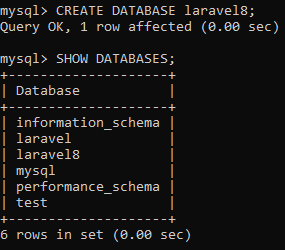
Joshua Budd Tradefull Take Home Evaluation Companion Document.

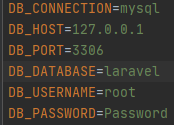
Hello! My name is Joshua Budd and this is a little document to walk you through what sort of program I have made for the Tradefull take home evaluation. The objective of this document is to try and explain some of the design and implementation decisions and how they work to whoever looks at my application. I created a simple application with the latest version of Laravel (currently Laravel 8 as of October 2020). The purpose of this application is to showcase CRUD operations within Laravel using a MySQL database and display user information to the user while allowing them to freely edit, update, add and delete users. Obviously, this isn’t how you would set up users in a non-test environment, but it is useful for demonstrating various structures and data manipulation within the Laravel framework.

Step 1. Create the database in which we will be storing the user information. By ensuring that the latest version of MySQL is installed, we can use a simple CREATE DATABASE Laravel command to create said database. Use the command SHOW DATABASES; to get a list of databases to make sure our added database is present.



Step 2. Now that the database is created on our machine, we need to create a new Laravel project to work in. We do this by downloading the latest version of Laravel from <https://laravel.com/> we will also need composer <https://getcomposer.org/> for dependencies and updating. Once Laravel and composer have been installed and properly configured as per their download page, we can create a new Laravel project from the command line with the command Laravel new ‘project name’. Once this is run, all of the Laravel dependencies and framework files will be downloaded and ready to start creating php MVC websites!

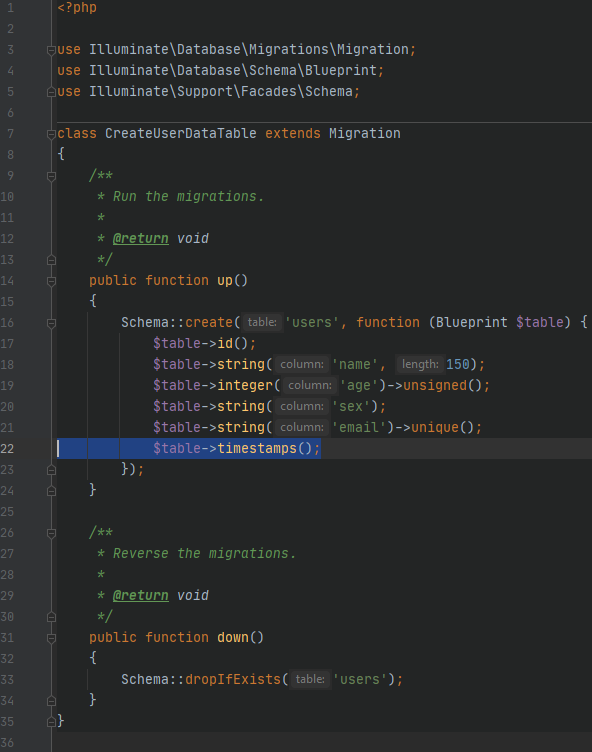
Step 3. Now that our Laravel project is set up, we are ready to start building the website. The first file that I created was the migration file so that we can set up the table/columns for our database. This migration file not only will create the table we tell it to in our database, but it can also rollback the table so we don’t have to manually keep dropping a table if we wish to re-add or change its structure. Very handy.

3a. First however, we need to properly connect the Laravel project to the database and make sure it is using the right credentials. Navigate to the .env file under the root directory. In here, we set up the DB\_CONNECTION, DB\_DATABASE, DB\_ USERNAME, and DB\_ PASSWORD variables to properly point and login to our recently created database. Mine looks like this: 

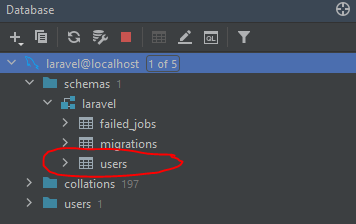
3b. Creating the migration comes next. We do this via the command line, so navigate to the root directory of your Laravel project and there, we will use artisan commands to create a file for our migration. The command to do this is php artisan make:migration create\_user\_data\_table ￼

3c. Navigate via PHPStorm to our create user data table file we just made. There, we will define the table’s columns and what datatypes they should accept. Columns that we will add include columns such as name and email.

