Summary

This application was created using the Scala Play Framework. It uses an isolated Slick Database and Flyway. You can view this on github at the following location: https://github.com/eoincos/digitary-coding-challenge

This project was heavily influenced by the following 2 sample Play Framework projects:
https://developer.lightbend.com/start/?group=play&project=play-scala-forms-example
https://developer.lightbend.com/start/?group=play&project=play-scala-isolated-slick-example

The current functionality is the ability to view all users (Read) and to create new users (Create).

The routing has 2 options GET ”/” and POST “/”. The first route allows you to view all the users in the database and gives you a form to create new users. The second route allows you to post the form in to create a user.

The form is managed in a Twirl template called UserForm.

The front end is using a basic bootstrap CSS for styling.

Known Issues

No CRUD Rest API

The routes only use a GET and POST on the “/” path. To make this a good implementation of a CRUD Rest API I would recommend giving the ability to view specific users by creating a Router with the following options.

```
override def routes: Routes = {
  case GET(p"/") =>
    //view all users
  case POST(p"/") =>
    //create a new user
  case GET(p"/$id") =>
    //View a specific user
  case POST(p"/$id") =>
    //Updates a specific user
  case POST(p"/$id/delete") =>
    //Deletes a specific user
}
```

There is a sample REST API project in the Play Framework that I didn’t get a chance to integrate that would have been a great starting point. You can find it here:

https://developer.lightbend.com/start/?group=play&project=play-scala-rest-api-example

Verifying inputs

In the UserForm object there could be verifying functions to check in more detail the content that is being sent. This could be functionality such as checking the telephone numbers only contain numbers using regular expressions.

Update and Delete not implemented

This was the last piece of functionality I was working on before I had to stop working on this assignment. The issue was trying to debug an issue with returning recursive Futures in the Action.async. Have a look at the code below that was not committed to GitHub:

```
userDAO.lookup(user.id).map { result =>
   if(result.isEmpty) {
     //if it doesn't exist, create it
     userDAO.create(user).map { id =>
       Redirect(routes.HomeController.index())
     }
   } else {
     //else it does exist, update it
     userDAO.update(user).map { id =>
       Redirect(routes.HomeController.index())
     }
   }
 }
```

This code gives an error saying it's only expecting one Future instead of 2 Futures.

```
 found   : controllers.UserForm.Data =>
scala.concurrent.Future[scala.concurrent.Future[play.api.mvc.Resul
t]]
 required: controllers.UserForm.Data =>
scala.concurrent.Future[play.api.mvc.Result]]
```

Lessons Learned

Learning the Play Framework

The learning curve on Play Framework is much longer than I was prepared for. Getting the sample projects up and running is straightforward however debugging issues and getting to grasps with the conventions is difficult. The two biggest issues were getting used to Twirl templates and the conventions to write the Action.async in the Controller.

This could be because of the new changes in 2.6.x. There isn’t much material online yet such as tutorials or community questions.

Learning the Slick Database

Getting this database up and running with the project was difficult. There is a steep learning curve with this database. Creating in isolation helped but it was very tough. It does seem to be the standard database to use for Scala Play Framework projects.