**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT THÀNH PHỐ HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO BỘ MÔN:**

**ĐIỆN TOÁN ĐÁM MÂY**

TP. Hồ Chí Minh, tháng 01 năm 2021

**ĐIỂM SỐ**

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

**ĐỀ TÀI:**

**CHƯƠNG TRÌNH QUẢN LÝ SINH VIÊN**

|  |  |  |  |
| --- | --- | --- | --- |
| 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 dể dàng hơn. Không những thế nhóm chúng em cũng cảm ơn các bạn cùng khoa đã không ngại chia sẽ 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](#_Toc60860945)

[1.1. Tổng quan chương trình 1](#_Toc60860946)

[1.1.1. Yêu cầu đồ án 1](#_Toc60860947)

[1.1.2. Phương hướng thực hiện 1](#_Toc60860948)

[1.2. FrontEnd 1](#_Toc60860949)

[1.2.1. Giới thiệu về phần mềm quản lý sinh viên 1](#_Toc60860950)

[1.2.2. Tổng quan phần mềm 2](#_Toc60860951)

[1.3. BackEnd 3](#_Toc60860952)

[1.3.1. Cấu trúc Packaging 3](#_Toc60860953)

[1.3.2. Controller 3](#_Toc60860954)

[1.3.3. Model 5](#_Toc60860955)

[1.3.4. Repository 7](#_Toc60860956)

[1.3.5. application.properties 8](#_Toc60860957)

[1.4. Database 8](#_Toc60860958)

[Chương 2. Docker 9](#_Toc60860959)

[2.1. Docker 9](#_Toc60860960)

[2.1.1. Images 9](#_Toc60860961)

[2.1.2. Container 10](#_Toc60860962)

[2.2. Tạo Docker file 10](#_Toc60860963)

[2.2.1. Các lệnh 10](#_Toc60860964)

[2.2.2. Tạo dockerfile cho Spring boot 11](#_Toc60860965)

[2.2.3. Tạo docker file frontend (Angular) 12](#_Toc60860966)

[2.3. Tạo docker compose 13](#_Toc60860967)

[2.3.1. Lệnh 13](#_Toc60860968)

[2.3.2. Tạo docker-compose cho QLSV 14](#_Toc60860969)

[2.4. Hướng dẫn build chương trình 16](#_Toc60860970)

[Chương 3. Những điều đạt được và hướng phát triển 20](#_Toc60860971)

[3.1. Những điều đạt được 20](#_Toc60860972)

[3.2. Hướng phát triển 20](#_Toc60860973)

# Chương trình quản lý sinh viên

## Tổng quan chương trình

### Yêu cầu đồ 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

### 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 :MySQ

1. 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>

## FrontEnd

### 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

### Tổng quan phần mềm

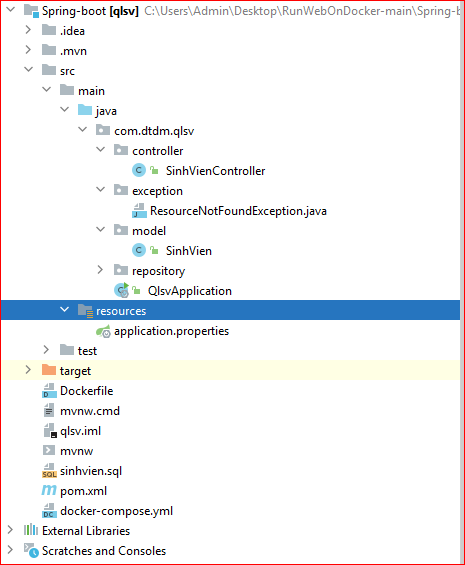
Giao diện của phần mềm được viết bằng Angular 8

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 |  |

## BackEnd

### Cấu trúc Packaging



### Controller

Tạo các phương thức

|  |  |  |
| --- | --- | --- |
| stt | Phương thức | Chức năng |
| 1 | public List<SinhVien> getAllsinhViens() | Lấy toàn bộ data sinh viên |
| 2 | public Map<String, Boolean> deleteSinhVien(@PathVariable(value = "id") Integer sinhVienId) | Xóa sinh viên |
| 3 | public SinhVien createSinhvien(@Valid @RequestBody SinhVien sinhVien) | Tạo sinh viên |
| 4 | public ResponseEntity<SinhVien> updateSinhVien(@PathVariable(value = "id") | 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;  } } |

