

Software Architectures

Play Framework

Nov 22, 2018

Assistants: Humberto Rodríguez Avila, Kennedy Kambona

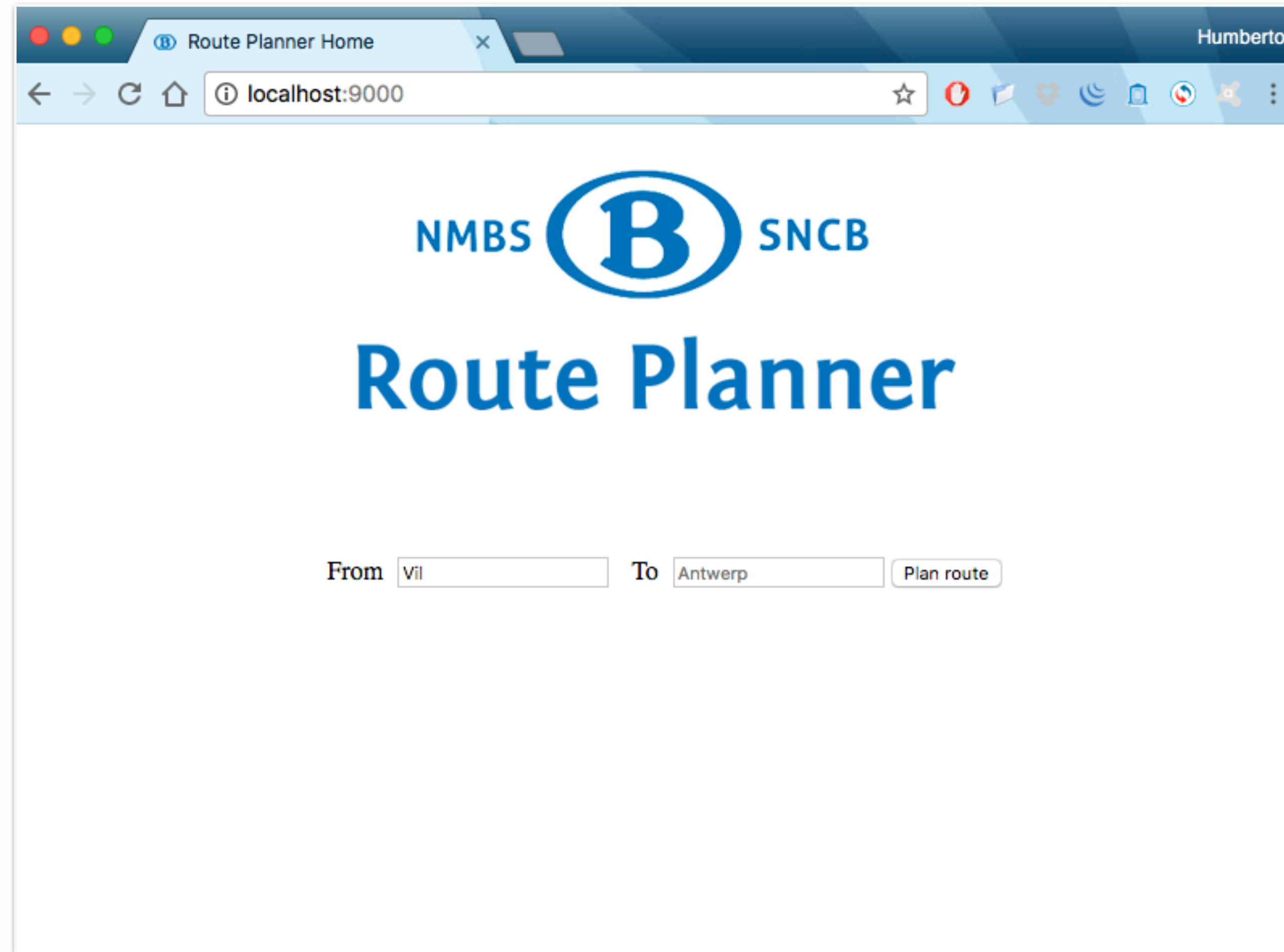
Email: {rhumbert, kkambona}@vub.be



VRIJE
UNIVERSITEIT
BRUSSEL

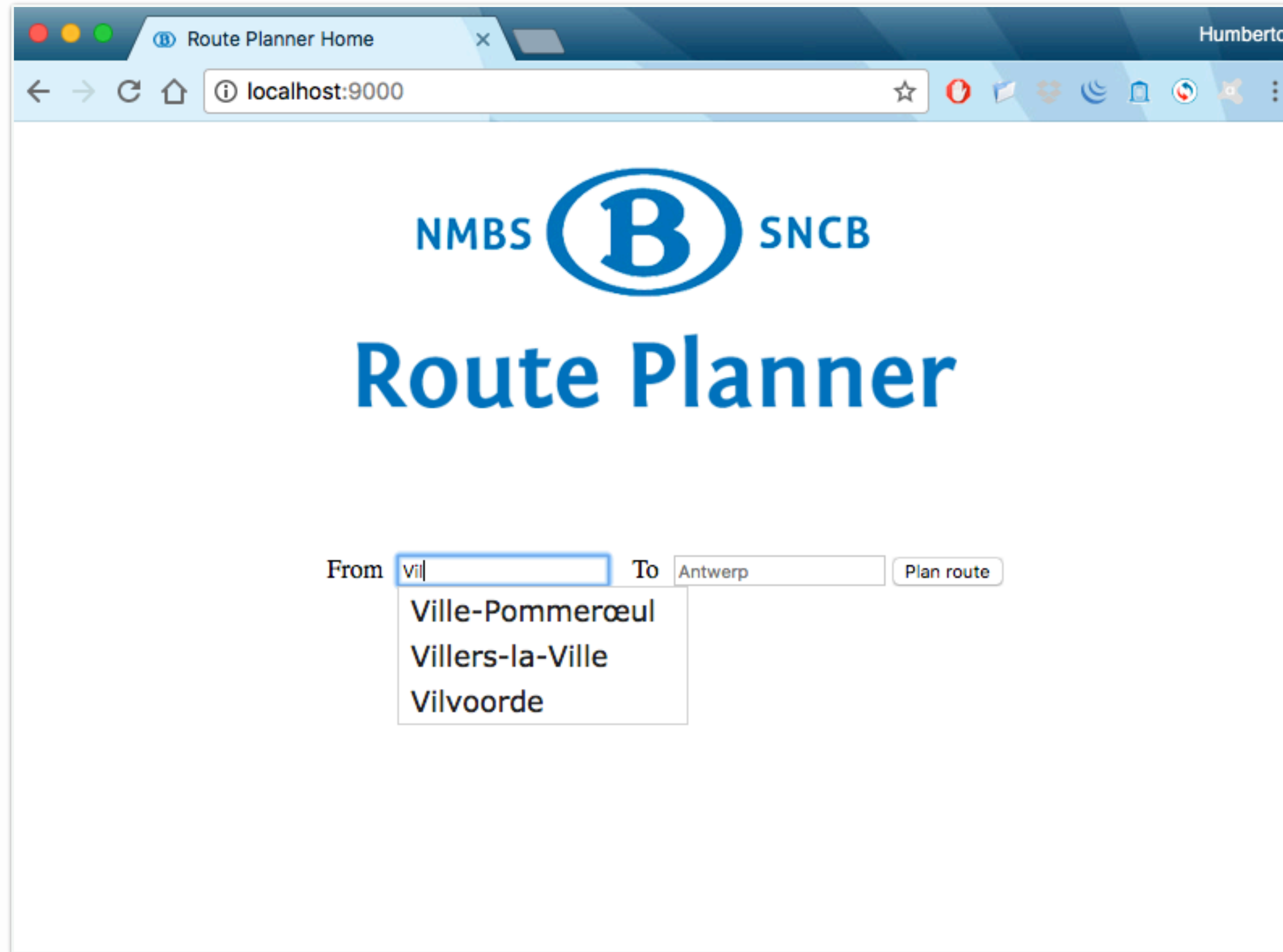


NMBS-SNCB Route Planner Application




Route Planning

NMBS-SNCB Route Planner Application



The screenshot shows a web browser window with the title "Route Planner Home" and a user profile "Humberto". The address bar shows "localhost:9000". The main content area features the NMBS-SNCB logo and the text "Route Planner". Below this, there is a form with two input fields: "From" and "To". The "From" field contains the text "vill" and has a dropdown menu open showing three suggestions: "Ville-Pommerœul", "Villers-la-Ville", and "Vilvoorde". The "To" field contains the text "Antwerp". A "Plan route" button is located to the right of the "To" field.