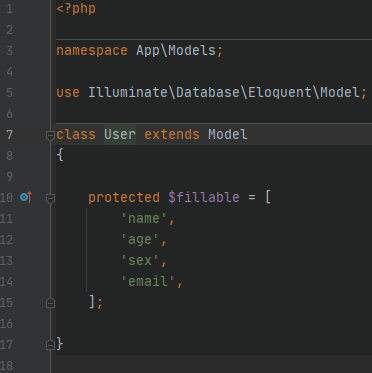
￼



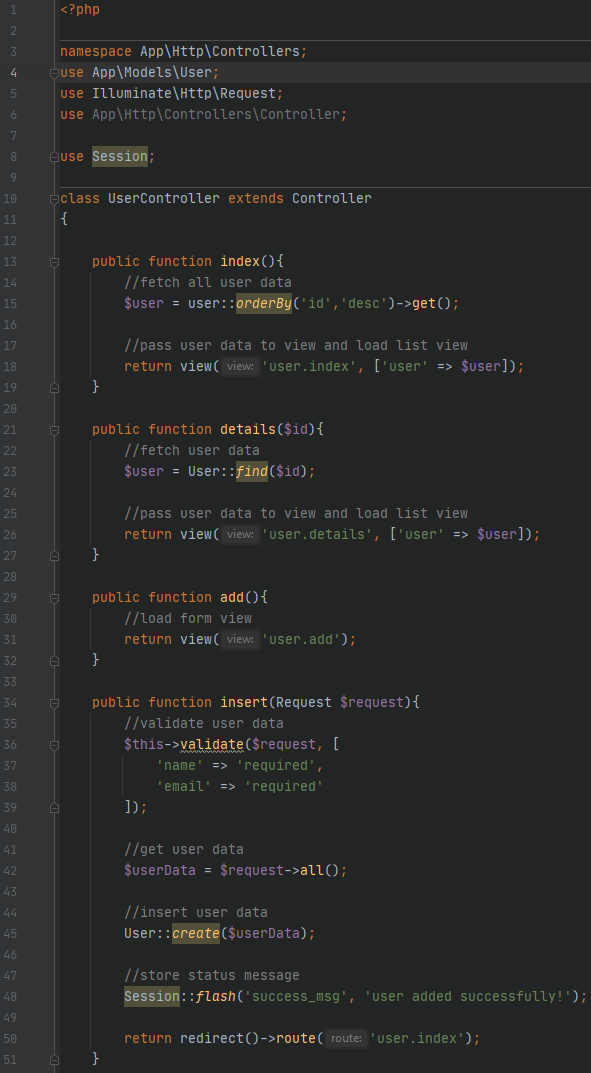
3d. Now that the migration is completed, we just need to run it via the command line and confirm that the table is created! The command to run a migration is php artisan migrate

We can see the created table in the database GUI view via PHPStorm!

4. We can now create the model for our MVC app. Create a new file under app/Http/Models folder called User.php. This class will extend the model base class and define attributes of our database and how to handle the data sent to it.

