Software Architectures

Play Framework

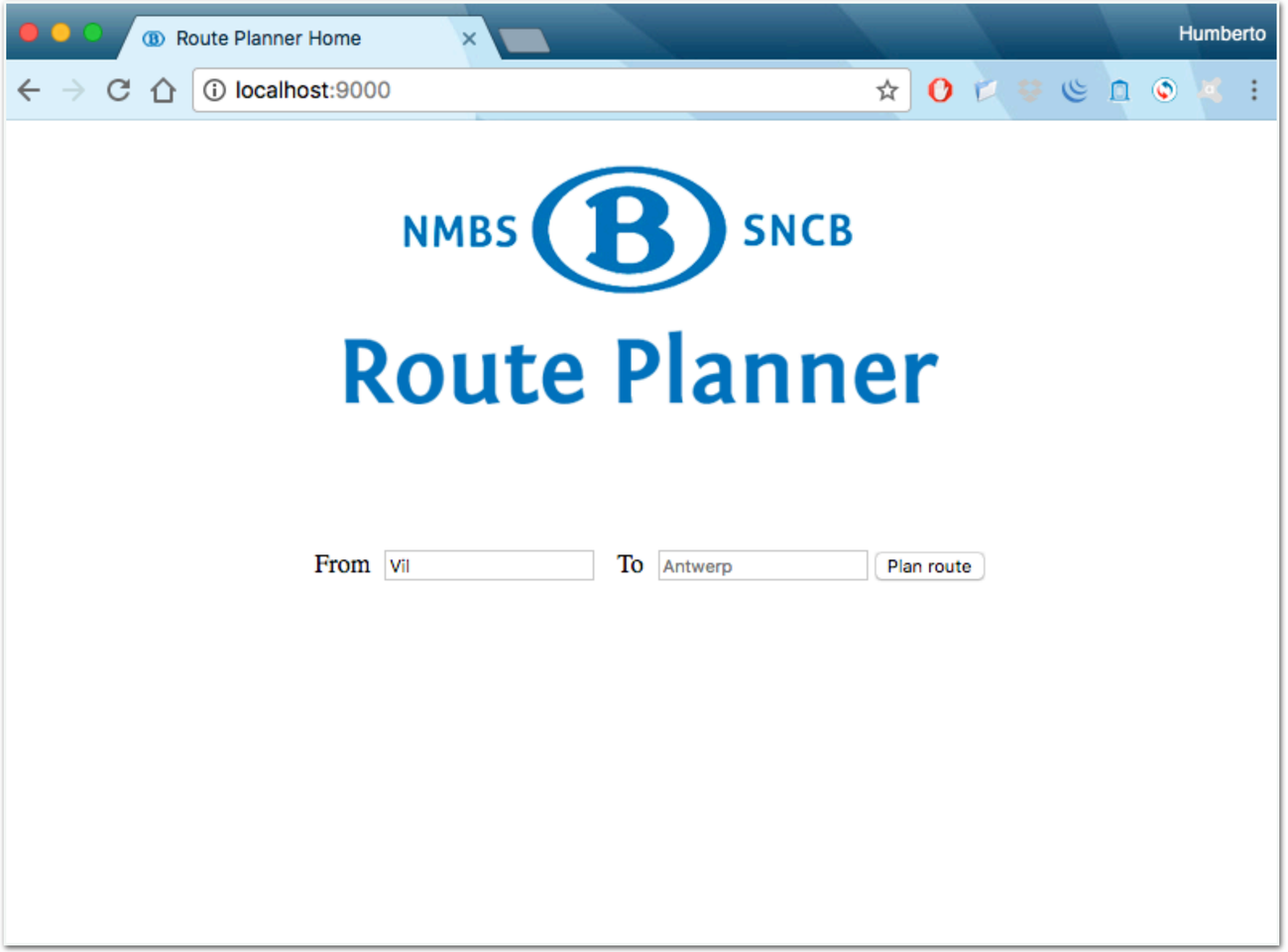
Nov 22, 2018

Assistants: Humberto Rodríguez Avila, Kennedy Kambona
Email: {rhumbert, kkambona}@vub.be

