A.D. 1308
unipg
DIPARTIMENTO
DI INGEGNERIA

Progetto di
Virtual Networks and Cloud Computing
Corso di Laurea in Ingegneria Informatica e Robotica
Curriculum Data Science – A.A. 2023-2024
Dipartimento di Ingegneria

docente
Prof. Gianluca Reali

Docker Web Application

363433 **Gian Marco Ferri** gianmarco.ferri@studenti.unipg.it

Indice

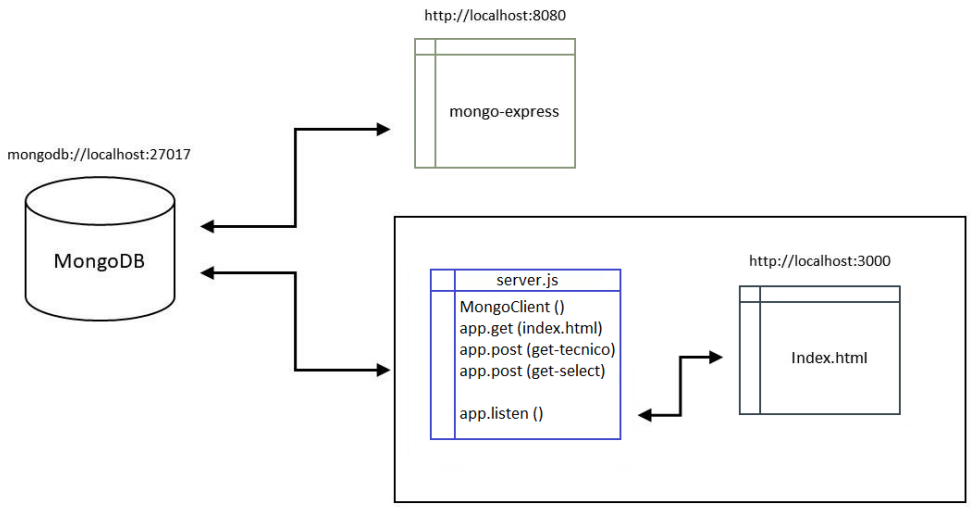
1. Introduzione

In questo progetto è stata sviluppata una semplice applicazione client-server per gestire la ricerca di tecnici specializzati.

L'applicazione è composta da tre componenti principali:

- Un **database** non relazionale realizzato con MongoDB.
- Un'**interfaccia grafica** ottenuta tramite Mongo-Express.
- **Node.js**, utilizzato per l'interazione con il database e per la visualizzazione dei contenuti dinamici presenti nella pagina HTML.

Ogni componente è stato racchiuso in un container per consentire la distribuzione e l'esecuzione tramite Docker.

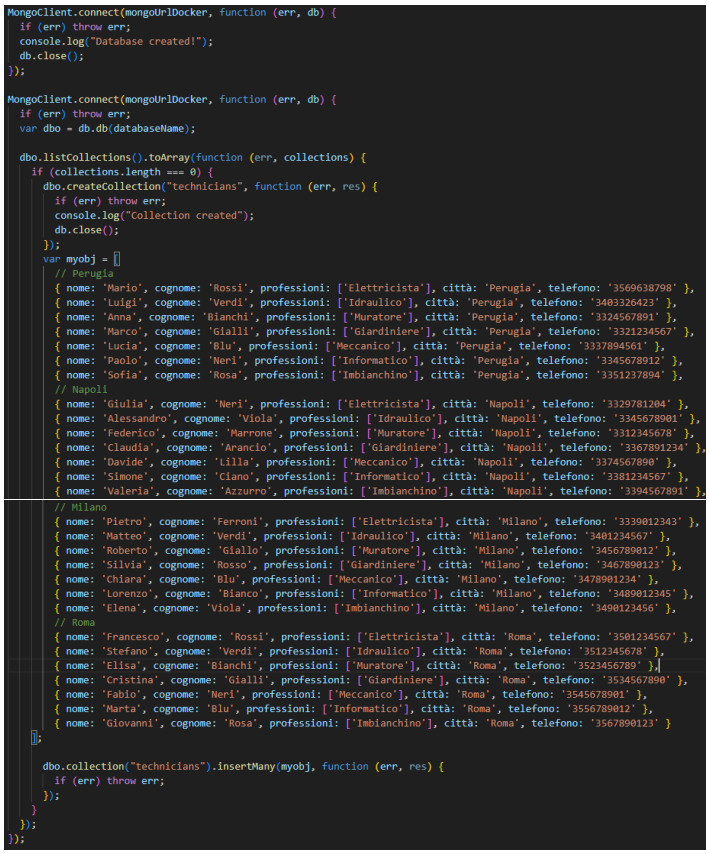


Struttura Web Application

2. Database

Il database MongoDB è stato implementato utilizzando l'immagine pubblica "Mongo" disponibile su Docker Hub. Inoltre, per fornire un'interfaccia di gestione, è stata impiegata l'immagine Mongo-Express. Entrambe le immagini sono state incluse tra i servizi nel file "docker-compose".

