



**HCMUTE**

**BÁO CÁO BỘ MÔN:  
ĐIỆN TOÁN ĐÁM MÂY**

**ĐỀ TÀI:  
CHƯƠNG TRÌNH QUẢN LÝ SINH VIÊN**

Giảng viên hướng dẫn:

**TS. Huỳnh Xuân Phụng**

Sinh viên thực hiện

MSSV

Trần Ngọc Hoàng

18110228

Hoàng Dương Hùng

18110296

Huỳnh Ngọc Phúc

18110338

TP. Hồ Chí Minh, tháng 01 năm 2021

**ĐIỂM SỐ**

TIÊU CHÍ	NỘI DUNG	TRÌNH BÀY	TỔNG
ĐIỂM			

**NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Ký tên

## **LỜI CẢM ƠN**

Để hoàn thành tốt đề tài báo cáo này, nhóm em xin gửi tới thầy Huỳnh Xuân Phụng lời cảm ơn chân thành và sâu sắc nhất. Trong thời gian qua thầy là người hướng dẫn giúp đỡ cho nhóm chúng em hoàn thành tốt đồ án cuối kì này.

Nhóm chúng em cũng gửi lời cảm ơn chân thành đến quý thầy cô trong Khoa đã tận tình cung cấp cho chúng em những kiến thức cần thiết giúp nhóm chúng em có thể hoàn thành đồ án được suôn sẻ và dễ dàng hơn. Không những thế nhóm chúng em cũng cảm ơn các bạn cùng khoa đã không ngại chia sẻ những kiến thức giúp nhóm hoàn thiện hơn .

Cuối lời, chúng em kính chúc quý thầy, quý cô luôn dồi dào sức khỏe và thành công hơn nữa trong sự nghiệp trồng người. Một lần nữa chúng em xin chân thành cảm ơn.

**TP.HCM, ngày 6 tháng 1 năm 2021**

**Nhóm sinh viên thực hiện**

# Mục lục

Chương 1.	Chương trình quản lý sinh viên .....	1
1.1.	Tổng quan chương trình.....	1
1.1.1.	Yêu cầu đồ án .....	1
1.1.2.	Phương hướng thực hiện.....	1
1.2.	FrontEnd.....	1
1.2.1.	Giới thiệu về phần mềm quản lý sinh viên .....	1
1.2.2.	Tổng quan phần mềm .....	2
1.3.	BackEnd .....	3
1.3.1.	Cấu trúc Packaging .....	3
1.3.2.	Controller .....	3
1.3.3.	Model .....	5
1.3.4.	Repository .....	7
1.3.5.	application.properties.....	8
1.4.	Database .....	8
Chương 2.	Docker.....	9
2.1.	Docker .....	9
2.1.1.	Images .....	9
2.1.2.	Container.....	10
2.2.	Tạo Docker file .....	10
2.2.1.	Các lệnh .....	10
2.2.2.	Tạo dockerfile cho Spring boot .....	11
2.2.3.	Tạo docker file frontend (Angular).....	12
2.3.	Tạo docker compose .....	13
2.3.1.	Lệnh .....	13
2.3.2.	Tạo docker-compose cho QLSV .....	14
2.4.	Hướng dẫn build chương trình.....	16
Chương 3.	Những điều đạt được và hướng phát triển .....	20
3.1.	Những điều đạt được.....	20
3.2.	Hướng phát triển .....	20

# Chương 1.Chương trình quản lý sinh viên

## 1.1. Tổng quan chương trình

### 1.1.1. Yêu cầu đề án

*Chương trình đơn giản( quản lý sinh viên)*

- 1 docker chạy web UI
- 1 docker chạy business( kết nối web server đến database) sử dụng Java
- 1 docker chạy database

### 1.1.2. Phương hướng thực hiện

#### 1. Xây dựng chương trình

WEB UI : xây dựng giao diện với Angular 8

BUSINESS :

FrameWork : Spring Framework

Module : Spring Boot

Quản lý thư viện : maven

DATABASE :MySQL

#### 2. Kế hoạch thực hiện

Quản lý kế hoạch được chia và quản lý trên phần mềm trello

Link : <https://trello.com/b/suvh5VF9/vi%E1%BA%Bft-%E1%BB%A9ng-d%E1%BB%A5ng-k%E1%BA%Bft-h%E1%BB%A3p-nhi%E1%BB%81u-docker-v%E1%BB%9Bi-nhauh%C3%B9ngho%C3%A0ngph%C3%BAc>