NMBS  SNCB

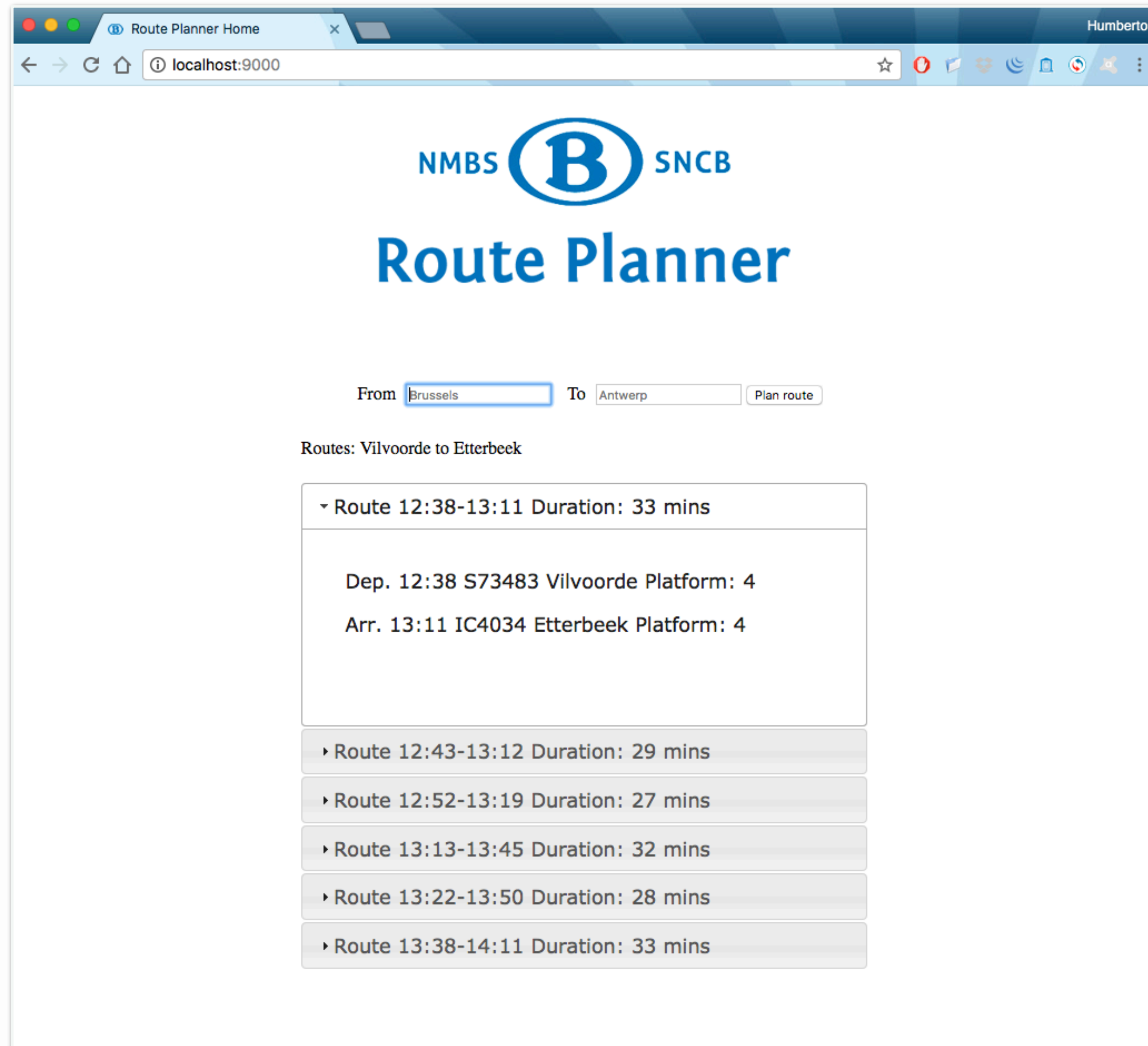
Route Planner

From To [Plan route](#)

- Ville-Pommerœul
- Villers-la-Ville
- Vilvoorde

Autocomplete

NMBS-SNCB Route Planner Application



The screenshot shows a web browser window with the title "Route Planner Home" and a user profile "Humberto". The address bar shows "localhost:9000". The main content area features the NMBS-SNCB logo and the title "Route Planner". Below this, there is a search form with "From" and "To" fields. The "From" field contains "Brussels" and the "To" field contains "Antwerp". A "Plan route" button is next to the "To" field. Below the search form, the text "Routes: Vilvoorde to Etterbeek" is displayed. A list of routes is shown, with the first route expanded to show details.