### 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;  } } |

### 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.

### 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.SpringImplicitNamingStrategy  server.servlet.context-path=/qlsv  server.port=8080 |

## Database

Nền tảng MySQL

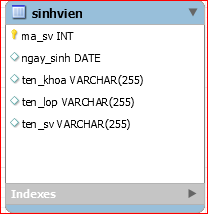
Tạo cở sở dữ liệu

|  |
| --- |
| 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 dể tự động tạo các bảng vì vậy chúng ta không cần tạo bảng thủ công trong mysql.

|  |  |  |  |
| --- | --- | --- | --- |
| stt | tên | Kiểu dữ liệu | Nội dung |
| 1 | ma\_sv | int | Mã sinh viên (khóa chính) |
| 2 | Ngay\_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 |

Bảng sinh viên sau khi chay chương trình

# Docker

## 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.

### 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 | *mysql* | Môi trường chạy Mysql |
| 2 | *quanlysinhvien* | Build backend |
| 3 | *web-ui-qlsv* | Build frontend |

### 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

## 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

### 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 chưa 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

### 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

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

target/quanlysinhvien.jar : thư mục nguồn.

quanlysinhvien.jar: thư mục đích.

1. Đây là cổng container quanlysinhvien dùng để lắng nghe, cho phép các container khác trên cùng mạng liên lạc qua công này hoặc ánh xạ cổng host vào cổng này.'
2. 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"] |

### 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 :

1. Tải node js , phiên bản cuối mới nhất
2. Copy file package.json vào /app
3. Chạy npm install , cài đặt npm (đây là công cụ để tạo và quản lý thư viện lập trình Javascript cho [Node.js](https://nodejs.org/))
4. Copy toàn bộ project vào workdir
5. Build chương trình
6. 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 |

## 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.

### 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

### 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

|  |
| --- |
| depends\_on:  - mysql-standalone |

Ta dùng denpends\_on để kiểm tra

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.

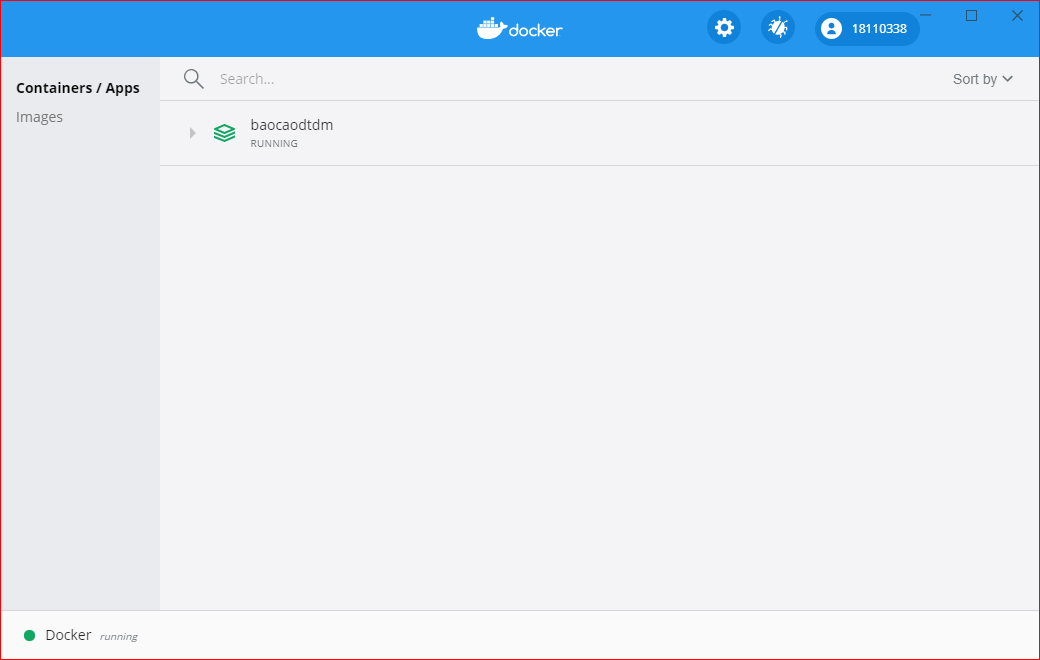
build:

* 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.