## 1.2. FrontEnd

### 1.2.1. Giới thiệu về phần mềm quản lý sinh viên

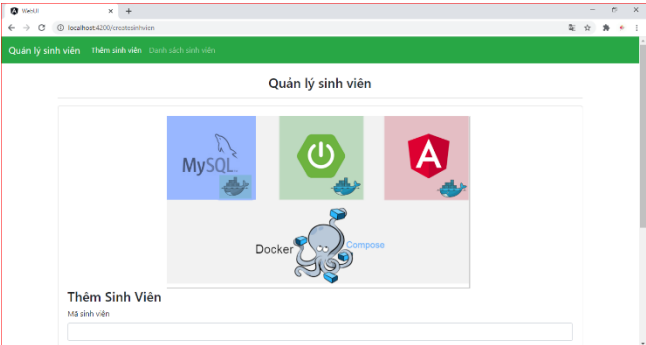
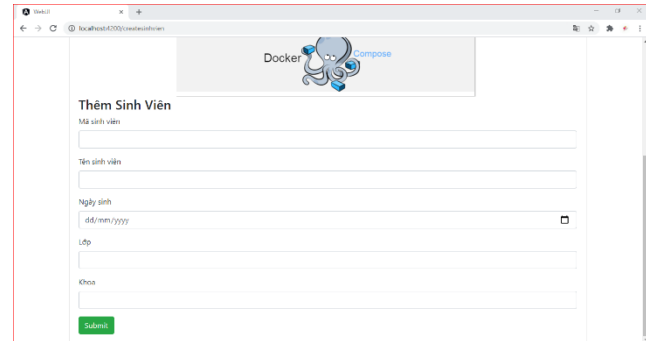
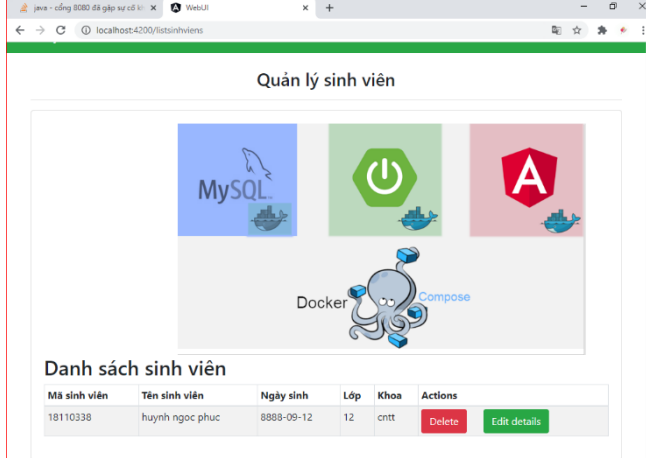
Phần mềm giúp cho người quản lý có thể biết được thông tin của các sinh viên trong lớp, giúp việc tra thông tin sinh viên nhanh chóng và nắm bắt được sĩ số của lớp

Phần mềm quản lý sinh viên đơn giản, gồm các chức năng thêm sửa xóa sinh viên

### 1.2.2. Tổng quan phần mềm

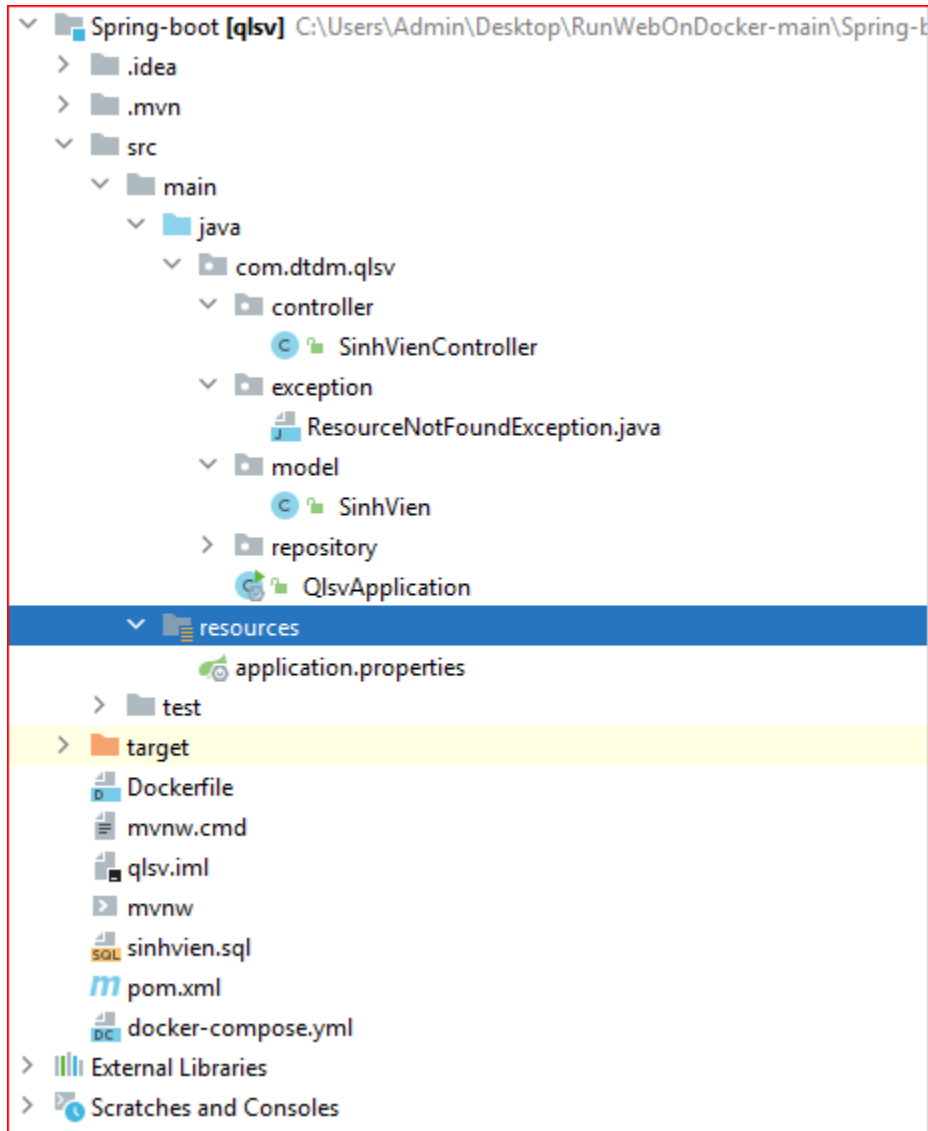
Giao diện của phần mềm được viết bằng Angular 8