VRIJE UNIVERSITEIT BRUSSEL

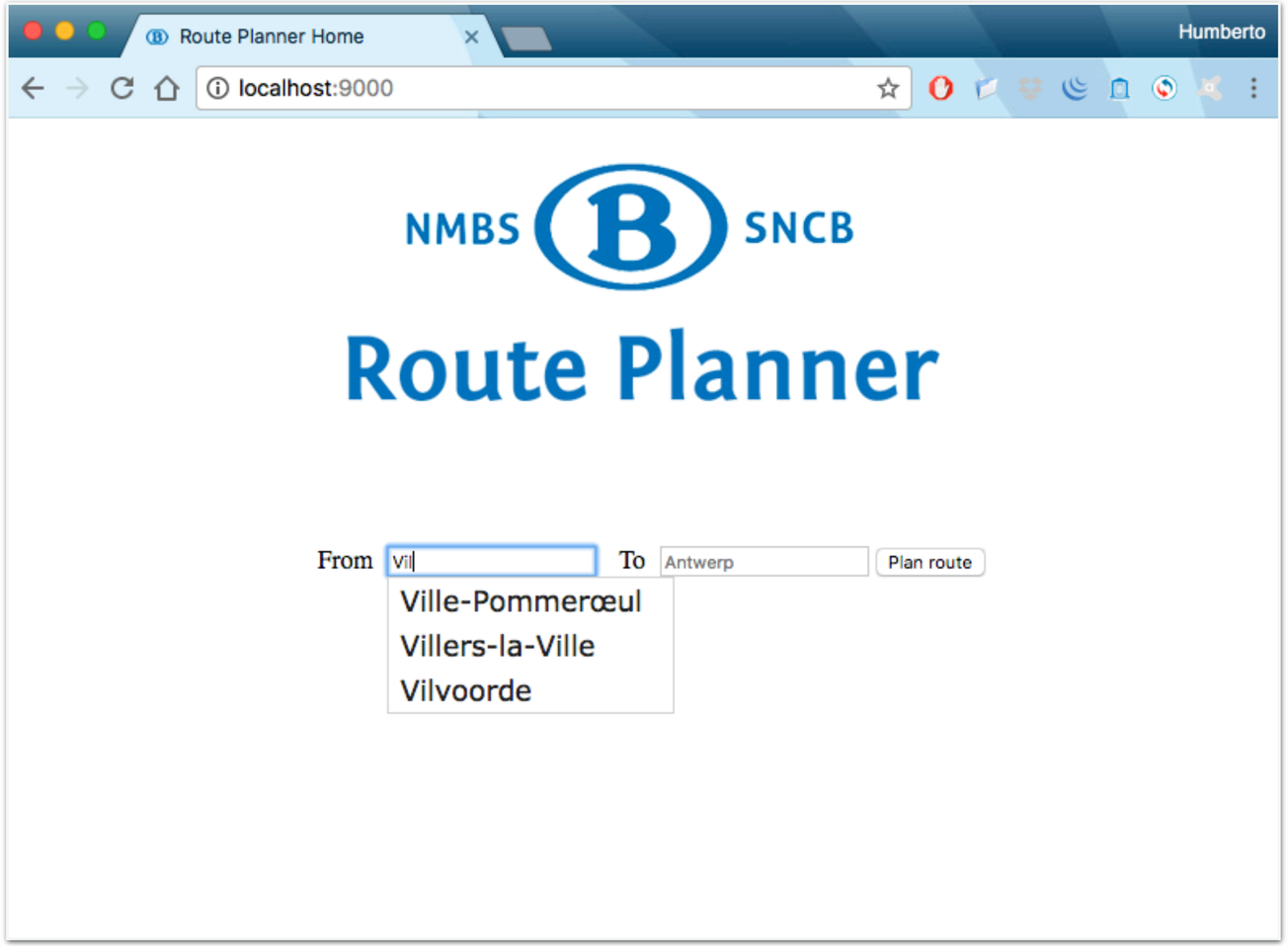
Software Languages.Lab

NMBS-SNCB Route Planner Application



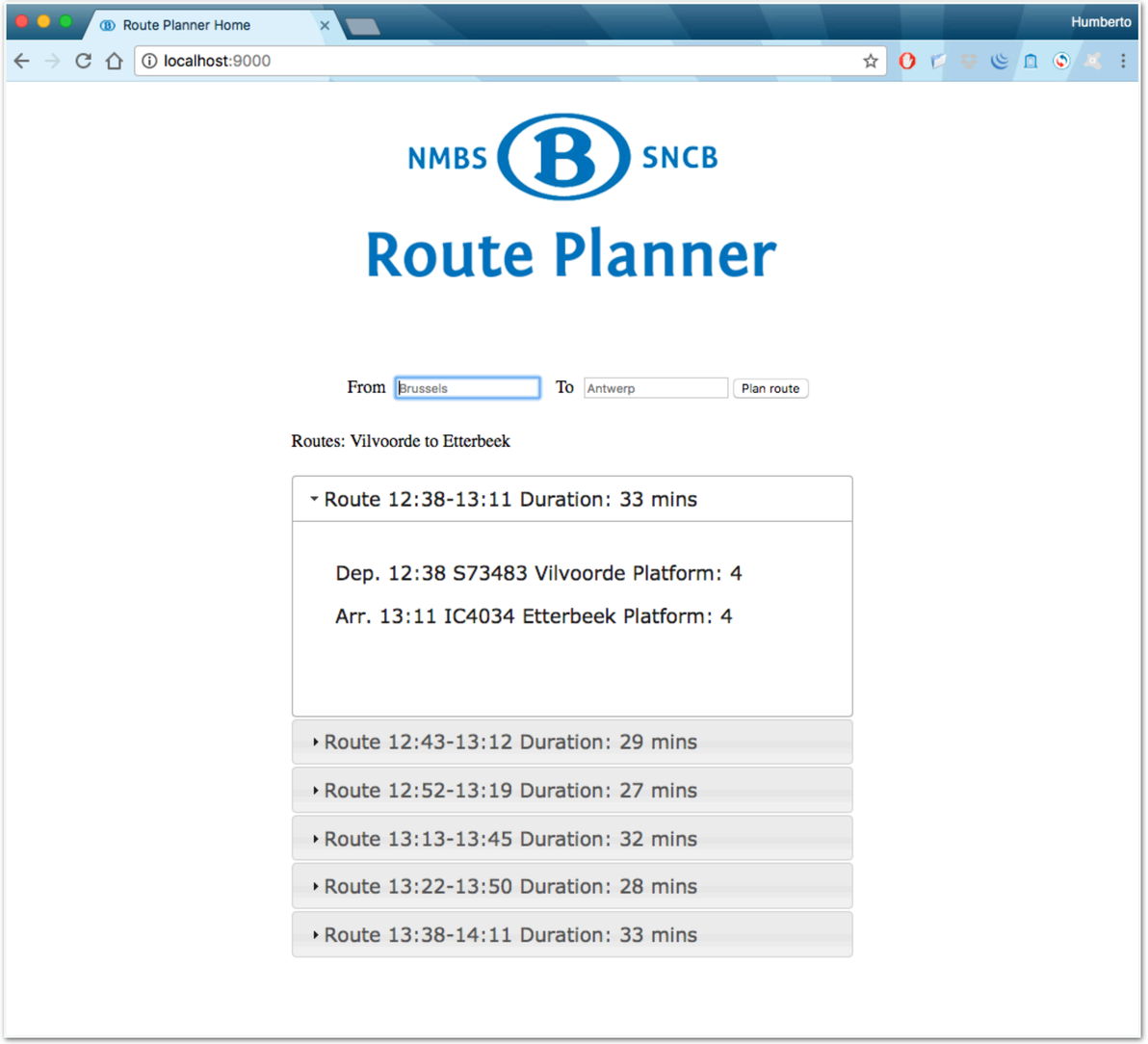
Route Planning

NMBS-SNCB Route Planner Application



Autocomplete

NMBS-SNCB Route Planner Application



Routes info

Create a Play Framework project

1. Open a terminal in your Scala Workspace

```
$ sbt new playframework/play-scala-seed.g8
```

Create new project from a template
Fill only the name of your new project

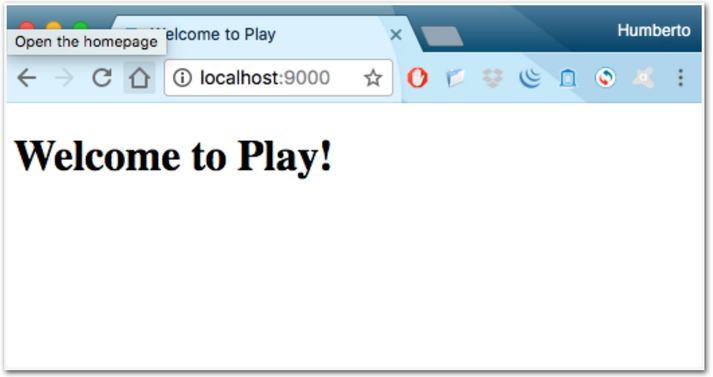
2. Run your project

```
$ cd route-planner
$ sbt run
```

