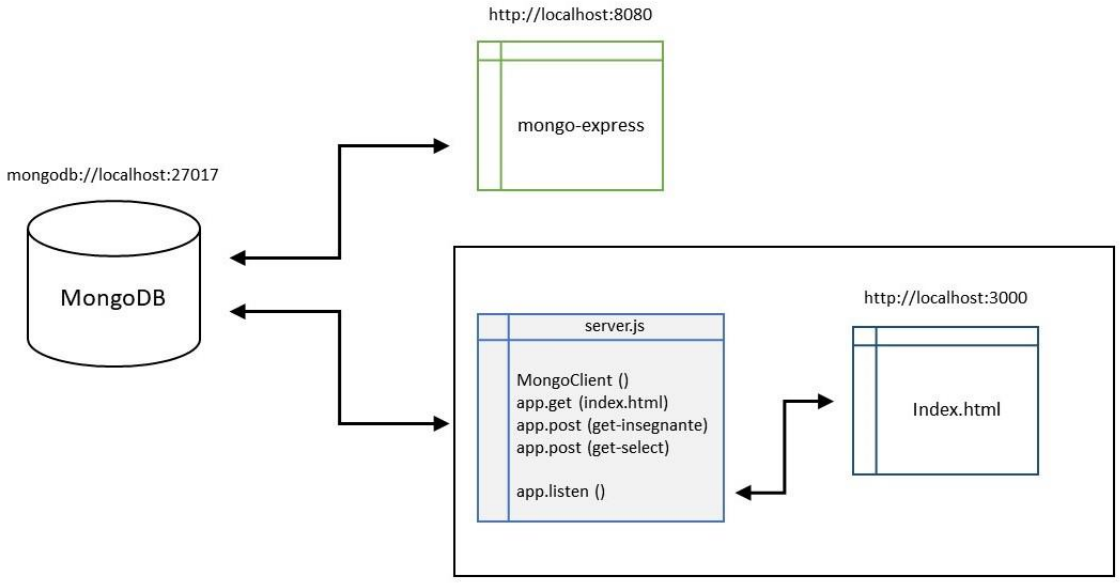

Figura 1: struttura WebApp

Database

Il database MongoDB è stato realizzato mediante l'uso dell'immagine pubblica 'Mongo' presente all'interno di Docker Hub. Inoltre, abbiamo utilizzato l'immagine Mongo-express che permette di accedere ad un'interfaccia per gestire il database di MongoDB. Entrambe le immagini sono state inserite tra i servizi nel docker-compose. All'interno del DB troviamo una serie di Insegnanti contraddistinti dai campi nome, cognome, materie, città e numero telefonico. Per semplificare l'utilizzo, al lancio dell'applicazione viene creato e popolato un DB nel caso in cui non fosse già presente.

```
MongoClient.connect(mongoUrlDocker, function (err, db) {
  if (err) throw err;
  console.log("Database created!");
  db.close();
});

MongoClient.connect(mongoUrlDocker, function (err, db) {
  if (err) throw err;
  var dbo = db.db(databaseName);


  dbo.listCollections().toArray(function (err, collections) {
    if (collections.length === 0) {
      dbo.createCollection("teachers", function (err, res) {
        if (err) throw err;
        console.log("Collection created");
        db.close();
      });
      var myobj = [
        {nome: 'Folco', cognome: 'Giorgetti', materie: ['Matematica', 'Fisica','Inglese'], città: 'Perugia', telefono: '3569638798'},
        {nome: 'Gianluca', cognome: 'Coletta', materie: ['Matematica', 'Fisica'], città: 'Perugia', telefono: '3403326423'},
        {nome: 'Leonardo', cognome: 'Lorenzi', materie: ['Chimica'], città: 'Perugia', telefono: '3324567891'},
        {nome: 'Riccardo', cognome: 'Tusino', materie: ['Latino', 'Inglese'], città: 'Bari', telefono: '3329781204'},
        {nome: 'Pietro', cognome: 'Farsi ', materie: ['Matematica', 'Fisica','Inglese', 'Latino'], città: 'Milano', telefono: '3339012343'},
        {nome: 'Paolo', cognome: 'Rossi', materie: ['Matematica', 'Informatica'], città: 'Milano', telefono: '3315622289'},
        {nome: 'Michele', cognome: 'Verdi', materie: ['Chimica', 'Biologia'], città: 'Bari', telefono: '3334568975'},
        {nome: 'Gianni', cognome: 'Bianchi', materie: ['Fisica', 'Matematica', 'Inglese', 'Informatica'], città: 'Terni', telefono: '3405695872'},
        {nome: 'Benedetta', cognome: 'Mancinelli', materie: ['Economia', 'Matematica'], città: 'Terni', telefono: '3506891235'},
        {nome: 'Paola', cognome: 'Verdi', materie: ['Fisica'], città: 'Terni', telefono: '3405623215'},
        {nome: 'Francesco', cognome: 'Pirelli', materie: ['Fisica', 'Informatica'], città: 'Bari', telefono: '3403692581'},
        {nome: 'Gianfranco', cognome: 'Pirillo', materie: ['Inglese'], città: 'Terni', telefono: '3506981237'}
      ];
      dbo.collection("teachers").insertMany(myobj, function (err, res) {
        if (err) throw err;
      });
    }
  });
});
```

Figura 2: codice implementazione DB

Backend e Frontend

Il lato server dell'applicazione è stato realizzato mediante l'uso di Node.js che si occupa della comunicazione con Mongo e della gestione delle richieste da parte del client. Per la creazione dell'immagine è stato utilizzato il campo 'build' nel file docker-compose. Per quanto riguarda il lato Frontend è stata sviluppata un'interfaccia utente tramite l'uso di codice JavaScript e HTML.