From To Plan route

Routes: Vilvoorde to Etterbeek

- ▼ Route 12:38-13:11 Duration: 33 mins
 - Dep. 12:38 S73483 Vilvoorde Platform: 4
 - Arr. 13:11 IC4034 Etterbeek Platform: 4
- Route 12:43-13:12 Duration: 29 mins
- Route 12:52-13:19 Duration: 27 mins
- Route 13:13-13:45 Duration: 32 mins
- Route 13:22-13:50 Duration: 28 mins
- Route 13:38-14:11 Duration: 33 mins

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

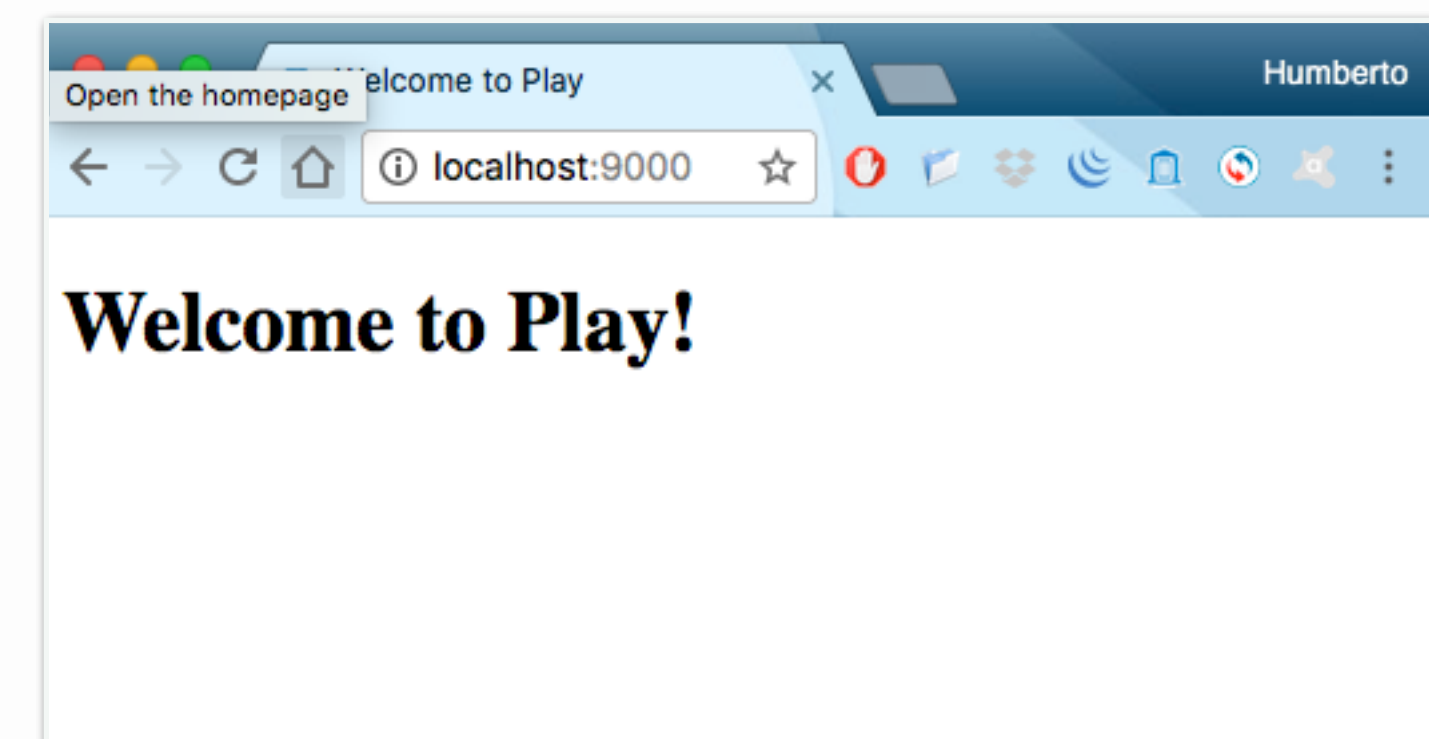
```
$ cd route-planner
```

```
$ sbt run
```