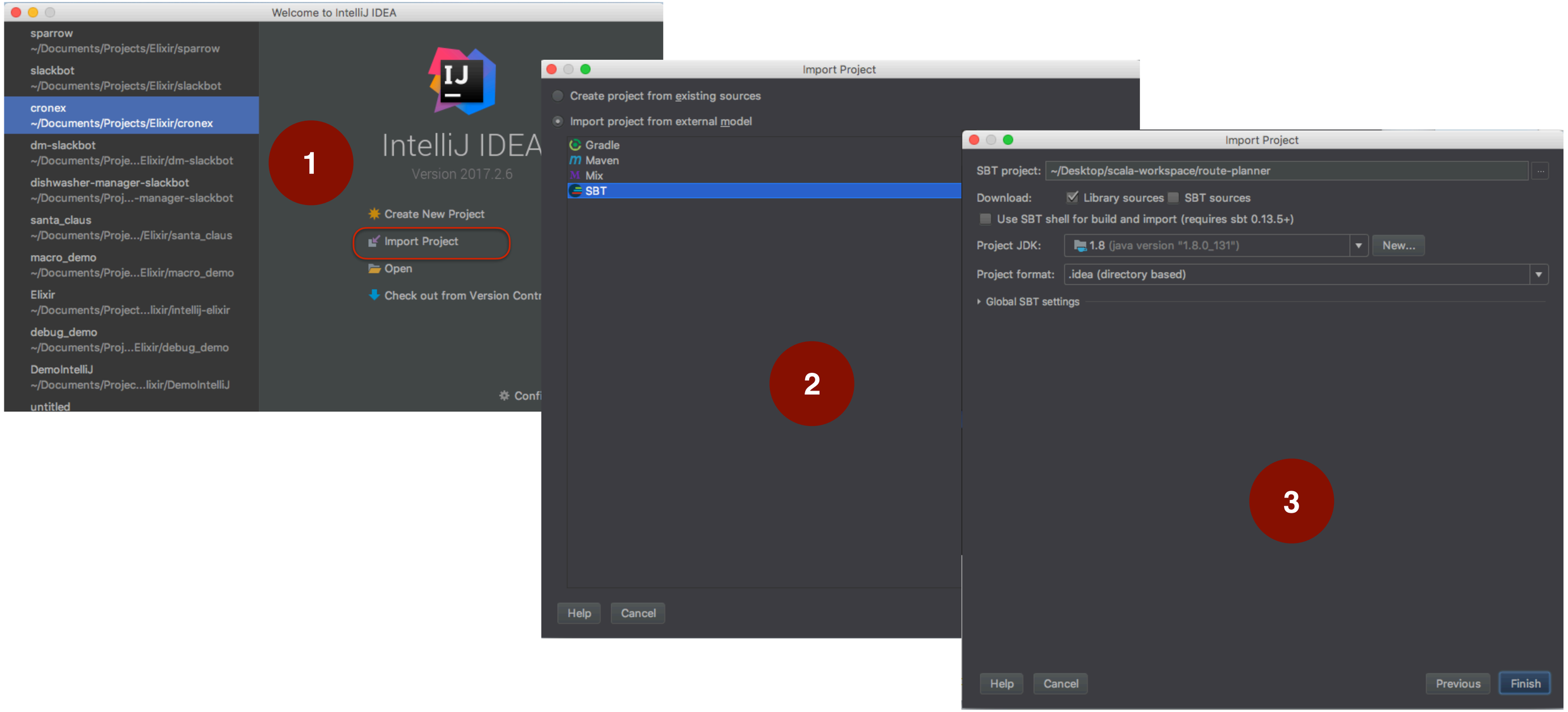
wait…….

