Dwebble System Documentation

Group 12

Spring, 2020

Team Members

Elias Julian Marko Garcia
Ibrahim AbuAnz

Contents

Revision History

Version	Date	Name	Description
1	2020.04.26	Elias	Initial Document

1 Introduction

`dwebble` is a Rust based application that uses the Rocket web framework that assists the UMKC CSEE department in automating the scheduling of professors and courses they teach for a given semester given the constraints on professorial staffs' schedules, conflicts between sequencing and order that classes should be taught, and other minutia involved with class scheduling.

2 Project Dependencies

It is assumed that the project will be setup on a server controlled by the University or the University of Missouri system. The specifics of the server are not important other than it have the following dependencies installed:

- Docker (version > 19.0)
  - available via docker.com or, possibly, the system's package manager (so long as version meets requirements).
- Docker Compose (version > 1.25.4)
  - Should be packaged with Docker proper, but if the install method used deviates or is abnormal, follow the guidelines provided at docker.com
- Rust (version = current nightly toolchain)
  - Ideally and easily installed directly via rustup, it is furthermore advised not to use whatever Rust version provided by the OS' software packaging system.
  - After installing, simply run `cargo build –release` to install the necessary version of Rust's nightly toolchain along with all necessary libraries and the final binary of dwebble.
- Update the production section in `Rocket.toml` to point address, port, and database URI to whatever is chosen by the University or UM IT system administrators
  - The defaults easily function for production purposes with small editing but can obviously differ based on setup.

  - Deviation from a docker oriented system is left to university/UM administrators to resolve due to the system being designed with dockerization in mind for deployment.
- Update the `docker-compose.yml` file to point to the port bindings and database URI desired
  - Again, these specifics are left to the IT administration to resolve as they see fit.

3 Access Application Code

User access to the internet, Git version control, and GitHub is assumed. The application code can be found at https://github.com/umkc-cs-451-2020-spring/dwebble/

4 Installing on Server

4.1 Installing the Application

So long as the dependencies section is met, installation is fairly trivial and is as follows:

1. Enter the project directory
2. Run `docker-compose up`
3. Enter dwebble's application directory via `cd dwebble/`
4. Run the dwebble binary via `cargo run –release`
5. The application is now running at its configured address and open port for usage within the network of deployment.

4.2 Accessing the Application

As configured, the application will only be accessible within the local network and, when defined as `localhost`, only from the same machine it is deployed.

From the same system, this would mean simply going to the address of localhost at the port opened for the application, which is defined as `localhost:8000` by default; however, accessing this web application via browser by external machines is an endeavor left to the university/UM's IT administrators to resolve given the complexities and concerns involved, and unknown, to this development team.

5 System Maintenance

5.1 Rust

Installing Rust via Rustup has multiple benefits, the primary being that updating and running a recent version of rust is trivial. `rustup update` is all that is needed to ensure that Rust is up to date. Similarly running either `cargo build –release` or `cargo run –release` will ensure that the newest version of the application’s library dependencies are downloaded, compiled, and built with a production binary of dwebble.

5.2 Docker Image (and Database)

With the current project setup, docker entirely handles the problem of installing and running a development/production database using PostgreSQL and Alpine linux. So long as the image used within the docker-compose.yml file is an up to date version of PostgreSQL, maintaining the database is trivial. The only endeavor left is to ensure a recent version of docker is used, which can be handled by the software packaging system if a Linux derivative is used as the server OS or via the install instructions provided earlier.