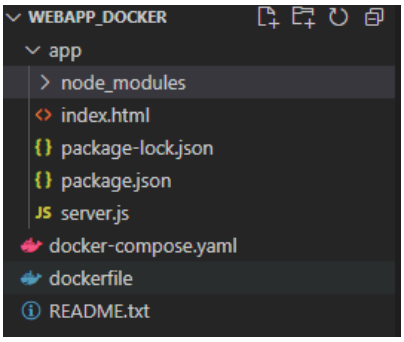

Figura 3: file della WebApp

Dockerfile

Questo file racchiude tutti i comandi per la creazione dell'immagine che si occupa del servizio web sia lato utente che lato server; una volta importata l'immagine di Node.js viene creata la directory /app nel container e vengono copiati al suo interno i file necessari per utilizzare l'applicazione. Infine, viene eseguita l'istruzione 'npm install' all'interno della directory tramite il comando 'RUN'.

dockerfile

```
FROM node:latest

ENV MONGO_DB_USERNAME=admin \
    MONGO_DB_PWD=password

RUN mkdir -p /home/app

COPY ./app /home/app

# set default dir so that next commands executes in /home/app dir
WORKDIR /home/app

# will execute npm install in /home/app because of WORKDIR
RUN npm install
```

Figura 4: codice dockerfile

Docker-Compose

Il file Docker-Compose, scritto in formato yaml, rappresenta l'insieme dei comandi e delle features che caratterizzano i singoli servizi. Ogni servizio viene generato autonomamente a partire dall'immagine specificata nel campo 'image', la quale viene inserita in un container. Per ogni immagine vengono anche specificate le porte su cui i servizi verranno avviati. Con il comando 'docker-compose -f docker-compose.yaml up' viene lanciato il docker deamon, che crea ed esegue le immagini inserendole nei container. Successivamente è possibile utilizzare la Web App.

docker-compose.yaml

```
version: '3'
services:

  server:
    build: .
    container_name: tesina
    ports:
    - 3000:3000
    command: node server.js
  #tesina:
  #  image: tesina:1.0
  #  ports:
  #    - 3000:3000
  mongodb:
    image: mongo
    ports:
      - 27017:27017
    environment:
      - MONGO_INITDB_ROOT_USERNAME=admin
      - MONGO_INITDB_ROOT_PASSWORD=password

  mongo-express:
    image: mongo-express:0.54.0
    restart: always
    ports:
      - 8080:8081
    environment:
      - ME_CONFIG_MONGODB_ADMINUSERNAME=admin
      - ME_CONFIG_MONGODB_ADMINPASSWORD=password
      - ME_CONFIG_MONGODB_SERVER=mongodb
```

Figura 5: codice docker-compose.yaml

Interfaccia utente

L’interfaccia permette all’utente di selezionare una città e una materia (verranno mostrate solo quelle presenti nel DB) per cercare gli insegnanti relativi ai due campi. Successivamente premendo il bottone ‘Cerca Insegnante’ verrà mandata una richiesta al server in ascolto sulla porta 3000, il quale tramite delle funzioni effettuerà delle richieste al container MongoDB in ascolto sulla porta 27017. Successivamente verrà mostrata a schermo una tabella contenente solo gli insegnanti coerenti con la richiesta dell’utente.

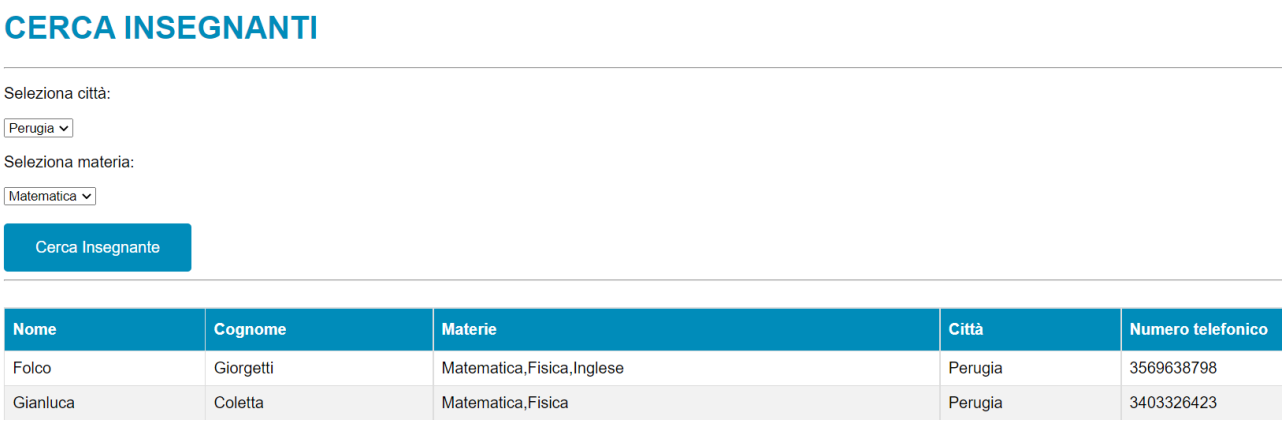

Figura 6: interfaccia utente WebApp