3. Test your project

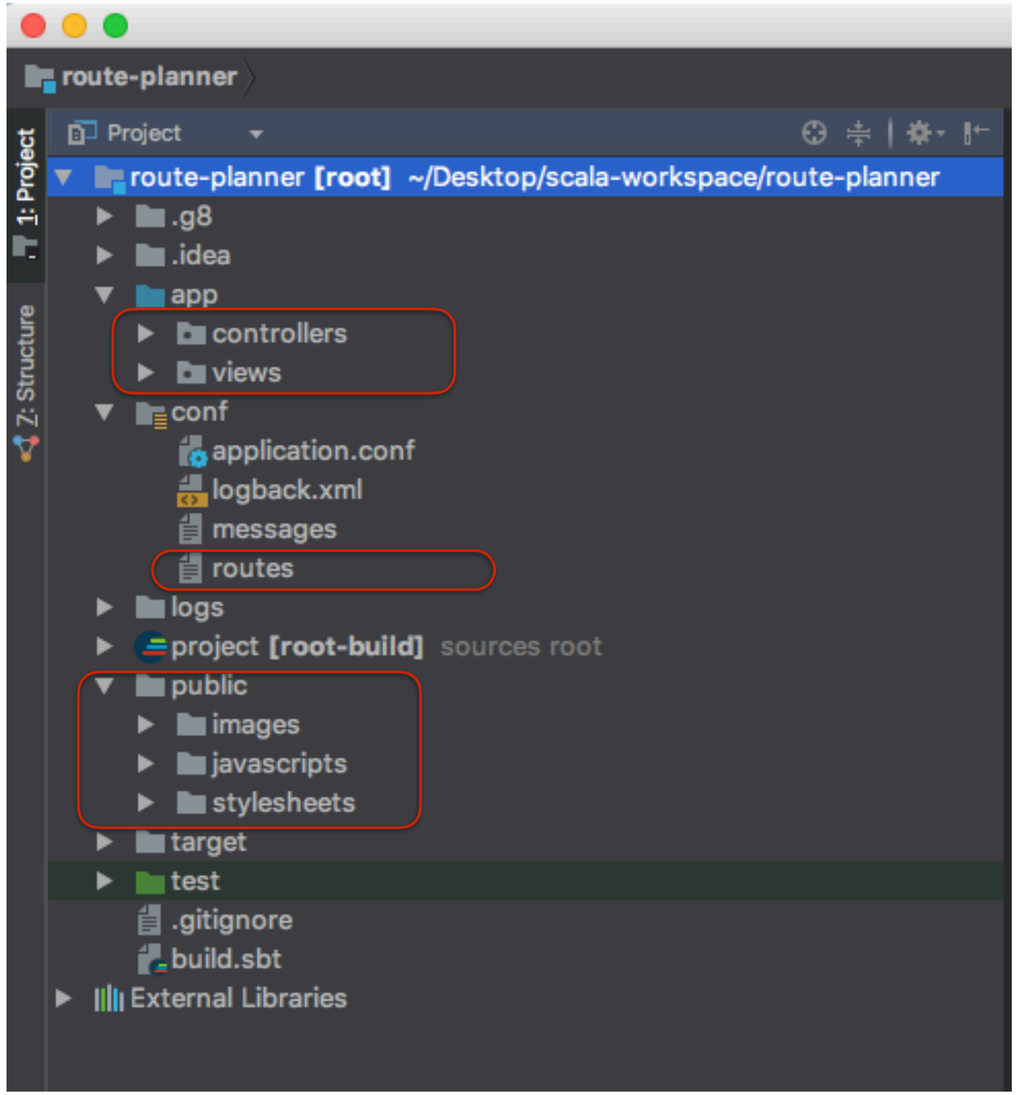
http://localhost:9000/



Import project to IntelliJ



Play Project Structure



Default Files: `main.scala.html`

main.scala.html

```
@*
 * This template is called from the `index` template. This template
 * handles the rendering of the page header and body tags. It takes
 * two arguments, a `String` for the title of the page and an `Html`
 * object to insert into the body of the page.
 *@
@(title: String)(content: Html)

<!DOCTYPE html>
<html lang="en">
    <head>
        @* Here's where we render the page title `String`. *@
        <title>@title</title>
        <link rel="stylesheet" media="screen" href="@routes.Assets.versioned("stylesheets/main.css")">
        <link rel="shortcut icon" type="image/png" href="@routes.Assets.versioned("images/favicon.png")">

    </head>
    <body>
        @* And here's where we render the `Html` object containing
         * the page content. *@
        @content

      <script src="@routes.Assets.versioned("javascripts/main.js")" type="text/javascript"></script>
    </body>
</html>
```

Default Files: `index.scala.html`

index.scala.html

```
@()

@main("Welcome to Play") {
  <h1>Welcome to Play!</h1>
}
```

Default Files: `routes`

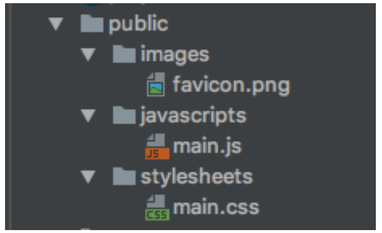
space/route-planner/conf/routes

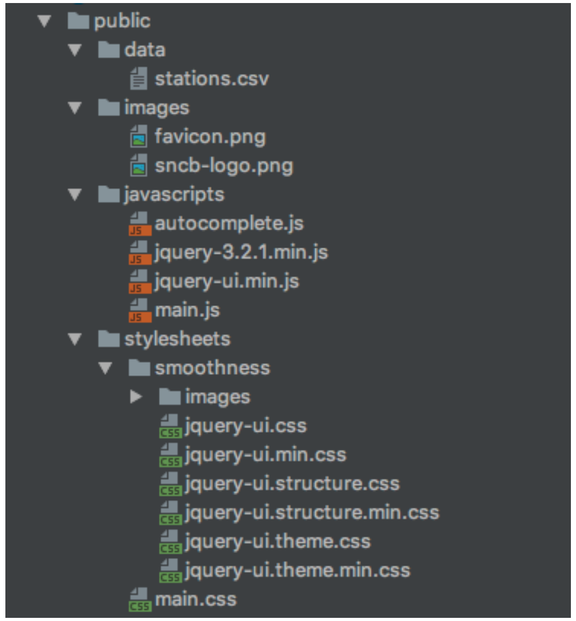
```
# Routes
# This file defines all application routes (Higher priority routes first)
# https://www.playframework.com/documentation/latest/ScalaRouting
# ~~~~

# An example controller showing a sample home page
GET     /                           controllers.HomeController.index

# Map static resources from the /public folder to the /assets URL path
GET     /assets/*file               controllers.Assets.versioned(path="/public", file: Asset)

```

Building Route Planner App

1. Download https://goo.gl/Y9r59h

2. Replace the default **public** directory





Create a RouteForm Object

RouteForm.scala

```scala
package controllers

object RouteForm {

  import play.api.data.Forms._
  import play.api.data.Form

  /**
    * A form processing DTO that maps to the form below.
    *
    * Using a class specifically for form binding reduces the chances
    * of a parameter tampering attack and makes code clearer.
    */
  case class Data(from: String, to: String)

  /**
    * The form definition for the "create a route" form.
    * It specifies the form fields and their types,
    * as well as how to convert from a RouteData to form data and vice versa.
    */
  val form = Form(
    mapping(
      "From" -> nonEmptyText,
      "To" -> nonEmptyText
    )(Data.apply)(Data.unapply)
  )



}
```