Angular là một javascript framework do google phát triển để xây dựng các Single Page Application (SPA) bằng JavaScript , HTML và TypeScript . Angular cung cấp các tính năng tích hợp cho animation , http service và có các tính năng như auto-complete , navigation , toolbar , menus ,... Code được viết bằng TypeScript , biên dịch thành JavaScript và hiển thị tương tự trong trình duyệt.

stt	Chức năng	ảnh minh họa
1	Trang chủ	
2	Thêm sinh viên	
3	Xem, sửa xóa danh sách sinh viên	

## 1.3. BackEnd

### 1.3.1. Cấu trúc Packaging



### 1.3.2. Controller

Tạo các phương thức

stt	Phương thức	Chức năng
1	<code>public List&lt;SinhVien&gt; getAllsinhViens()</code>	Lấy toàn bộ data sinh viên
2	<code>public Map&lt;String, Boolean&gt; deleteSinhVien(@PathVariable(value = "id") Integer sinhVienId)</code>	Xóa sinh viên

3	<code>public SinhVien createSinhvien(@Valid @RequestBody SinhVien sinhVien)</code>	Tạo sinh viên
4	<code>public ResponseEntity&lt;SinhVien&gt; updateSinhVien(@PathVariable(value = "id")</code>	Cập nhập sinh viên

## Code

```
@RequestMapping("/api/sinhvien")
public class SinhVienController {
    @Autowired
    private SinhVienRepository sinhVienRepository;

    @GetMapping("/listsinhviens")
    public List<SinhVien> getAllsinhViens() {
        return sinhVienRepository.findAll();
    }

    @PostMapping("/createsinhvien")
    public SinhVien createSinhvien(@Valid @RequestBody SinhVien sinhVien) {
        return sinhVienRepository.save(sinhVien);
    }

    @PutMapping("/listsinhviens/{id}")
    public ResponseEntity<SinhVien> updateSinhVien(@PathVariable(value =
"id") Integer sinhVienId,
                                                    @Valid @RequestBody
SinhVien sinhVienDetails) throws ResourceNotFoundException{
        SinhVien sinhVien=sinhVienRepository.findById(sinhVienId).
            orElseThrow(() -> new ResourceNotFoundException("Không tìm
thấy sinh viên với mã :: " + sinhVienId));
        sinhVien.setMa_SV(sinhVienDetails.getMa_SV());
        sinhVien.setTen_SV(sinhVienDetails.getTen_SV());
        sinhVien.setNgay_Sinh(sinhVienDetails.getNgay_Sinh());
        sinhVien.setTen_Lop(sinhVienDetails.getTen_Lop());
        sinhVien.setTen_Khoa(sinhVienDetails.getTen_Khoa());
        final SinhVien updateSinhVien=sinhVienRepository.save(sinhVien);
        return ResponseEntity.ok(updateSinhVien);
    }
}
```



```

    }

    @DeleteMapping("/deletesinhvien/{id}")
    public Map<String, Boolean> deleteSinhVien(@PathVariable(value = "id")
Integer sinhVienId)
        throws ResourceNotFoundException{
        SinhVien sinhVien=sinhVienRepository.findById(sinhVienId).
            orElseThrow(() -> new ResourceNotFoundException("Không tìm
thấy sinh viên với mã :: " + sinhVienId));
        sinhVienRepository.delete(sinhVien);
        Map<String, Boolean> response= new HashMap<>();
        response.put("deleted", Boolean.TRUE);
        return response;
    }
}

```

### 1.3.3. Model

Tạo các thuộc tính của 1 sinh viên

Ở đây ta dùng hibernate nên ở đây ta thêm các *@Entity*, *@Table*, *@Column* để hibernate tự động tạo bảng trong database quanlysinhvien

Code

```

@Entity
@Table(name = "SINHVIEN")
public class SinhVien {
    @Id
    @Column(name = "ma_SV")
    private int ma_SV;

    @Column(name = "ten_SV")
    private String ten_SV;

    @Column(name="ngay_Sinh")
    private Date ngay_Sinh;
}

```

```

@Column(name="ten_Lop")
private String ten_Lop;

@Column(name="ten_Khoa")
private String ten_Khoa;

public SinhVien() {
}

public SinhVien(int ma_SV, String ten_SV, Date ngay_Sinh, String
ten_Lop, String ten_Khoa) {
    this.ma_SV = ma_SV;
    this.ten_SV = ten_SV;
    this.ngay_Sinh = ngay_Sinh;
    this.ten_Lop = ten_Lop;
    this.ten_Khoa = ten_Khoa;
}

public int getMa_SV() {
    return ma_SV;
}

public void setMa_SV(int ma_SV) {
    this.ma_SV = ma_SV;
}

public String getTen_SV() {
    return ten_SV;
}

public void setTen_SV(String ten_SV) {
    this.ten_SV = ten_SV;
}

public Date getNgay_Sinh() {
    return ngay_Sinh;
}

public void setNgay_Sinh(Date ngay_Sinh) {
    this.ngay_Sinh = ngay_Sinh;
}

```

```

    }

    public String getTen_Lop() {
        return ten_Lop;
    }

    public void setTen_Lop(String ten_Lop) {
        this.ten_Lop = ten_Lop;
    }

    public String getTen_Khoa() {
        return ten_Khoa;
    }

    public void setTen_Khoa(String ten_Khoa) {
        this.ten_Khoa = ten_Khoa;
    }
}