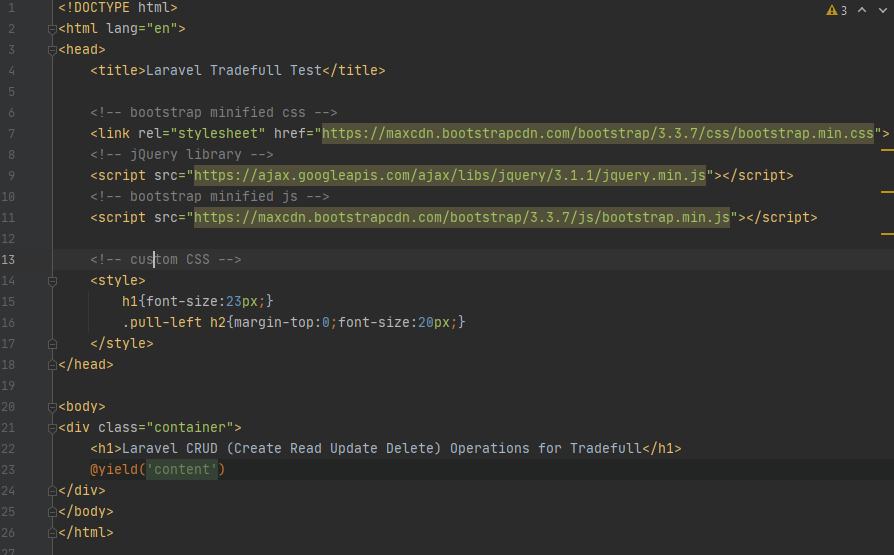


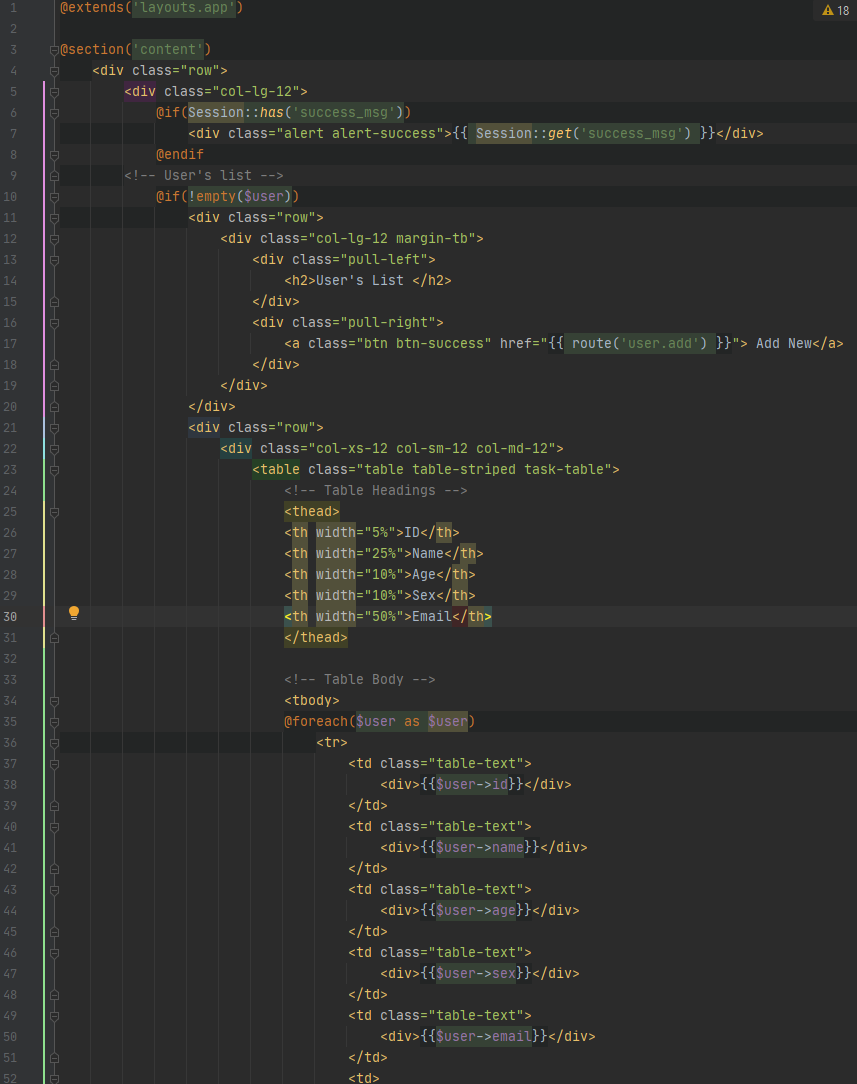
5. The next step we will take is to create the UserController.php controller in the folder app/Http/Controllers. This is the controller part of our MVC website that will control actions such as validation and re-routing. Here, we will define what to do with the action that the user is taking and re-direct them via routes to the proper place they want to be all while loading proper views.