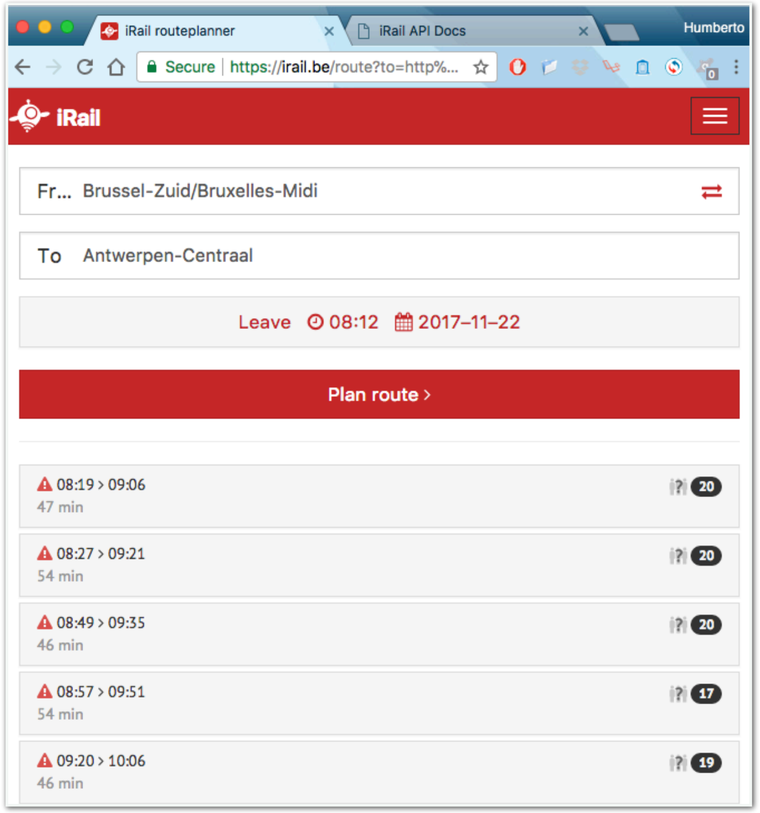
Right click the **`controllers`** package, and New->Scala Class

Update `main.scala.html`

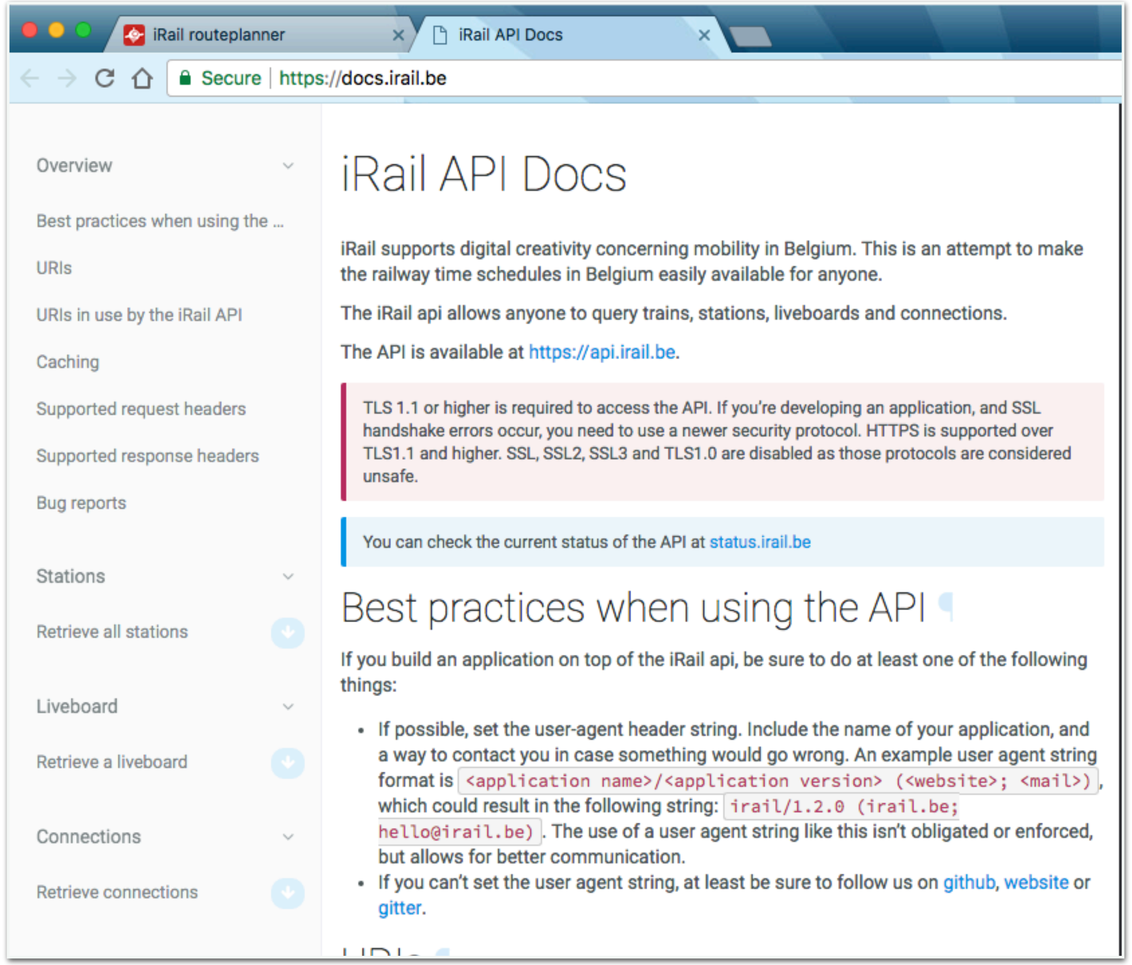
main.scala.html

```
@(title: String)(content: Html)

<!DOCTYPE html>
<html lang="en">
    <head>
        <title>@title</title>
        <link rel="stylesheet" media="screen" href='@routes.Assets.versioned("stylesheets/main.css")'>
        <link rel="stylesheet" media="screen" href='@routes.Assets.versioned("stylesheets/smoothness/jquery-ui.min.css")'>
        <link rel="shortcut icon" type="image/png" href='@routes.Assets.versioned("images/favicon.png")'>
    </head>
    <body>
        @content

        <script src='@routes.Assets.versioned("javascripts/jquery-3.2.1.min.js")' type="text/javascript"></script>
        <script src='@routes.Assets.versioned("javascripts/jquery-ui.min.js")' type="text/javascript"></script>
        <script type="text/javascript" src="@routes.Application.javascriptRoutes"></script>

        <script src='@routes.Assets.versioned("javascripts/autocomplete.js")' type="text/javascript"></script>
        <script src='@routes.Assets.versioned("javascripts/main.js")' type="text/javascript"></script>
    </body>
</html>
```

