Trailhead Checkin – Sprint 2 Reflection

For sprint 2, my team started working on the development of the app for our sponsor. Our instructor suggested using the Laravel PHP framework and a Docker container for deployment to deal with dependencies. The goal for this sprint was to familiarize ourselves with the Laravel framework and practice building an app using it. There are several videos online where developers do walkthroughs explaining the system, the directory structure, and how to use the artisan command to generate controllers and migrate database schemas.

I chose to use the “Laracasts” video series and went through the getting started video for Laravel 5.7. The app that this walkthrough creates isn’t very complex, but it covers the basic of the system. One of the things that stuck out to me was all the work that the framework does for the developer. In just a couple of commands, Laravel builds a skeleton app and deploys a local server to localhost:8000. Developers can create layouts so that several pages will look the same, and the specific parts of each page are differentiated from the layout using the @yield and @section built-ins.

Laravel uses the Model-View-Controller design pattern to separate the responsibilities of the app. Routes are specified in a simple routing file, with different responses triggered depending on the type of HTTP request. Database migrations are performed and rolled back with artisan migrate. There is a built-in versioning system for the changes to the database which I think is convenient.

One thing that the Laracast tutorial was a little lacking in was the deployment process. When the app is first created, there’s a .gitignore file that removes some directories and files from version control, making some changes necessary when deploying to a server versus running the app locally. We had a simple starting point for the app ready to show our client, but we couldn’t get it to deploy on Heroku, we kept getting HTTP 500 errors. It turned out that pointing Heroku to our github repo wasn’t working, but using the Heroku CLI for deployment does, as long as the repo root is correct and the Procfile is correct. We deployed the starter app to Heroku, so our client had something to interact with and give feedback.