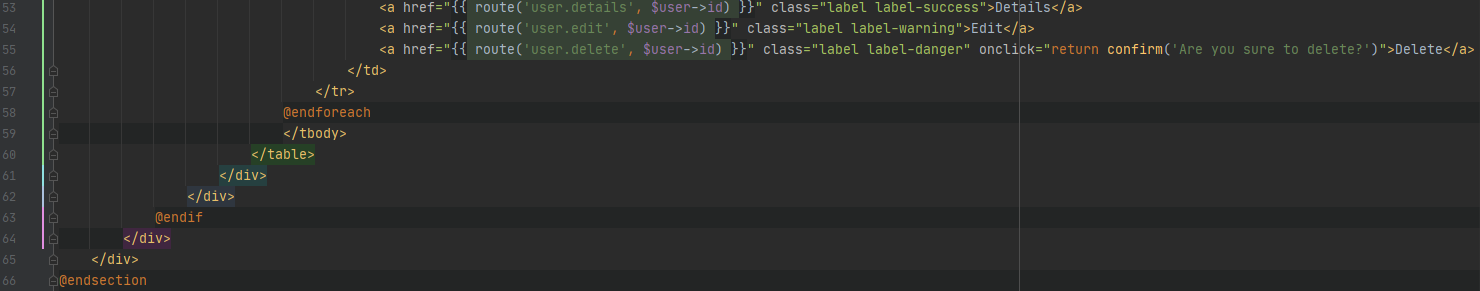


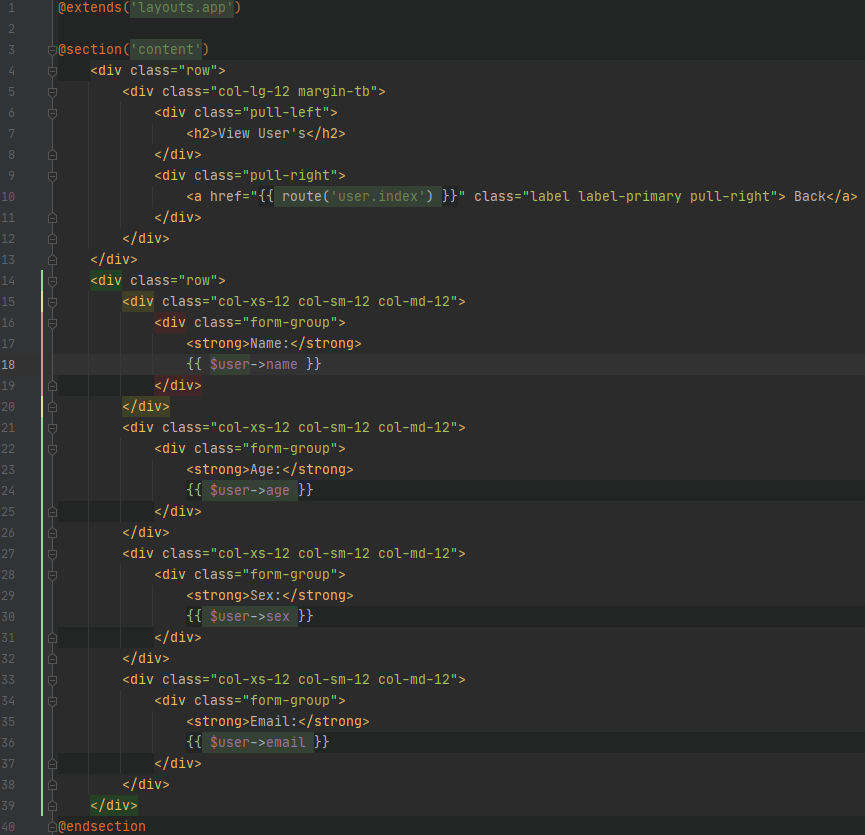
6. After we have created properly the longest file in our test application, we will define the routes for our website. This is done in the web.php file under the routes folder. Your routes file should look something like this: 

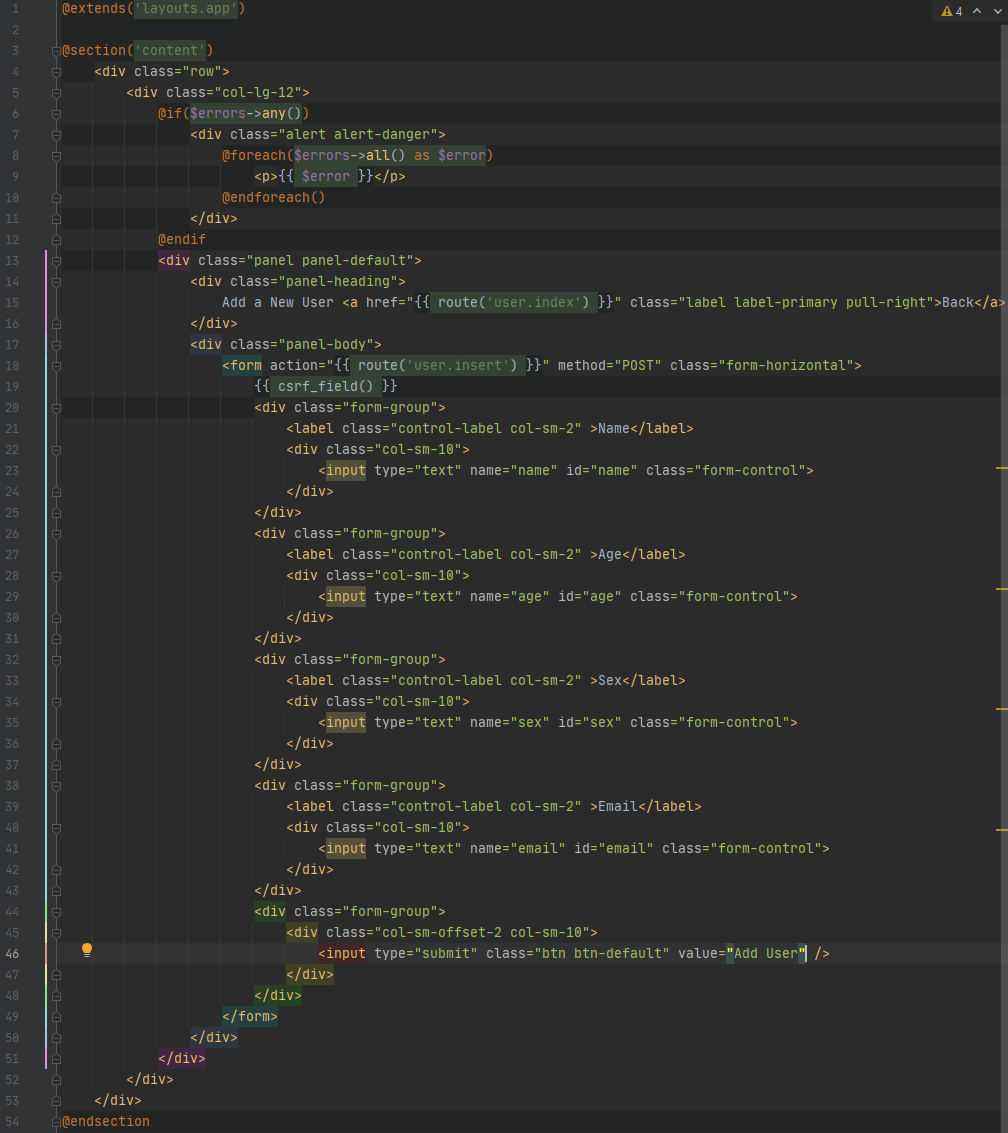
7. For our application, we will use a base layout for our view and CSS to style it to not look like too much of an eyesore. We can do this by defining minified css and js scripts that will tell the browsers where to pull its formatting from.