## Hướng dẫn build chương trình

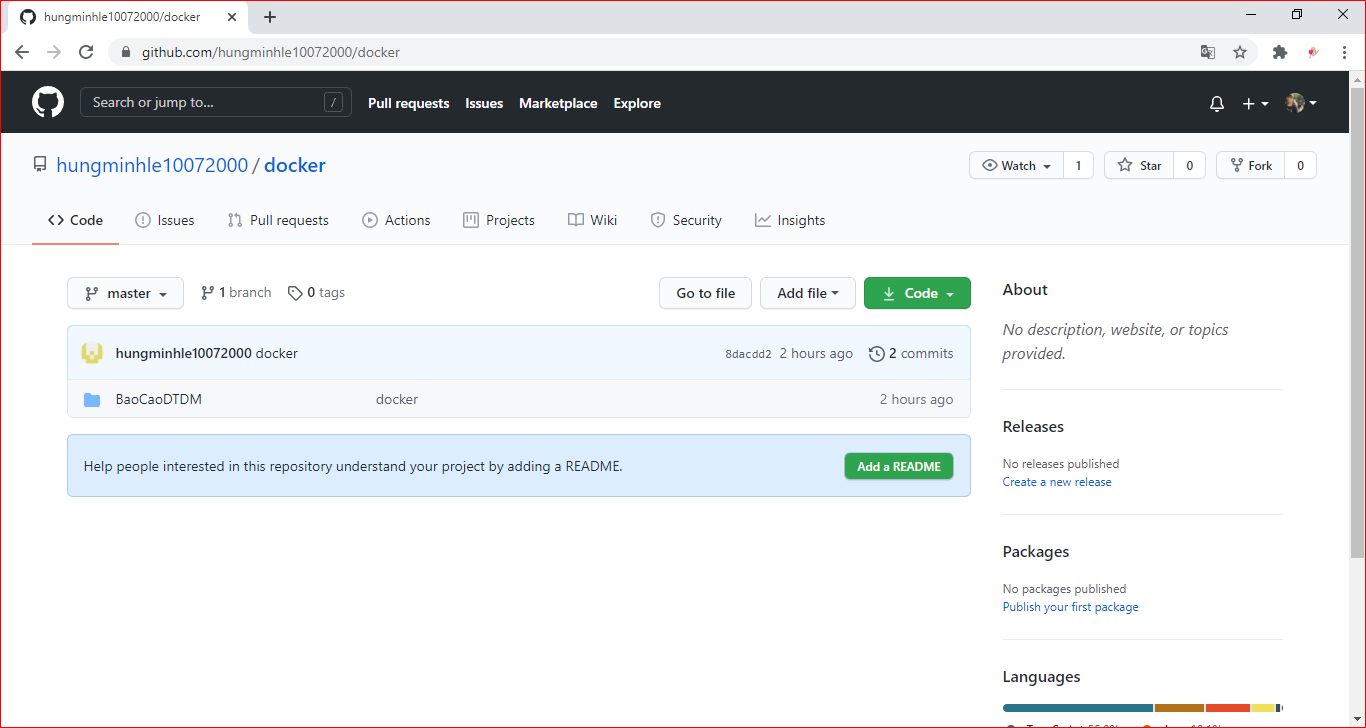
1. Tải docker

* Link hướng dẫn : [*https://docs.docker.com/docker-for-windows/install/*](https://docs.docker.com/docker-for-windows/install/)
* Đăng kí tài khoản