iRails API



https://irail.be/



https://docs.irail.be/

iRails API

GET Retrieve connections

`/connections/{?from,to,date,time,timesel,format,lang,fast,typeOfTransport,alerts,results}`

URI Parameters Hide

from `string` (required) Example: Gent-Sint-Pieters
The name or id of the station of departure.

to `string` (required) Example: Mechelen
The name or id of the destination.

timesel `string` (optional) **Default:** departure Example: departure
Whether the results should show arrivals departures in the station.

Choices: `departure` `arrival`

typeOfTransport `string` (optional) **Default:** trains Example: trains
The types of transport to include in the search

Choices: `trains` `nointernationaltrains` `all`

alerts `boolean` (optional) Example: false
(Deprecated) Wether or not to include alerts about a route in the response. This could be railworks etc, announced on the NMBS website. **Note: alerts are now always included**

results `number` (optional) Example: 6
(Deprecated) The number of results to return. Default value is 6. This might be used as a guideline for the server, but there is no guarantee the server will return this exact amount of results!

time `string` (optional) **Default:** current time in Belgium Example: 1230
The time to query.

The time is formatted as hhmm.

date `string` (optional) **Default:** current date in Belgium Example: 300917
The date to query.

The date is formatted as ddmmyy.

fast `boolean` (optional) **Default:** false Example: false
(Deprecated) Whether or not you want stations in the result to

GET /connections/?from=Gent-Sint-Pieters&to=Mechelen&date=300917&time=1230&timesel=departure&format=json&lang=en&fast=false&typeOfTransport=trains&alerts=false&results=6

Requests Json XML

Headers

```
Accept: application/json
```

Responses 200 400 404 500

Headers

```
Content-Type: application/json
Access-Control-Allow-Origin: "*"
cache-control: Public
etag: "e5e7c8ae25bb71cdfce80412c2b1be54"
```

Body


```
{
  "version": "1.1",
  "timestamp": 1489622781,
  "connection": [
    {
      "id": 0,
      "departure": {
        "delay": 0,
        "station": "Antwerp-Central",
        "stationinfo": {
          "id": "BE.NMBS.008821006",
          "@id": "http://irail.be/stations/NMBS/008821006",
          "locationX": 4.421101,
          "locationY": 51.2172,
          "standardname": "Antwerpen-Centraal",
          "name": "Antwerp-Central"
        },
        "time": 1497783600,
        "vehicle": "BE.NMBS.IC3033",
```

Create a JsonModel Object

JsonModel.scala

```scala
package model

object JsonModel {

  case class Connection(id: String, departure: Departure, arrival: Arrival, duration: String)
  case class Departure(station: String, time: String, vehicle: String, platform: String  )
  case class Arrival(station: String, time: String, vehicle: String, platform: String)


}
```

Right click `App` folder to create package “model”, and then Right click over it to create a **New** >**Scala Class**