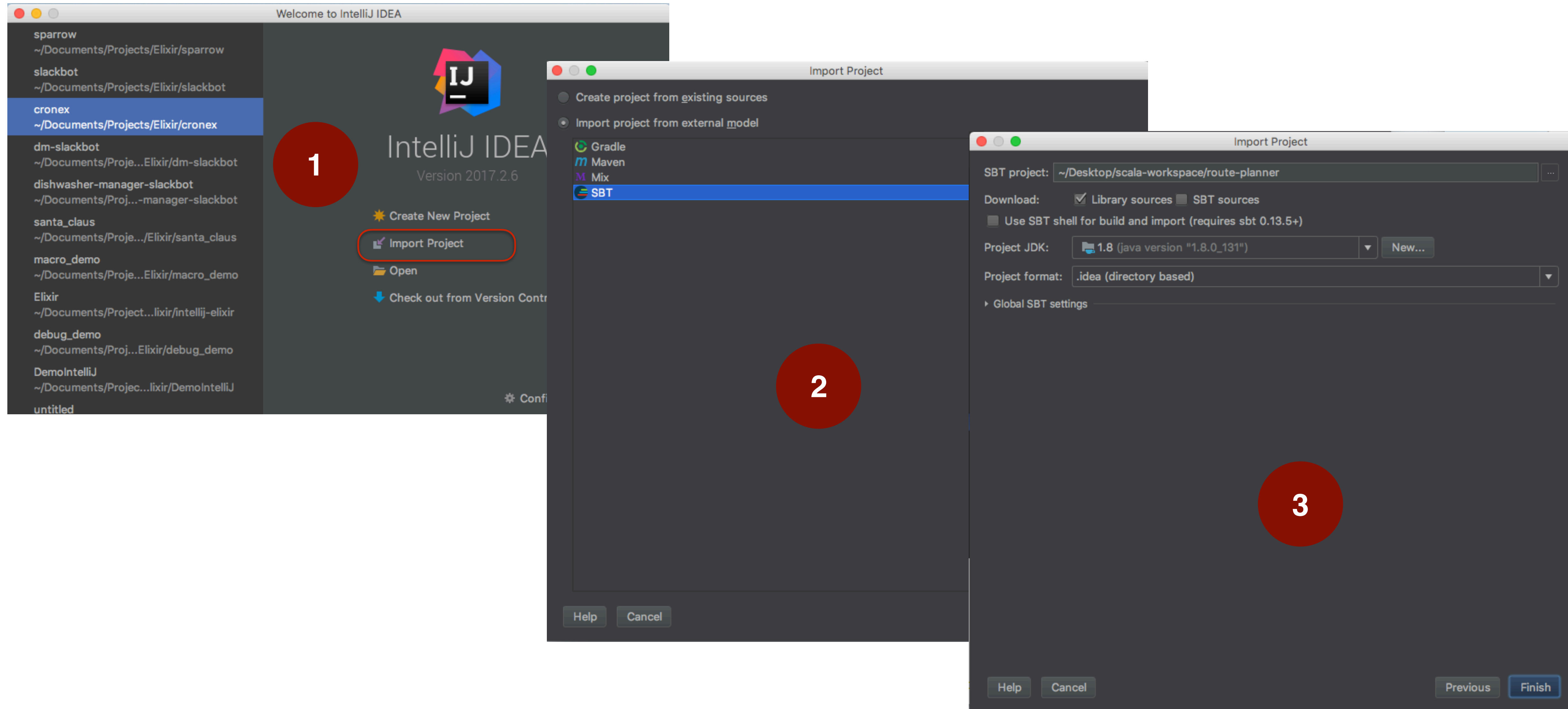
wait.....