```

#### 1.3.4. Repository

Để đạt được mục đích giảm thiểu code như mình đã nói, Spring Data định nghĩa một interface chính tên là Repository nằm trong module Spring Data Common, module này sẽ được sử dụng cho tất cả các module còn lại trong Spring Data project. Nội dung của interface này đơn giản như sau:

```

package com.dtdm.qlsv.repository;

import com.dtdm.qlsv.model.SinhVien;
import org.springframework.data.jpa.repository.JpaRepository;

public interface SinhVienRepository extends JpaRepository<SinhVien, Integer>
{
}

```

Interface này sử dụng 2 generic type:

- + *Sinhvien* là domain class mà *SinhVienRepository* sẽ quản lý

- + *Integer* là kiểu dữ liệu của *Sinhvien* của domain class mà *SinhVienRepository* quản lý.

### 1.3.5. application.properties

Cấu hình Mysql trong project spring boot

Code

```
#spring.datasource.url = jdbc:mysql://mysql-standalone:3306/quanlysinhvien?useSSL=false
spring.datasource.url = jdbc:mysql://mysql-standalone:3306/quanlysinhvien
spring.datasource.username = sa
spring.datasource.password = password

spring.datasource.tomcat.test-while-idle=true
spring.datasource.tomcat.validation-query=Select 1

spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect =
org.hibernate.dialect.MySQL5InnoDBDialect

spring.jpa.hibernate.ddl-auto = update

spring.jpa.hibernate.naming.implicit-
strategy=org.springframework.boot.orm.jpa.hibernate.SpringImplicitNamingStrate
gy

server.servlet.context-path=/qlsv

server.port=8080
```

## 1.4. Database

Nền tảng MySQL

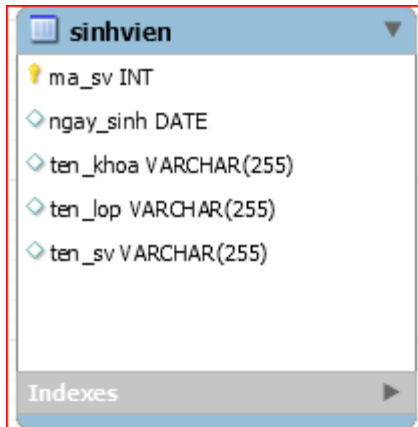
Tạo cơ sở dữ liệu

```
create database quanlysinhvien
```

Hibernate framework cung cấp phương tiện để tạo ra các bảng cơ sở dữ liệu tự động. Vì vậy, không cần phải tạo ra các bảng trong cơ sở dữ liệu bằng tay.

Trong chương trình sử dụng hibernate để tự động tạo các bảng vì vậy chúng ta không cần tạo bảng thủ công trong mysql.

Bảng sinh viên sau khi chạy chương trình



stt	tên	Kiểu dữ liệu	Nội dung
1	ma_sv	int	Mã sinh viên (khóa chính)
2	Ngày_sinh	date	Ngày sinh
3	Ten_khoa	Varchar(255)	Tên khoa
4	Ten_lop	Varchar(255)	Tên lớp
5	Ten_sv	Varchar(255)	Tên sinh viên

## Chương 2.Docker

### 2.1. Docker

**Docker** là một nền tảng để cung cấp cách để building, deploying và running ứng dụng dễ dàng hơn bằng cách sử dụng các containers (trên nền tảng ảo hóa). Ban đầu viết bằng Python, hiện tại đã chuyển sang Golang.

#### 2.1.1. Images

Docker image là một file bất biến - không thay đổi, chứa các source code, libraries, dependencies, tools và các files khác cần thiết cho một ứng dụng để chạy

Do tính chất read-only của chúng, những images này đôi khi được gọi là snapshots. Chúng đại diện cho một application và virtual environment của nó tại một thời điểm cụ thể. Tính nhất quán này là một trong những tính năng tuyệt vời của Docker. Nó cho phép các developers test và thử nghiệm phần mềm trong điều kiện ổn định, thống nhất.

Trong phần mềm quản lý sinh viên đơn giản này, ta cần tạo 3 images

stt	Tên images	Chức năng
1	<i>mysql</i>	Môi trường chạy Mysql
2	<i>quanlysinhvien</i>	Build backend
3	<i>web-ui-qlsv</i>	Build frontend

### 2.1.2. Container

Các containers cho phép lập trình viên đóng gói một ứng dụng với tất cả các phần cần thiết, chẳng hạn như thư viện và các phụ thuộc khác, và gói tất cả ra dưới dạng một package

## 2.2. Tạo Docker file

Dockerfile là một file text, trong đó chứa các dòng chỉ thị để Docker đọc và chạy theo chỉ thị đó để cuối cùng bạn có một image mới theo nhu cầu.

Chạy 1 dockerfile trên docker theo lệnh

```
docker build -t nameimage:version --force-rm -f Dockerfile .
```

-Dấu ‘. ‘ở cuối lệnh docker build ở trên, có nghĩa tìm file có tên Dockerfile ở thư mục hiện tại.