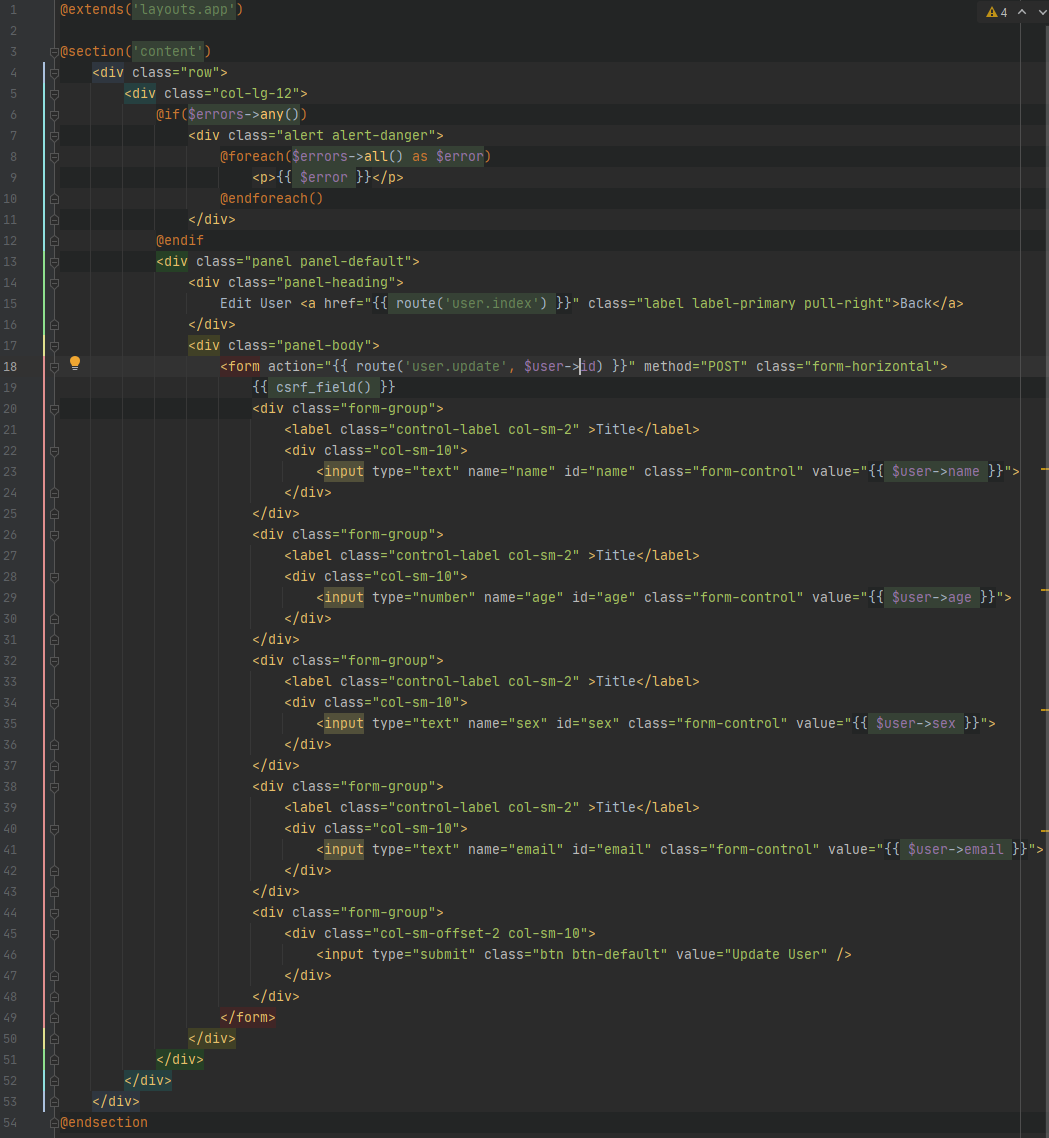
8. Now that we have the majority of our logical structures in place, we can start defining how our views will look in our MVC application. The first one we will make will be the index page. We name these files with the .blade.php format so we can take advantage of the blade templating engine.