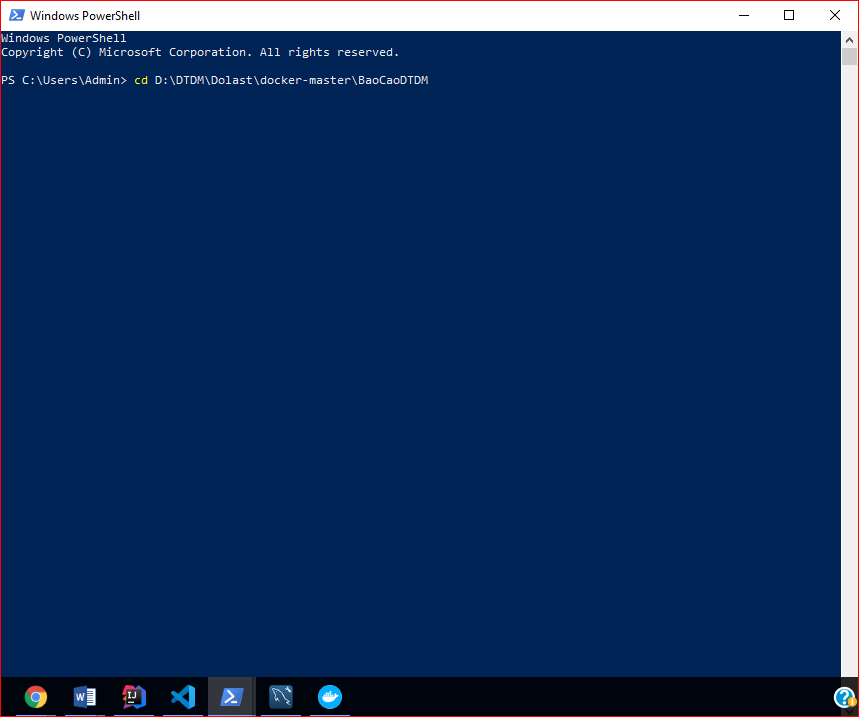
1. Tải code

Link github : *https://github.com/hungminhle10072000/docker*



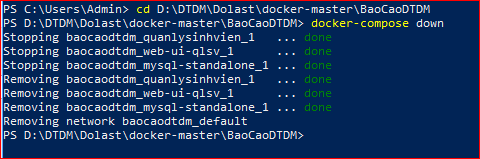
1. Build docker
   1. Mở windown powerShell
   2. Truy cập thư mục chứa file code