-t nameimage:version là đặt tên và tag được gán cho image mới tạo ra

### 2.2.1. Các lệnh

Cơ bản :

- + FROM : mọi Docker file đều có chỉ thị này, chỉ định image cơ sở
- + COPY ADD : sao chép dữ liệu
- + ENV : thiết lập biến môi trường
- + RUN : chạy các lệnh.
- + VOLUME : gắn ổ đĩa, thư mục
- + USER : user
- + WORKDIR : thư mục làm việc
- + EXPOSE : thiết lập cổng

**From:** Là base image để chúng ta tiến hành build một image mới. Command này phải được đặt trên cùng của Dockerfile

**Maintainer:** Command này là tùy chọn, có thể có hoặc không. Nó chứa thông tin của người tiến hành xây dựng lên images.

**Run:** Sử dụng khi muốn thực thi một command trong quá trình build image

**Copy:** Copy một file từ host machine tới docker image. Có thể sử dụng URL cho tệp tin cần copy, khi đó docker sẽ tiến hành tải tệp tin đó đến thư mục đích.

**Arg:** Định nghĩa các biến cho quá trình build docker image. Biến này chỉ có scope khi build image, không sử dụng được khi docker container running.\

Cách truyền khi chạy lệnh docker build:

--build-arg key1=value1

--build-arg key2=value2

**Env:** Định nghĩa các biến môi trường. ENV có thể được gán bằng 2 cách:

Gán giá trị mặc định thông qua biến **ARG** khi build image

Gán trực tiếp hoặc override khi run container

**Workdir:** Định nghĩa directory cho **Cmd**

**User:** Đặt user hoặc UID cho container được tạo bởi image

**Volume:** Cho phép truy cập / liên kết thư mục giữa các container và máy chủ (host machine)

**Entrypoint:** Định nghĩa những commands sẽ được chạy đầu tiên khi container chạy.\

Các lệnh thêm vào sau docker run [OPTIONS] [Extra commands] sẽ được thêm vào chuỗi các commands của entrypoint

**Cmd:** Định nghĩa các commands mặc định khi không có Entrypoint và Extra Commands

### 2.2.2. Tạo dockerfile cho Spring boot

DockerFile này để tạo images .Images này giúp cho chương trình java build trên hệ thống docker

Tạo file dockerFile cho backend(spring boot)

1. # xây dựng image mới từ image openjdk:8 trên kho lưu trữ công cộng (Docker Hub).  
(jdk 8)

Hình ảnh này do người khác chuẩn bị và chứa tất cả các phụ thuộc cần thiết mà chúng tôi cần để chạy bất kỳ ứng dụng Java nào. Chạy trên cổng 8080

- Chỉ thị ADD được dùng để thêm thư mục, file vào images.

target/quanlysinhvien.jar : thư mục nguồn.

quanlysinhvien.jar: thư mục đích.

- Đây là cổng container quanlysinhvien dùng để lắng nghe, cho phép các container khác trên cùng mạng liên lạc qua cổng này hoặc ánh xạ cổng host vào cổng này.'
- Lệnh yêu cầu Docker chạy ứng dụng, trong đó giá trị đầu tiên là một lệnh và hai giá trị cuối cùng là các tham số.

Lệnh để chạy file java

### Code

```
FROM openjdk:8
ADD target/quanlysinhvien.jar quanlysinhvien.jar
EXPOSE 8080
ENTRYPOINT ["java", "-jar", "quanlysinhvien.jar"]
```

#### 2.2.3. Tạo docker file frontend (Angular)

Tạo images để build web giao diện quản lý sinh viên bằng Dockerfile. Trong chương trình quản lý sinh viên này ta chạy trên Node.js

Các bước tạo :

- Tải node js , phiên bản cuối mới nhất
- Copy file package.json vào /app
- Chạy npm install , cài đặt npm (đây là công cụ để tạo và quản lý thư viện lập trình Javascript cho Node.js)
- Copy toàn bộ project vào workdir
- Build chương trình
- Tạo cổng



## Code mẫu

```
FROM node:latest as build-WebUI

WORKDIR /app
COPY package.json ./
RUN npm install
RUN npm install -g @angular/cli
COPY . .
RUN npm run build --prod

FROM nginx
COPY --from=build-WebUI /app/dist/WebUI /usr/share/nginx/html
EXPOSE 80
```

### 2.3. Tạo docker compose

Docker compose là công cụ dùng để định nghĩa và run multi-container cho Docker application. Với compose bạn sử dụng file YAML để config các services cho application của bạn. Sau đó dùng command để create và run từ những config đó.

- Khai báo app's environment trong Dockerfile.
- Khai báo các services cần thiết để chạy application trong file docker-compose.yml.
- Run docker-compose up để start và run app.

#### 2.3.1. Lệnh

- **version:** chỉ ra phiên bản docker-compose đã sử dụng.
- **services:** thiết lập các services(containers) muốn cài đặt và chạy.
- **image:** chỉ ra image được sử dụng trong lúc tạo ra container.
- **build:** dùng để tạo container.
- **ports:** thiết lập ports chạy tại máy host và trong container.
- **restart:** tự động khởi chạy khi container bị tắt

### 2.3.2. Tạo docker-compose cho QLSV

Ta cần build 3 container gồm mysql,backend,frontend để chạy chương trình quản lý sinh viên

Chú ý : ta cần build container mysql trước rồi mới build container backend

Ta dùng depends\_on để kiểm tra

```
depends_on:  
  - mysql-standalone
```

Các ports để container giao tiếp với máy host

images	Container	host
mysql	3306	3307
quanlysinhvien	8080	8080
web-ui-qlsv	4200	80