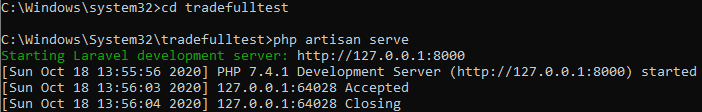


9. Now, we will create the view to display the details of a specific user.

10. Following the previous step, now we create the view for adding a new user to the database.

11. Lastly, we create the view for when the user wants to update an already existing user’s detail.

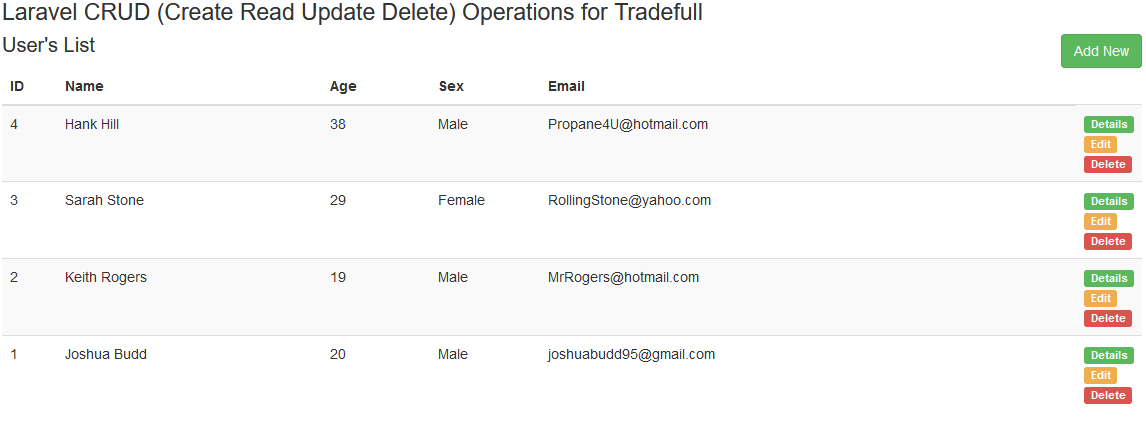
12. Artisan also has a function to locally host our Laravel apps for testing purposes. We can accomplish this my using the command line to run the following command from inside our Laravel root directory: php artisan serve

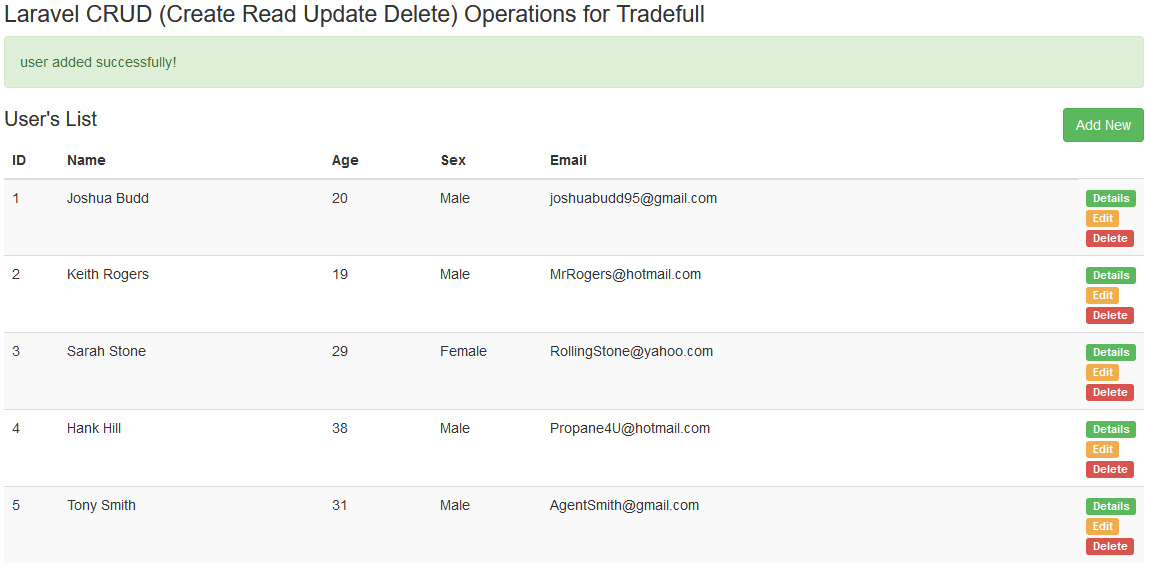
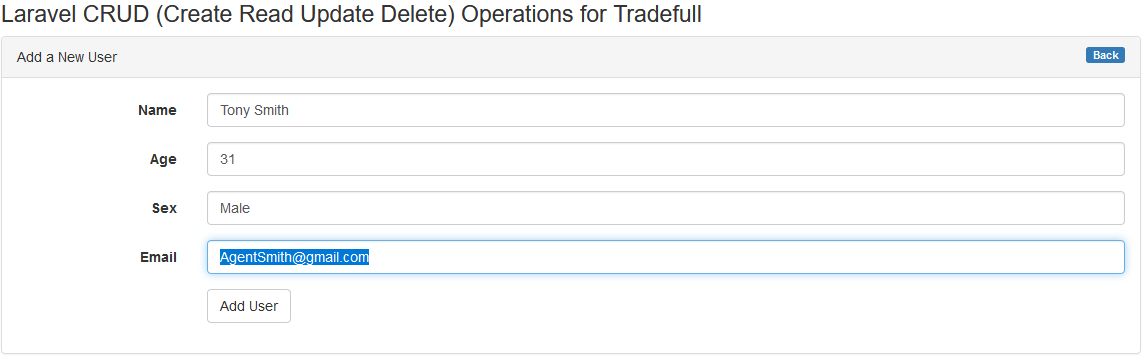