3. Test your project

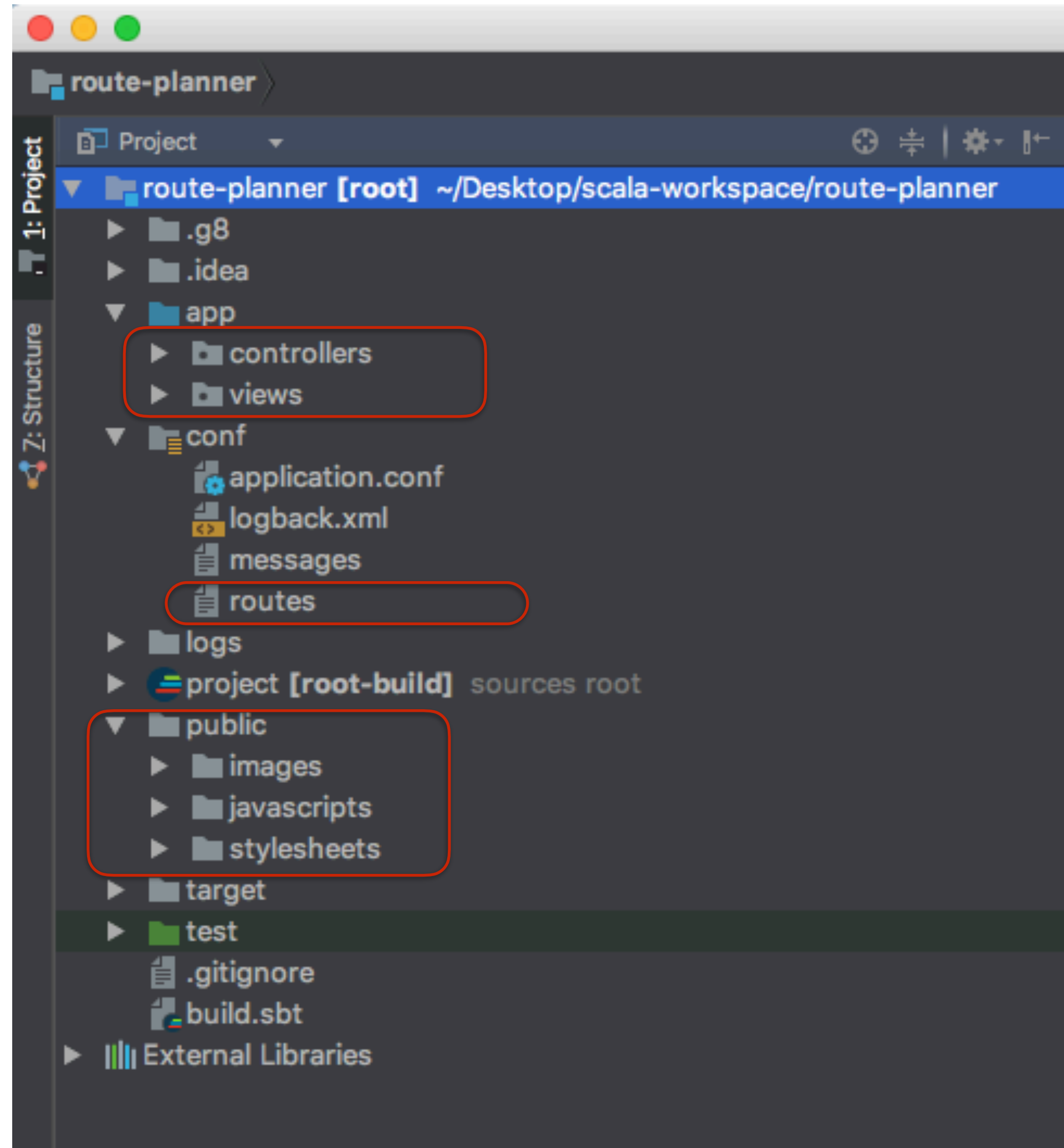
<http://localhost:9000/>



Import project to IntelliJ



Play Project Structure



Default Files: main.scala.html

```
main.scala.html x
1  @*
2  * This template is called from the `index` template. This template
3  * handles the rendering of the page header and body tags. It takes
4  * two arguments, a `String` for the title of the page and an `Html`
5  * object to insert into the body of the page.
6  *@
7  @(title: String)(content: Html)
8
9  <!DOCTYPE html>
10 <html lang="en">
11   <head>
12     @* Here's where we render the page title `String`. *@
13     <title>@title</title>
14     <link rel="stylesheet" media="screen" href="@routes.Assets.versioned("stylesheets/main.css")">
15     <link rel="shortcut icon" type="image/png" href="@routes.Assets.versioned("images/favicon.png")">
16
17   </head>
18   <body>
19     @* And here's where we render the `Html` object containing
20     * the page content. *@
21     @content
22
23     <script src="@routes.Assets.versioned("javascripts/main.js")" type="text/javascript"></script>
24   </body>
25 </html>
26
```


Default Files: `index.scala.html`

```
index.scala.html x
1  @()
2
3  @main("Welcome to Play") {
4    <h1>Welcome to Play!</h1>
5  }
6
```

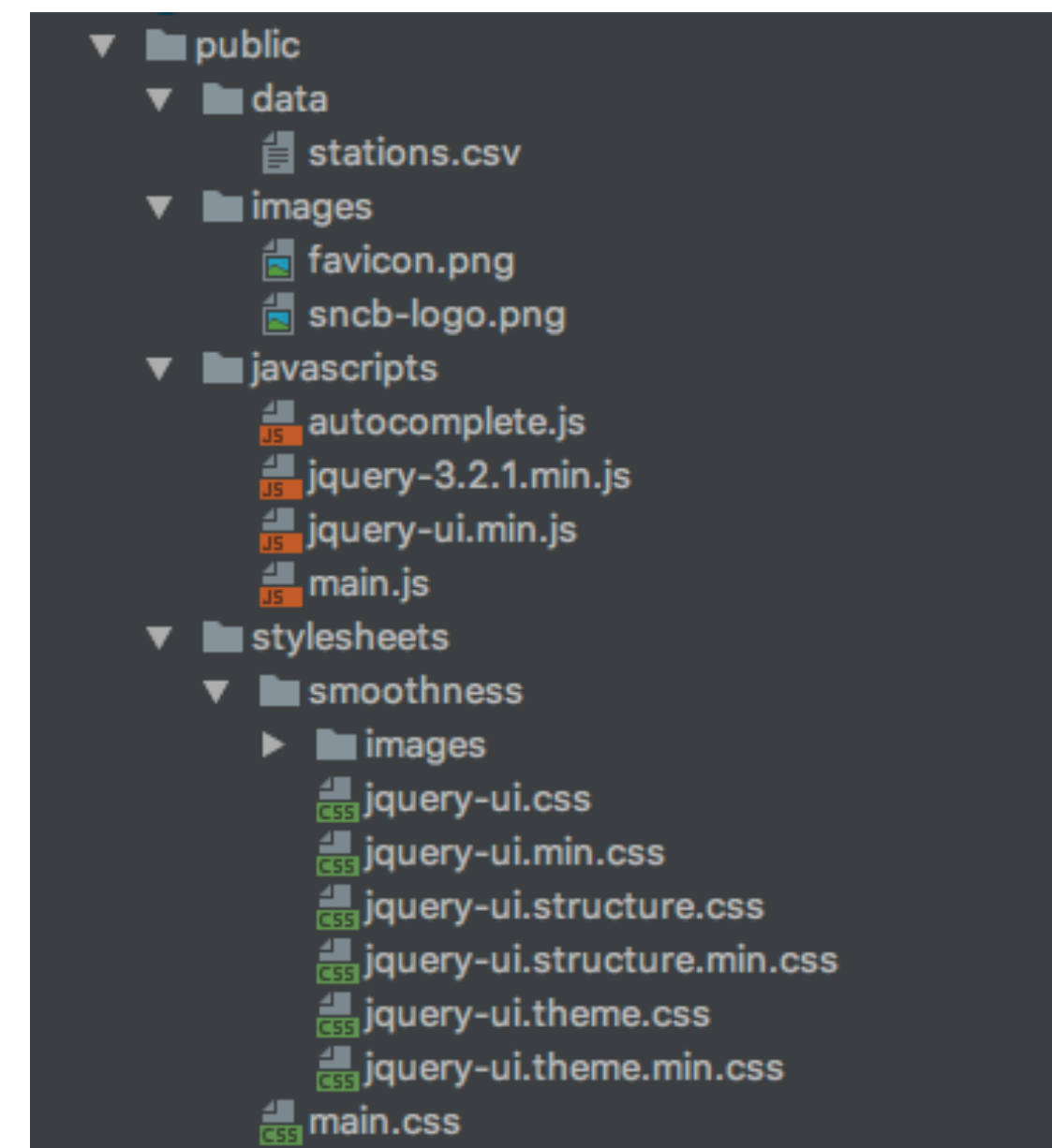
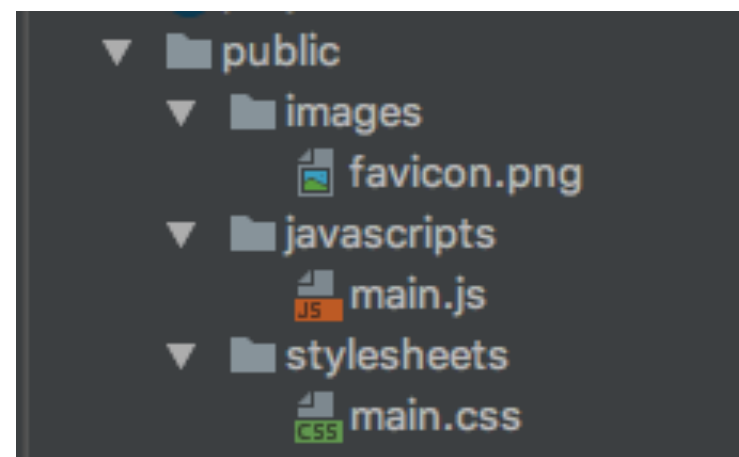
Default Files: routes

```
space/route-planner/conf/routes
1  # Routes
2  # This file defines all application routes (Higher priority routes first)
3  # https://www.playframework.com/documentation/latest/ScalaRouting
4  # ~~~~
5
6  # An example controller showing a sample home page
7  GET      /                               controllers.HomeController.index
8
9  # Map static resources from the /public folder to the /assets URL path
10 GET      /assets/*file                   controllers.Assets.versioned(path="/public", file: Asset)
11
```