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

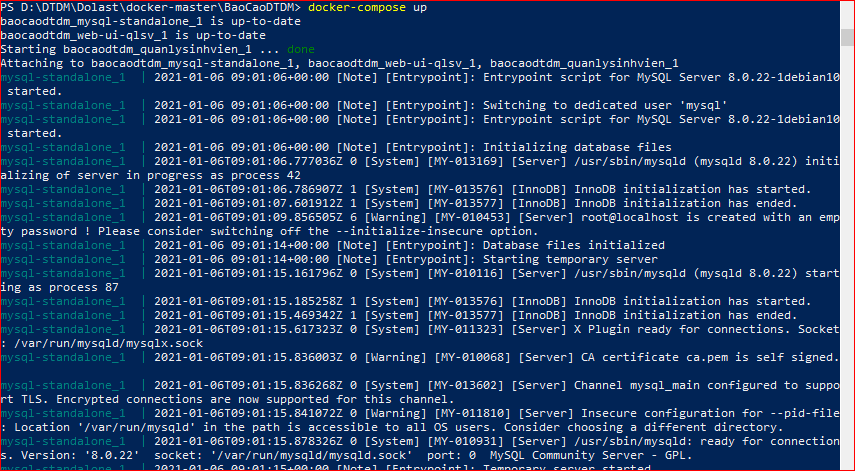


* 1. Chạy lệnh
* *Docker-compose down*

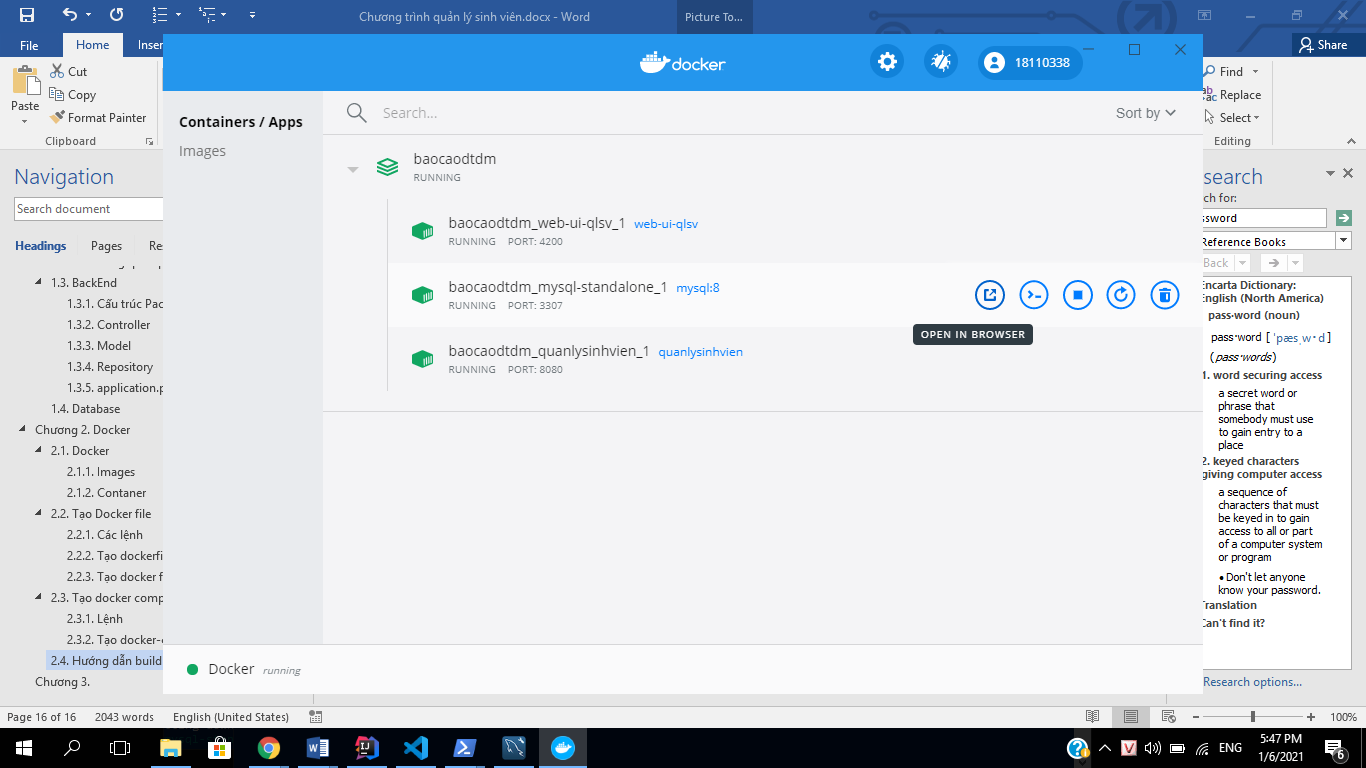
Dừng vùng chứa và xóa vùng chứa, mạng, ổ đĩa và hình ảnh được tạo bởi up.

*Minh họa*

* Sau đó chạy : *Docker-compose up*

*minh họa*

* Build

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

Click vào ***open in browse***

# Những điều đạt được và hướng phát triển

## Những điều đạt được

Về cơ bản, nhóm tự nhận xét phần mềm của nhóm đã giải quyết được được 95% yêu cầu mà đồ 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 kĩ năng mềm như: Làm việc nhóm, kỹ năng tự học, đọc tài liệu tiếng anh, …
* Rèn luyện tính kỹ luật, tự giác, tinh thần trách nhiệm.
* Rèn luyện kĩ năng làm báo cáo, thuyết trình cho đồ án tốt nghiệp trong tương lai.

## Hướng phát triển

* Phần mềm đơn giang 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