Nel database sono memorizzati dati relativi a diversi tecnici specializzati, identificati dai campi "Nome", "Cognome", "Professioni", "Città" e "Numero telefonico". All'avvio dell'applicazione, viene creato e popolato un database, qualora 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("technicians", function (err, res) {
        if (err) throw err;
        console.log("Collection created");
        db.close();
      });
      var myobj = [
        // Perugia
        { nome: 'Mario', cognome: 'Rossi', professioni: ['Elettricista'], città: 'Perugia', telefono: '3569638798' },
        { nome: 'Luigi', cognome: 'Verdi', professioni: ['Idraulico'], città: 'Perugia', telefono: '3403326423' },
        { nome: 'Anna', cognome: 'Bianchi', professioni: ['Muratore'], città: 'Perugia', telefono: '3324567891' },
        { nome: 'Marco', cognome: 'Gialli', professioni: ['Giardiniere'], città: 'Perugia', telefono: '3321234567' },
        { nome: 'Lucia', cognome: 'Blu', professioni: ['Meccanico'], città: 'Perugia', telefono: '3337894561' },
        { nome: 'Paolo', cognome: 'Neri', professioni: ['Informatico'], città: 'Perugia', telefono: '3345678912' },
        { nome: 'Sofia', cognome: 'Rosa', professioni: ['Imbianchino'], città: 'Perugia', telefono: '3351237894' },
        // Napoli
        { nome: 'Giulia', cognome: 'Neri', professioni: ['Elettricista'], città: 'Napoli', telefono: '3329781204' },
        { nome: 'Alessandro', cognome: 'Viola', professioni: ['Idraulico'], città: 'Napoli', telefono: '3345678901' },
        { nome: 'Federico', cognome: 'Marrone', professioni: ['Muratore'], città: 'Napoli', telefono: '3312345678' },
        { nome: 'Claudia', cognome: 'Arancio', professioni: ['Giardiniere'], città: 'Napoli', telefono: '3367891234' },
        { nome: 'Davide', cognome: 'Lilla', professioni: ['Meccanico'], città: 'Napoli', telefono: '3374567890' },
        { nome: 'Simone', cognome: 'Ciano', professioni: ['Informatico'], città: 'Napoli', telefono: '3381234567' },
        { nome: 'Valeria', cognome: 'Azzurro', professioni: ['Imbianchino'], città: 'Napoli', telefono: '3394567891' },
        // Milano
        { nome: 'Pietro', cognome: 'Ferroni', professioni: ['Elettricista'], città: 'Milano', telefono: '3339012343' },
        { nome: 'Matteo', cognome: 'Verdi', professioni: ['Idraulico'], città: 'Milano', telefono: '3401234567' },
        { nome: 'Roberto', cognome: 'Giallo', professioni: ['Muratore'], città: 'Milano', telefono: '3456789012' },
        { nome: 'Silvia', cognome: 'Rosso', professioni: ['Giardiniere'], città: 'Milano', telefono: '3467890123' },
        { nome: 'Chiara', cognome: 'Blu', professioni: ['Meccanico'], città: 'Milano', telefono: '3478901234' },
        { nome: 'Lorenzo', cognome: 'Bianco', professioni: ['Informatico'], città: 'Milano', telefono: '3489012345' },
        { nome: 'Elena', cognome: 'Viola', professioni: ['Imbianchino'], città: 'Milano', telefono: '3490123456' },
        // Roma
        { nome: 'Francesco', cognome: 'Rossi', professioni: ['Elettricista'], città: 'Roma', telefono: '3501234567' },
        { nome: 'Stefano', cognome: 'Verdi', professioni: ['Idraulico'], città: 'Roma', telefono: '3512345678' },
        { nome: 'Elisa', cognome: 'Bianchi', professioni: ['Muratore'], città: 'Roma', telefono: '3523456789' },
        { nome: 'Cristina', cognome: 'Gialli', professioni: ['Giardiniere'], città: 'Roma', telefono: '3534567890' },
        { nome: 'Fabio', cognome: 'Neri', professioni: ['Meccanico'], città: 'Roma', telefono: '3545678901' },
        { nome: 'Marta', cognome: 'Blu', professioni: ['Informatico'], città: 'Roma', telefono: '3556789012' },
        { nome: 'Giovanni', cognome: 'Rosa', professioni: ['Imbianchino'], città: 'Roma', telefono: '3567890123' }
      ];