Update `index.scala.html`

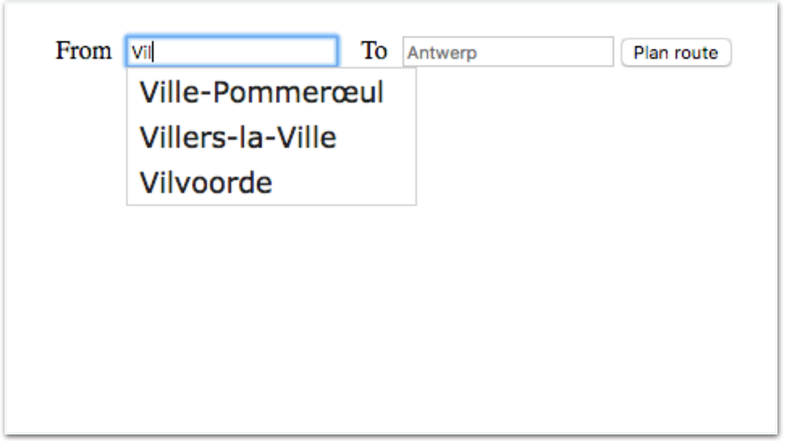
index.scala.html

```
@(routeForm: Form[RouteForm.Data], connections: Seq[model.JsonModel.Connection], postURL: Call)(implicit request: MessagesRequestHeader)

@main(title = "Route Planner Home") {

    <img id="sncb-logo" src='@routes.Assets.versioned("images/sncb-logo.png")' alt="Mountain View">

        @helper.form(postURL) {
            <div id="form">
                @helper.CSRF.formField
                <!--<label for="from">From</label>-->
                @helper.inputText(routeForm("From"), 'id -> "from", 'placeholder -> "Brussels", 'autofocus ->"")
                <!--<input type="text" id="from" placeholder="Brussels" autofocus=""/>-->
                <!--<label for="to">To</label>-->
                @helper.inputText(routeForm("To"), 'id -> "to", 'placeholder -> "Antwerp")
                <!--<input type="text" id="to" placeholder="Antwerp",/>-->
                <button> Plan route</button>
            </div>
        }
}
```

Retrieving Stations Name with Ajax

autocomplete.js

```
$("#to, #from").autocomplete({
    source: function(request, response){
        $.ajax(jsRoutes.controllers.RoutePlanner.stations(request.term))
            .done(function(data){
                response($.map(data, function(item){
                    return { label: item, value: item}
                }))
            })
            .fail(function(error){console.log(error)});
    },
    minLength: 2
});
```



```
/**...*/
@Singleton
class RoutePlanner  @Inject()(cc: MessagesControllerComponents, ws: WSClient) extends MessagesAbstractController(cc)  {


  var stationsFile = Environment.simple().getFile("/public/data/stations.csv")
  var stationsList = List[String]()
  private val connections = ArrayBuffer[Connection]()


  // URL to the method to calculate the connections.  You can call this directly from the template, but it
  // can be more convenient to leave the template completely stateless
  private val postURL = routes.RoutePlanner.calculateConnections()


  def index() = Action { implicit request: MessagesRequest[AnyContent] =>
    stationsList = io.Source.fromFile(stationsFile).getLines.toList
    Ok(views.html.index(form, connections, postURL))
  }

  // This will be the action that handles our ajax request
  def stations(term: String) = Action { implicit request: Request[AnyContent] =>
    val suggestions = stationsList.filter(_.startsWith(term))
    Ok(Json.toJson(suggestions))

  }

  // This will be the action that handles our form post
  def calculateConnections() = Action { implicit request: MessagesRequest[AnyContent] =>
    //"TODO: to implement for assignment 3"
    Ok(views.html.index(form, connections, postURL))
  }

}
```

Create an Application Controller

Application.scala

```scala
package controllers

import javax.inject.Inject
import play.api.mvc._
import play.api.routing._

class Application @Inject()(components: ControllerComponents) extends AbstractController(components) {

  def javascriptRoutes = Action { implicit request =>
    Ok(
      JavaScriptReverseRouter("jsRoutes")(
        routes.javascript.RoutePlanner.stations,
      )
    ).as("text/javascript")
  }

}
```

Update `index.scala.html`

index.scala.html

```
@(routeForm: Form[RouteForm.Data], connections: Seq[model.JsonModel.Connection], postURL: Call)(implicit request: MessagesRequestHeader)

@main(title = "Route Planner Home") {

    <img id="sncb-logo" src='@routes.Assets.versioned("images/sncb-logo.png")' alt="Mountain View">

        @helper.form(postURL) {
            <div id="form">
                @helper.CSRF.formField
                <!--<label for="from">From</label>-->
                @helper.inputText(routeForm("From"), 'id -> "from", 'placeholder -> "Brussels", 'autofocus ->"")
                <!--<input type="text" id="from" placeholder="Brussels" autofocus=""/>-->
                <!--<label for="to">To</label>-->
                @helper.inputText(routeForm("To"), 'id -> "to", 'placeholder -> "Antwerp")
                <!--<input type="text" id="to" placeholder="Antwerp",/>-->
                <button> Plan route</button>
            </div>
        }
}
```

```
    <div id="accordion">
        @for(c <- connections) {
        <h4>Route  @c.departure.time-@c.arrival.time  Duration: @c.duration mins </h4>
        <div>
            <p>Dep. @c.departure.time @c.departure.vehicle  @c.departure.station  Platform: @c.departure.platform </p>
            <p>Arr. @c.arrival.time @c.arrival.vehicle  @c.arrival.station  Platform: @c.arrival.platform </p>
        </div>
        }
    </div>
```

Software Architectures

Play Framework

Nov 22, 2018

Assistants: Humberto Rodríguez Avila, Kennedy Kambona
Email: {rhumbert, kkambona}@vub.be

VRIJE UNIVERSITEIT BRUSSEL

Software Languages.Lab