Code

```
version: '3'
services:
  mysql-standalone:
    image: 'mysql:8'
    environment:
      - MYSQL_ROOT_PASSWORD=password
      - MYSQL_DATABASE=quanlysinhvien
      - MYSQL_USER=sa
      - MYSQL_PASSWORD=password
    ports:
      - 3307:3306

  quanlysinhvien:
    image: 'quanlysinhvien'
    build:
      context: ./Spring-boot
      dockerfile: ./Dockerfile
    restart: on-failure
    depends_on:
      - mysql-standalone
    ports:
      - 8080:8080
    environment:
      - DATABASE_HOST=mysql-standalone
      - DATABASE_USER=sa
      - DATABASE_PASSWORD=password
      - DATABASE_NAME=quanlysinhvien
      - DATABASE_PORT=3306

  web-ui-qlsv:
    image: 'web-ui-qlsv'
    build:
      context: ./WebUI
      dockerfile: ./Dockerfile
    ports:
      - 4200:80
```

## Giải thích code

- ✓ version: '3' //chọn viết theo phiên bản 3 của docker-compose
- ✓ service: Các dịch vụ, container nằm trong services

Dịch vụ đầu tiên có tên là mysql-standalone, được build từ image mysql:8.

- ✓ port: mapping port 3306 trong container với port 3307 ở máy local. (chú ý port đằng trước là port chỉ định máy local, port sau là port trong container)
- ✓ environment: Các biến môi trường setting mặc định trong container (giải thích các biến đó để làm gì)

- ✓ quanlysinhvien: Define service thứ 2, là service chứa ứng dụng web app của project.

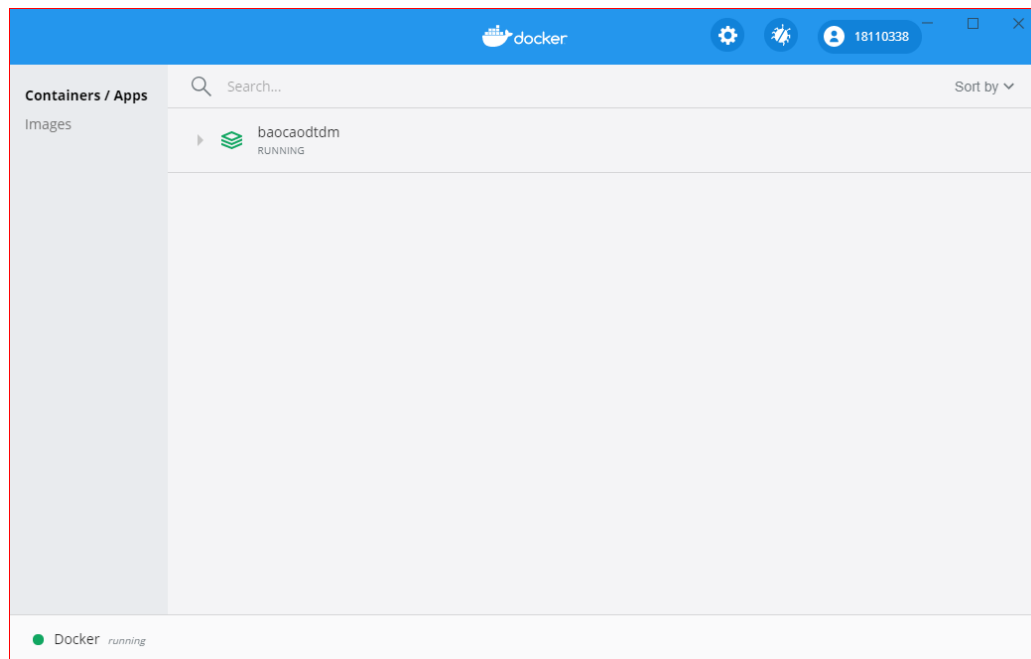
build:

- ✓ context: ./Spring-boot
- ✓ dockerfile: ./Dockerfile /// build service bằng file Dockerfile trong thư mục Spring-boot
- ✓ depends\_on: Thiết lập container quanlysinhvien phụ thuộc vào container mysql-standalone.
- ✓ restart: on-failure // Khởi động lại vùng chứa nếu nó thoát do lỗi, biểu hiện dưới dạng mã thoát khác 0.