Building Route Planner App

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

2. Replace the default **public** directory



Create a RouteForm Object

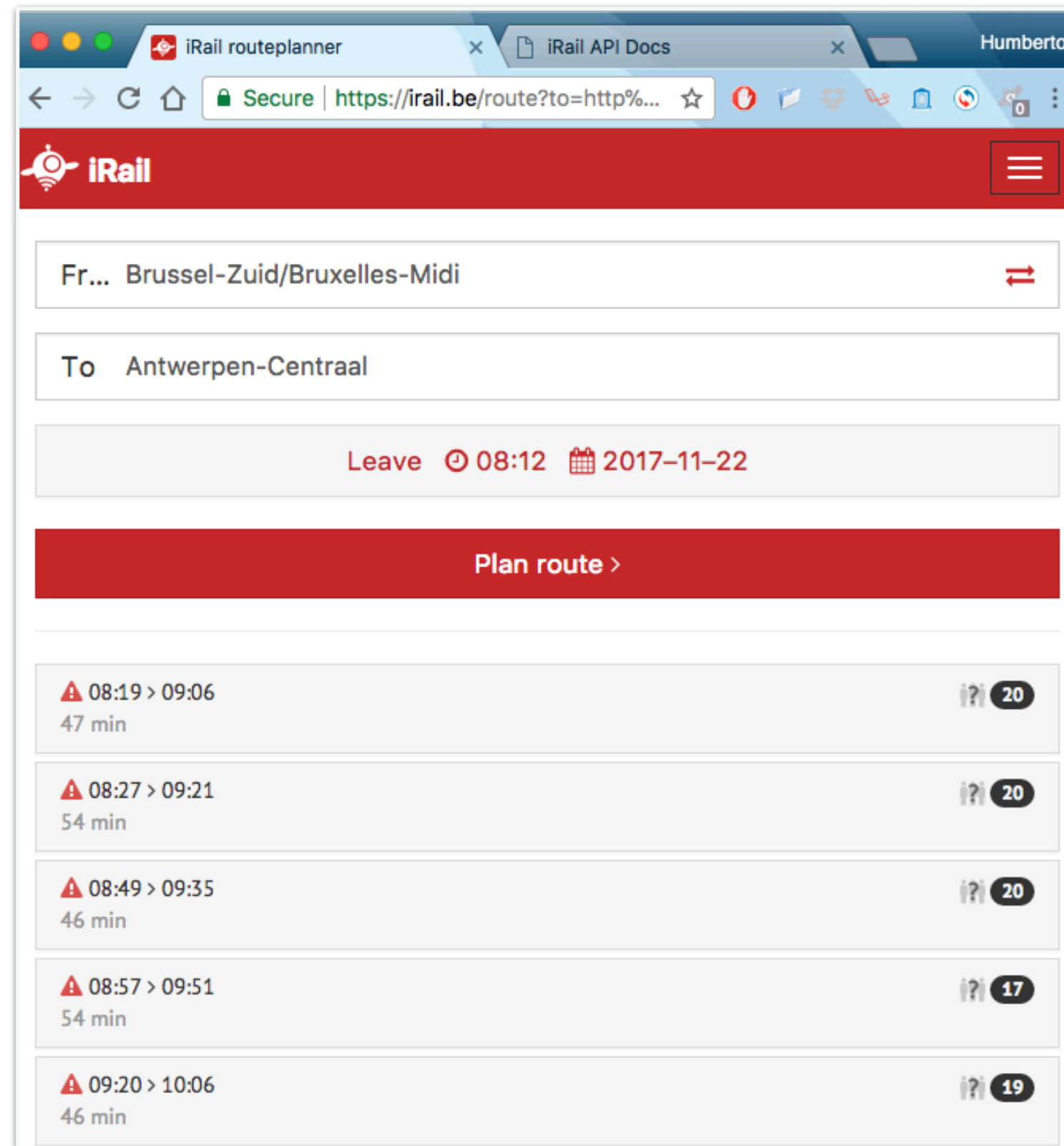
```
RouteForm.scala x
1  package controllers
2
3  object RouteForm {
4
5      import play.api.data.Forms._
6      import play.api.data.Form
7
8      /**
9       * A form processing DTO that maps to the form below.
10     *
11     * Using a class specifically for form binding reduces the chances
12     * of a parameter tampering attack and makes code clearer.
13     */
14     case class Data(from: String, to: String)
15
16     /**
17     * The form definition for the "create a route" form.
18     * It specifies the form fields and their types,
19     * as well as how to convert from a RouteData to form data and vice versa.
20     */
21     val form = Form(
22         mapping(
23             "From" => nonEmptyText,
24             "To"   => nonEmptyText
25         )(Data.apply)(Data.unapply)
26     )
27
28
29
30 }
31
```