      dbo.collection("technicians").insertMany(myobj, function (err, res) {
        if (err) throw err;
      });
    }
  });
});
```

Codice implementazione database

3. Backend e Frontend

Backend

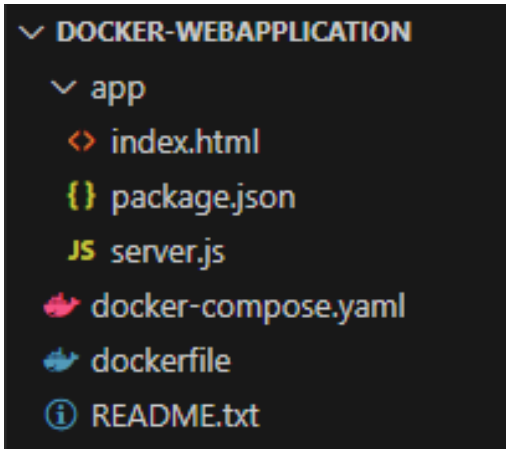
Il lato server dell'applicazione è stato sviluppato utilizzando Node.js, che si occupa della comunicazione con Mongo e delle richieste dei client.

L'immagine per il server è stata creata utilizzando il campo 'build' nel file docker-compose.

Frontend

L'interfaccia utente è stata creata utilizzando JavaScript e HTML. Essa consente agli utenti di selezionare una città e una professione per effettuare una ricerca nel database.

Dopo aver effettuato la selezione, cliccando il pulsante "Cerca Tecnico" viene inviata una richiesta al server, che interroga MongoDB e mostra una tabella con i risultati.



Directory della Web Application

4. Dockerfile

Il Dockerfile contiene tutti i comandi necessari per creare l'immagine che gestisce il servizio web, sia lato client che lato server.

Dopo aver importato l'immagine di Node.js, viene creata una directory "app" all'interno del container, dove sono copiati i file necessari per l'esecuzione dell'applicazione. Infine, viene eseguita l'istruzione "npm install" all'interno della directory tramite il comando "RUN".

dockerfile > ...

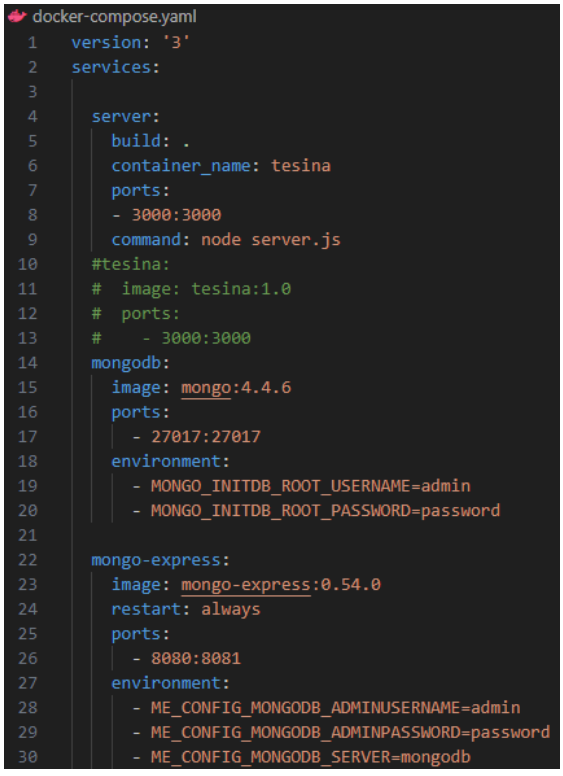
```
1   FROM node:latest
2
3   ENV MONGO_DB_USERNAME=admin \
4       MONGO_DB_PWD=password
5
6   RUN mkdir -p /home/app
7
8   COPY ./app /home/app
9
10  # set default dir so that next commands executes in /home/app dir
11  WORKDIR /home/app
12
13  # will execute npm install in /home/app because of WORKDIR
14  RUN npm install
15
16  # CMD ["node", "server.js"]
```

Codice Dockerfile

5. Docker-Compose

Il file Docker-Compose, scritto in formato YAML, descrive l'insieme dei comandi e delle caratteristiche di ciascun servizio. Ogni servizio viene creato autonomamente a partire dall'immagine specificata nel campo "image", che viene inserita in un container. Per ciascuna 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 Daemon, che crea ed esegue le immagini all'interno dei container. Successivamente è possibile utilizzare la Web Application.



Codice docker-compose.yaml

6. Interfaccia Utente

L'interfaccia utente consente di selezionare una città e una professione, mostrando solo quelle presenti nel database, per cercare i tecnici relativi ai due campi.

Quando si clicca il pulsante "Cerca Tecnico" viene inviata una richiesta al server in ascolto sulla porta 3000. Il server, tramite specifiche funzioni, effettuerà delle richieste al container MongoDB in ascolto sulla porta 27017. Verrà infine mostrata una tabella contenente solo i tecnici che corrispondono ai criteri selezionati dall'utente.

Interfaccia utente Web Application