## 2.4. Hướng dẫn build chương trình

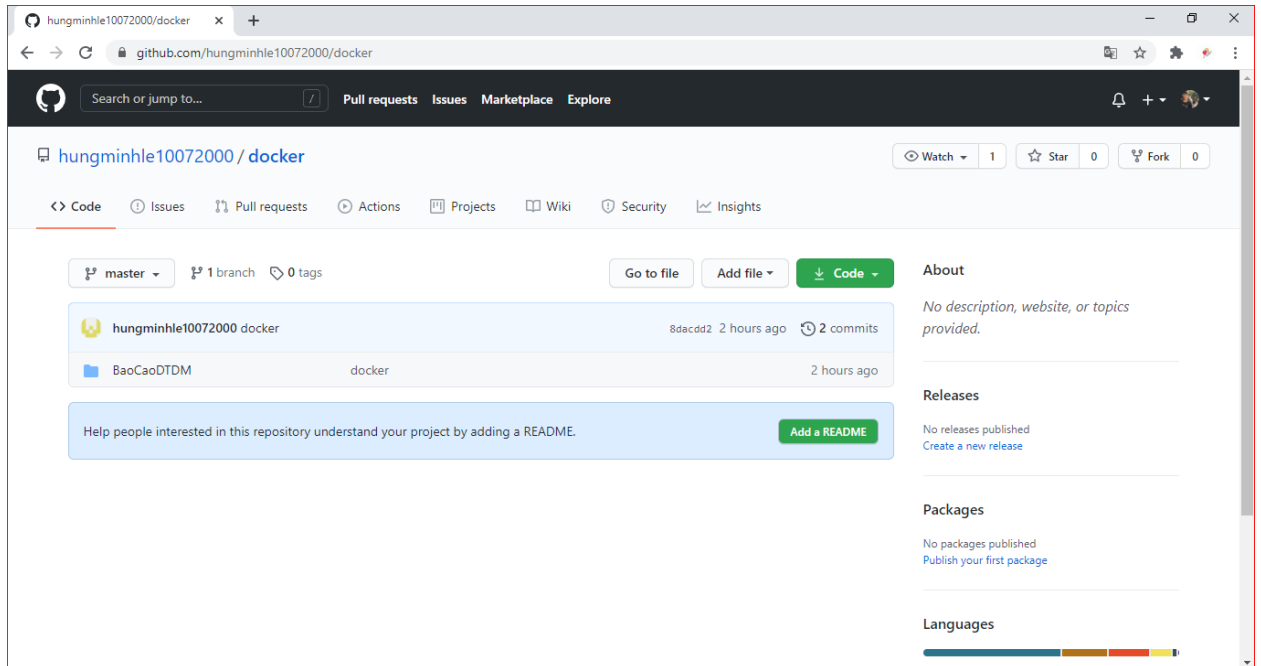
### 1. Tải docker

- Link hướng dẫn : <https://docs.docker.com/docker-for-windows/install/>
- Đăng kí tài khoản



### 2. Tải code

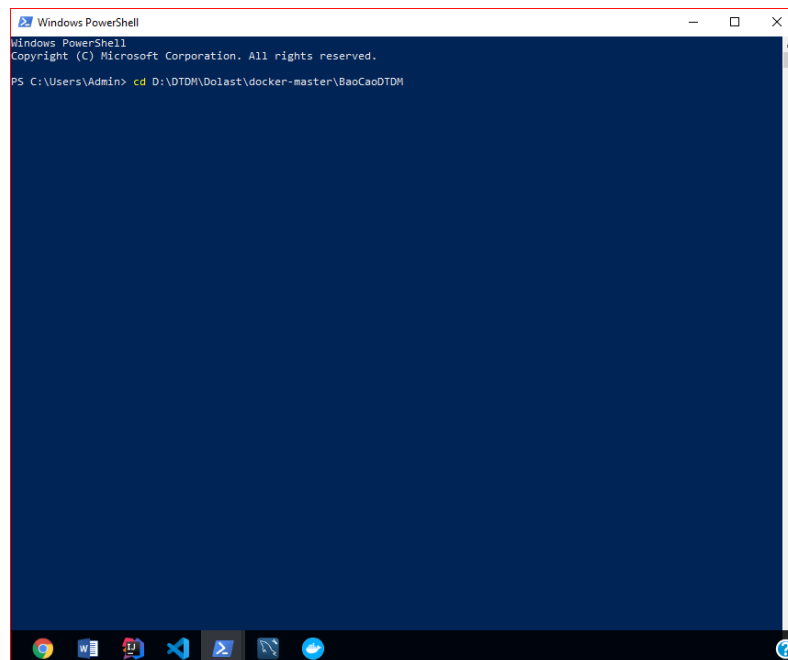
Link github : <https://github.com/hungminhle10072000/docker>



### 3. Build docker

- a. Mở window powerShell
- b. Truy cập thư mục chứa file code

Chạy lệnh cd “đường dẫn đến thư mục docker-compose”



- c. Chạy lệnh

- *Docker-compose down*

Dừng vùng chứa và xóa vùng chứa, mạng, ổ đĩa và hình ảnh được tạo bởi up.

*Minh họa*

```
PS C:\Users\Admin> cd D:\DTDM\Dolast\docker-master\BaoCaoDTDM
PS D:\DTDM\Dolast\docker-master\BaoCaoDTDM> docker-compose down
Stopping baocaodtdm_quanlysinhvien_1 ... done
Stopping baocaodtdm_web-ui-qlsv_1 ... done
Stopping baocaodtdm_mysql-standalone_1 ... done
Removing baocaodtdm_quanlysinhvien_1 ... done
Removing baocaodtdm_web-ui-qlsv_1 ... done
Removing baocaodtdm_mysql-standalone_1 ... done
Removing network baocaodtdm_default
PS D:\DTDM\Dolast\docker-master\BaoCaoDTDM>
```