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

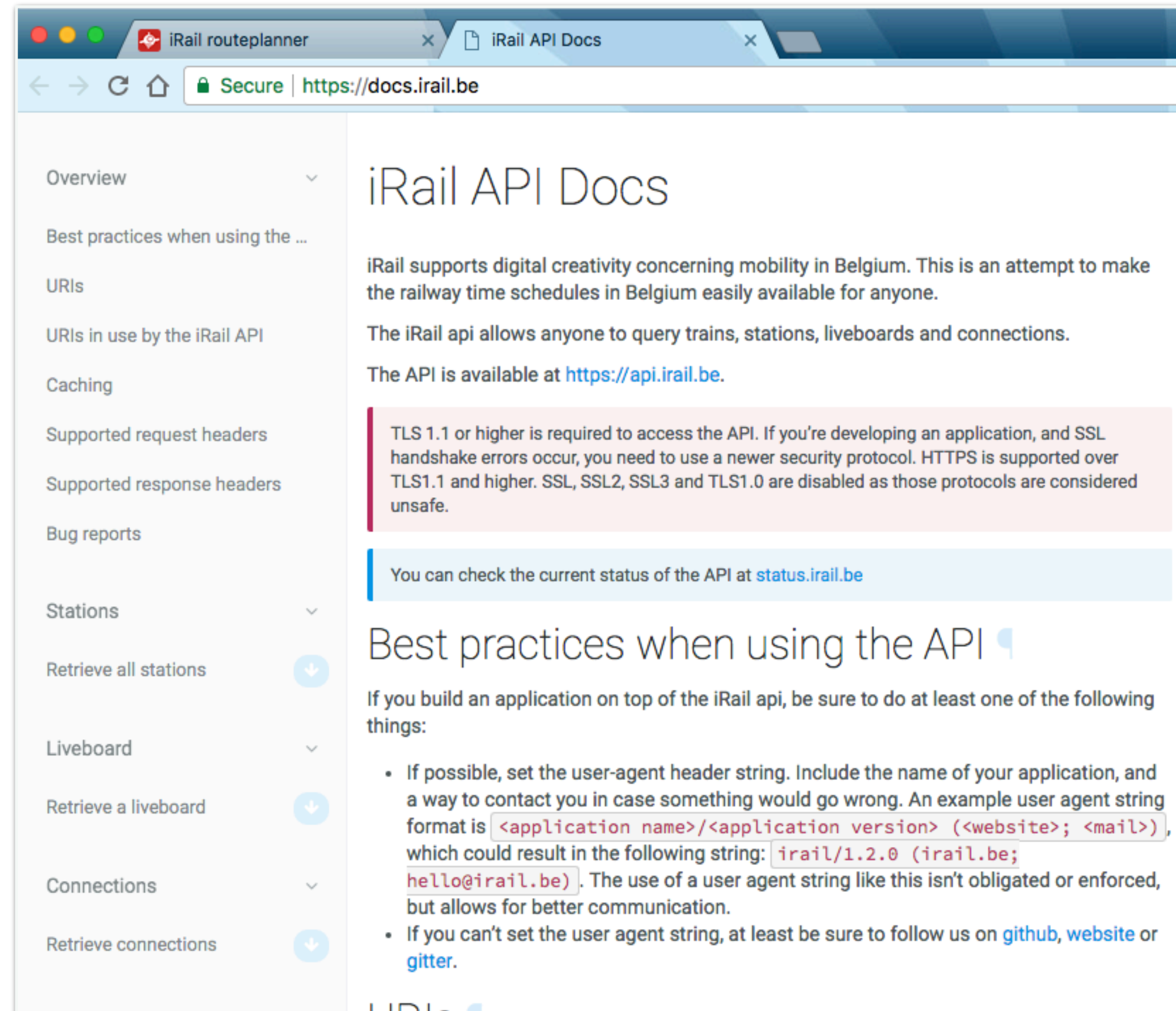
Update main.scala.html

```
main.scala.html x
1  @(title: String)(content: Html)
2
3  <!DOCTYPE html>
4  <html lang="en">
5    <head>
6      <title>@title</title>
7      <link rel="stylesheet" media="screen" href="@routes.Assets.versioned("stylesheets/main.css")">
8      <link rel="stylesheet" media="screen" href="@routes.Assets.versioned("stylesheets/smoothness/jquery-ui.min.css")">
9      <link rel="shortcut icon" type="image/png" href="@routes.Assets.versioned("images/favicon.png")">
10    </head>
11    <body>
12      @content
13
14      <script src="@routes.Assets.versioned("javascripts/jquery-3.2.1.min.js")" type="text/javascript"></script>
15      <script src="@routes.Assets.versioned("javascripts/jquery-ui.min.js")" type="text/javascript"></script>
16      <script type="text/javascript" src="@routes.Application.javascriptRoutes"></script>
17
18      <script src="@routes.Assets.versioned("javascripts/autocomplete.js")" type="text/javascript"></script>
19      <script src="@routes.Assets.versioned("javascripts/main.js")" type="text/javascript"></script>
20    </body>
21  </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 be matched with an id and name according to iRail (Stations, Plaatsen, etc.)

GET /connections/?from=Gent-Sint-Pieters&to=Mechelen&date=300917&time=1230×el=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: *

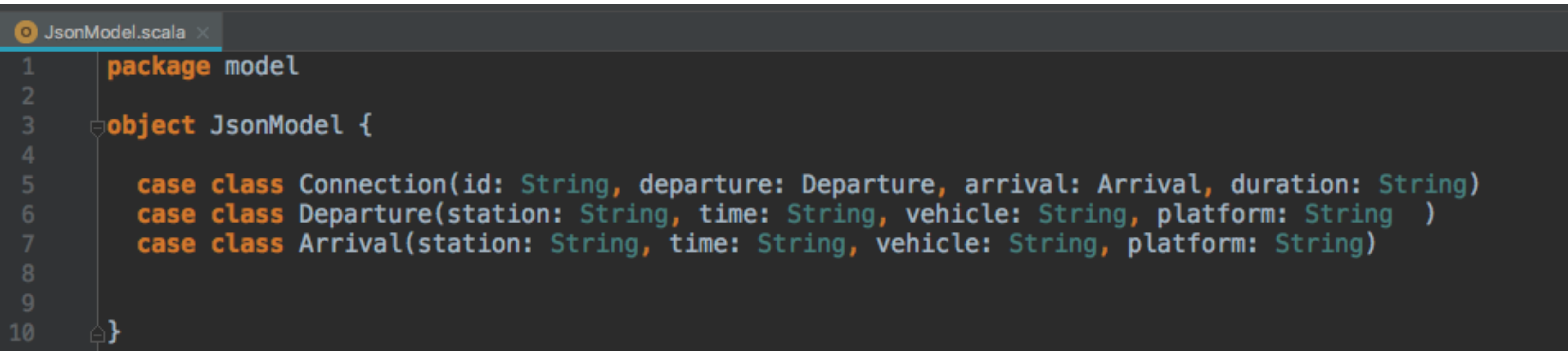
cache-control: Public

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",
 "vehicleinfo": {
 "id": "BE.NMBS.IC3033",
 "@id": "http://irail.be/voertuigen/NMBS/BE.NMBS.IC3033",
 "locationX": 4.347838,
 "locationY": 50.832534,
 "standardname": "IC3033",
 "name": "IC3033"
 }
 }
]
}

Create a JsonModel Object

A screenshot of an IDE window titled 'JsonModel.scala'. The code is written in Scala and defines a package 'model' containing an object 'JsonModel'. Inside 'JsonModel', there are three case classes: 'Connection' with parameters 'id: String', 'departure: Departure', 'arrival: Arrival', and 'duration: String'; 'Departure' with parameters 'station: String', 'time: String', 'vehicle: String', and 'platform: String'; and 'Arrival' with parameters 'station: String', 'time: String', 'vehicle: String', and 'platform: String'. The code is as follows:

```
1 package model
2
3 object JsonModel {
4
5     case class Connection(id: String, departure: Departure, arrival: Arrival, duration: String)
6     case class Departure(station: String, time: String, vehicle: String, platform: String )
7     case class Arrival(station: String, time: String, vehicle: String, platform: String)
8
9 }
10
```

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 x
1  @(routeForm: Form[RouteForm.Data], connections: Seq[model.JsonModel.Connection], postURL: Call)(implicit request: MessagesRequestHeader)
2
3  @main(title = "Route Planner Home") {
4
5      
6
7      @helper.form(postURL) {
8          <div id="form">
9              @helper.CSRF.formField
10             <!--<label for="from">From</label-->
11             @helper.inputText(routeForm("From"), 'id -> "from", 'placeholder -> "Brussels", 'autofocus -> "")
12             <!--<input type="text" id="from" placeholder="Brussels" autofocus="" />-->
13             <!--<label for="to">To</label-->
14             @helper.inputText(routeForm("To"), 'id -> "to", 'placeholder -> "Antwerp")
15             <!--<input type="text" id="to" placeholder="Antwerp", />-->
16             <button> Plan route</button>
17         </div>
18     }
19 }
```

Retrieving Stations Name with Ajax

```
autocomplete.js x
1  $("#to, #from").autocomplete({
2      source: function(request, response){
3          $.ajax(jsRoutes.controllers.RoutePlanner.stations(request.term))
4              .done(function(data){
5                  response($.map(data, function(item){
6                      return { label: item, value: item}
7                  }));
8              })
9              .fail(function(error){console.log(error)});
10     },
11     minLength: 2
12 });
```

From To

- Ville-Pommerœul
- Villers-la-Ville
- Vilvoorde


```

14
15 + /**...*/
19 @Singleton
20 - class RoutePlanner @Inject()(cc: MessagesControllerComponents, ws: WSClient) extends MessagesAbstractController(cc) {
21
22
23     var stationsFile = Environment.simple().getFile("/public/data/stations.csv")
24     var stationsList = List[String]()
25     private val connections = ArrayBuffer[Connection]()
26
27
28     // URL to the method to calculate the connections. You can call this directly from the template, but it
29     // can be more convenient to leave the template completely stateless
30     private val postURL = routes.RoutePlanner.calculateConnections()
31
32
33     def index() = Action { implicit request: MessagesRequest[AnyContent] =>
34         stationsList = io.Source.fromFile(stationsFile).getLines.toList
35         Ok(views.html.index(form, connections, postURL))
36     }
37
38     // This will be the action that handles our ajax request
39     def stations(term: String) = Action { implicit request: Request[AnyContent] =>
40         val suggestions = stationsList.filter(_.startsWith(term))
41         Ok(Json.toJson(suggestions))
42     }
43
44
45     // This will be the action that handles our form post
46     def calculateConnections() = Action { implicit request: MessagesRequest[AnyContent] =>
47         // "TODO: to implement for assignment 3"
48         Ok(views.html.index(form, connections, postURL))
49     }
50
51 }

```

Create an Application Controller

```
Application.scala x
1 package controllers
2
3 import javax.inject.Inject
4 import play.api.mvc._
5 import play.api.routing._
6
7 class Application @Inject()(components: ControllerComponents) extends AbstractController(components) {
8
9   def javascriptRoutes = Action { implicit request =>
10     ok(
11       JavaScriptReverseRouter("jsRoutes")(
12         routes.javascript.RoutePlanner.stations,
13       )
14     ).as("text/javascript")
15   }
16
17 }
```


Update index.scala.html

```
index.scala.html x
1  @(routeForm: Form[RouteForm.Data], connections: Seq[model.JsonModel.Connection], postURL: Call)(implicit request: MessagesRequestHeader)
2
3  @main(title = "Route Planner Home") {
4
5      
6
7      @helper.form(postURL) {
8          <div id="form">
9              @helper.CSRF.formField
10              <!--<label for="from">From</label-->
11              @helper.inputText(routeForm("From"), 'id -> "from", 'placeholder -> "Brussels", 'autofocus -> "")
12              <!--<input type="text" id="from" placeholder="Brussels" autofocus="" />-->
13              <!--<label for="to">To</label-->
14              @helper.inputText(routeForm("To"), 'id -> "to", 'placeholder -> "Antwerp")
15              <!--<input type="text" id="to" placeholder="Antwerp" />-->
16              <button> Plan route</button>
17          </div>
18      }
19  }
```

```
25  <div id="accordion">
26      @for(c <- connections) {
27          <h4>Route @c.departure.time-@c.arrival.time Duration: @c.duration mins </h4>
28          <div>
29              <p>Dep. @c.departure.time @c.departure.vehicle @c.departure.station Platform: @c.departure.platform </p>
30              <p>Arr. @c.arrival.time @c.arrival.vehicle @c.arrival.station Platform: @c.arrival.platform </p>
31          </div>
32      }
33  </div>
```

Software Architectures

Play Framework

Nov 22, 2018

Assistants: Humberto Rodríguez Avila, Kennedy Kambona

Email: {rhumbert, kkambona}@vub.be