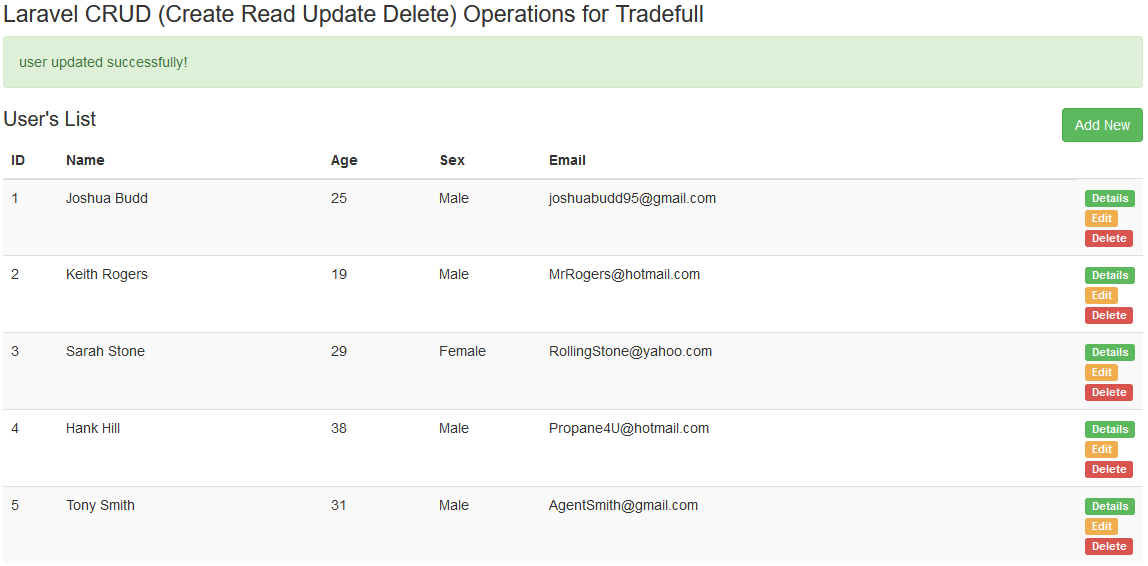
Now, if we visit the first route we configured in associate with the localhost we just set up, the url should look something like this:

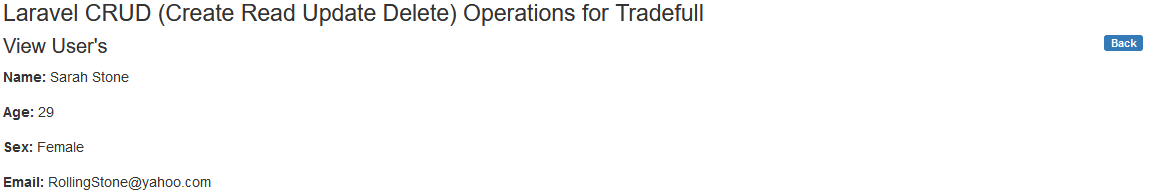
<http://127.0.0.1:8000/user>

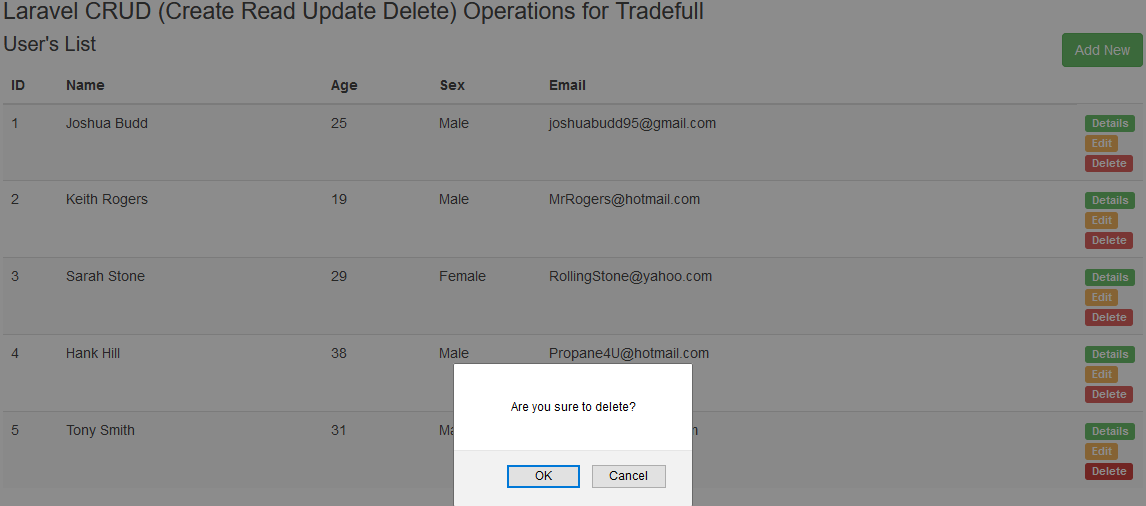
Once there, we see a view similar to this one (some users were added to help showcase how the information will look when rendered)

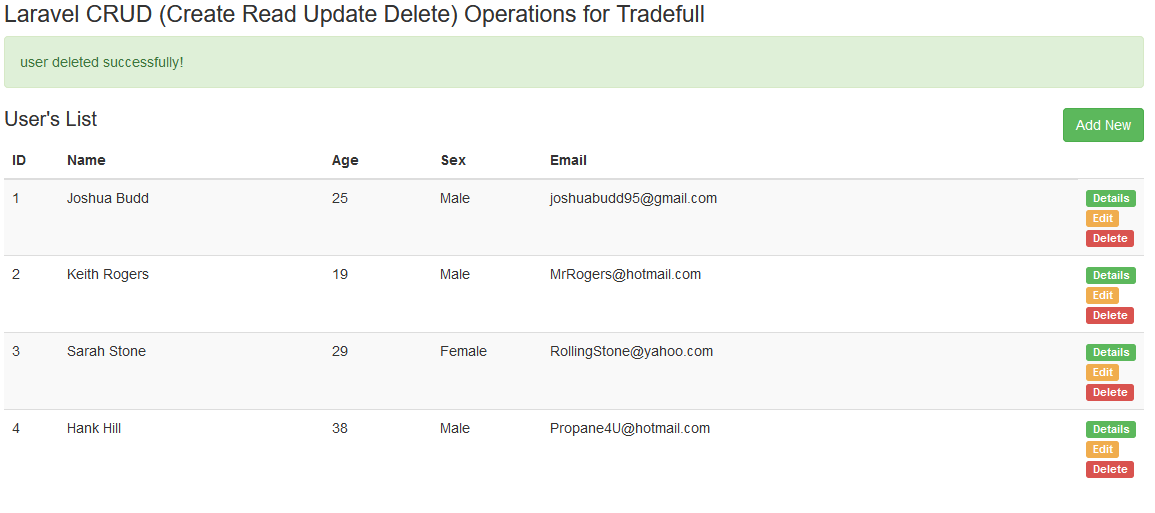
Great, no error’s and we can see that our databases’ information is being displayed correctly and neatly in ascending order. Now, let’s try adding a new user to the table.

Next, let’s see if we can edit an already present user in the table and save it correctly.

Great, the age was successfully updated for Joshua Budd to 25! Now, let’s see another view of a single user’s details via the green details button.

Looks good! Now, let’s see if we can successfully delete a user from the table without it breaking!



Perfect, we were able to delete Tony Smith out of our program.

All of the CRUD operations were able to be performed without the application throwing and errors and the information was properly reflected in the database. The PHP Laravel8 MVC application is working with no hiccups or problems! Now, this application can be improved on immensely with everything from styling to input validation, but this simple application showcases the essential CRUD operations that we set out to create in the first place. I hope you enjoyed my little tutorial and explanation of my CRUD Laravel application as much as I enjoyed making it!