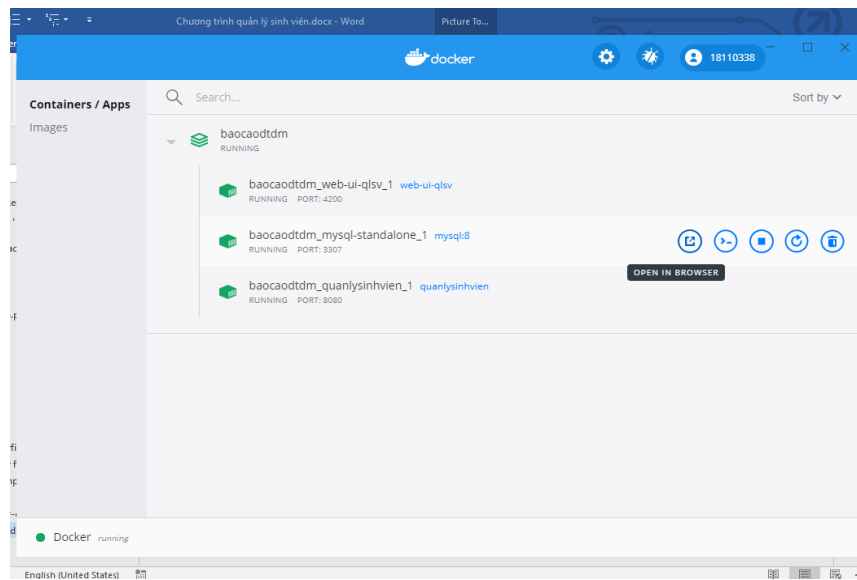
- Sau đó chạy : *Docker-compose up*

*minh họa*

```
PS D:\DTDM\Dolast\docker-master\BaoCaoDTDM> docker-compose up
baocaodtdm_mysql-standalone_1 is up-to-date
baocaodtdm_web-ui-qlsv_1 is up-to-date
Starting baocaodtdm_quanlysinhvien_1 ... done
Attaching to baocaodtdm_mysql-standalone_1, baocaodtdm_web-ui-qlsv_1, baocaodtdm_quanlysinhvien_1
mysql-standalone_1 | 2021-01-06 09:01:06+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.22-1debian10
mysql-standalone_1 | started.
mysql-standalone_1 | 2021-01-06 09:01:06+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
mysql-standalone_1 | 2021-01-06 09:01:06+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.22-1debian10
mysql-standalone_1 | started.
mysql-standalone_1 | 2021-01-06 09:01:06+00:00 [Note] [Entrypoint]: Initializing database files
mysql-standalone_1 | 2021-01-06T09:01:06.777036Z 0 [System] [MY-013169] [Server] /usr/sbin/mysqld (mysqld 8.0.22) initi
mysql-standalone_1 | progress as process 42
mysql-standalone_1 | 2021-01-06T09:01:06.786907Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
mysql-standalone_1 | 2021-01-06T09:01:07.601912Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has ended.
mysql-standalone_1 | 2021-01-06T09:01:09.856505Z 6 [Warning] [MY-010453] [Server] root@localhost is created with an emp
mysql-standalone_1 | ty password ! Please consider switching off the --initialize-insecure option.
mysql-standalone_1 | 2021-01-06 09:01:14+00:00 [Note] [Entrypoint]: Database files initialized
mysql-standalone_1 | 2021-01-06 09:01:14+00:00 [Note] [Entrypoint]: Starting temporary server
mysql-standalone_1 | 2021-01-06T09:01:15.161796Z 0 [System] [MY-010116] [Server] /usr/sbin/mysqld (mysqld 8.0.22) start
mysql-standalone_1 | ing as process 87
mysql-standalone_1 | 2021-01-06T09:01:15.185258Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
mysql-standalone_1 | 2021-01-06T09:01:15.469342Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has ended.
mysql-standalone_1 | 2021-01-06T09:01:15.617323Z 0 [System] [MY-011323] [Server] X Plugin ready for connections. Socket
mysql-standalone_1 | : /var/run/mysqld/mysqld.sock
mysql-standalone_1 | 2021-01-06T09:01:15.836003Z 0 [Warning] [MY-010068] [Server] CA certificate ca.pem is self signed.
mysql-standalone_1 | 2021-01-06T09:01:15.836268Z 0 [System] [MY-013602] [Server] Channel mysql_main configured to support
mysql-standalone_1 | rt TLS. Encrypted connections are now supported for this channel.
mysql-standalone_1 | 2021-01-06T09:01:15.841072Z 0 [Warning] [MY-011810] [Server] Insecure configuration for --pid-file
mysql-standalone_1 | : Location '/var/run/mysqld' in the path is accessible to all OS users. Consider choosing a different directory.
mysql-standalone_1 | 2021-01-06T09:01:15.878326Z 0 [System] [MY-010931] [Server] /usr/sbin/mysqld: ready for connection
mysql-standalone_1 | s. Version: '8.0.22' socket: '/var/run/mysqld/mysqld.sock' port: 0 MySQL Community Server - GPL.
mysql-standalone_1 | 2021-01-06 09:01:15+00:00 [Note] [Entrypoint]: Temporary server started
```

- Build

Sau khi chạy xong, ta vào docker xem đã có container baocaotmdt như hình:



Click vào *open in browse*

## **Chương 3. Những điều đạt được và hướng phát triển**

### **3.1. Những điều đạt được**

Về cơ bản, nhóm tự nhận xét phần mềm của nhóm đã giải quyết được được 95% yêu cầu mà đề án đặt đặt ra.

Những điều đạt được

- Biết thêm ngôn ngữ lập trình mới
- Hiểu rõ hơn về docker
- Có thể làm được chương trình web và Build trên docker
- Nhận biết được lợi ích khi sử dụng docker
- Rèn luyện các kỹ năng mềm như: Làm việc nhóm, kỹ năng tự học, đọc tài liệu tiếng anh, ...
- Rèn luyện tính kỷ luật, tự giác, tinh thần trách nhiệm.
- Rèn luyện kỹ năng làm báo cáo, thuyết trình cho đề án tốt nghiệp trong tương lai.

### **3.2. Hướng phát triển**

- Phần mềm đơn giản sẽ cập nhập thêm nhiều chức năng và giao diện đẹp hơn
- Viết code chưa được an toàn , sẽ tăng cường bảo mật
- Chỉ build trên locall, sẽ build được trên domain