Master : Mobiquité & Big Data

Project Report

Angular with MongoDb and Highcharts

Realized by:

- ROUZKI Zakarya

Table of contents:

1. Introduction

Angular is a JavaScript framework for building web applications and apps in JavaScript, html, and TypeScript, which is a superset of JavaScript. Angular provides built-in features for animation, http service, and materials which in turn has features such as auto-complete, navigation, toolbar, menus, etc. The code is written in TypeScript, which compiles to JavaScript and displays the same in the browser. In this project we will create an angular application and we will connect it to our NoSQL database which it's a **MongoDb** database, and after retrieving data we will use **Highcharts** library which it's is a pure JavaScript based charting library meant to enhance web applications by adding interactive charting capability. Highcharts provides a wide variety of charts. For example, line charts, spline charts, area charts, bar charts, pie charts and so on. This tutorial will teach you the basics of Highcharts.

2. Setting up the work environment:

2.1 – Front End environment:

In this section we will discuss the Environment Setup required for the front End environment (Angular, Highcharts).

To install Angular, we will need the following tools:

- ✓ Nodejs
- ✓ Npm
- ✓ Angular CLI
- ✓ IDE for writing your code

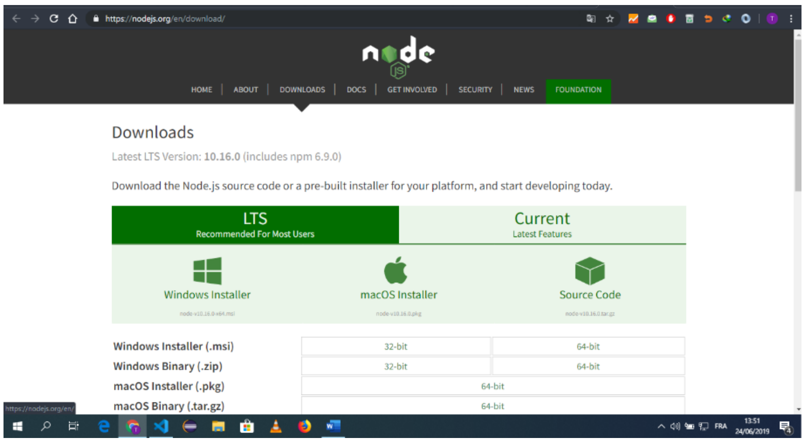
And for Highcharts we will demonstrate how to install it.

2.1.1 – NodeJS and Npm:

First what is Nodejs?

Node.js is an open source, cross-platform runtime environment for developing server-side and networking applications. Node.js applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux.

To install nodejs, go the homepage https://nodejs.org/en/download/ of nodejs and install the package based on your OS.



Downloads

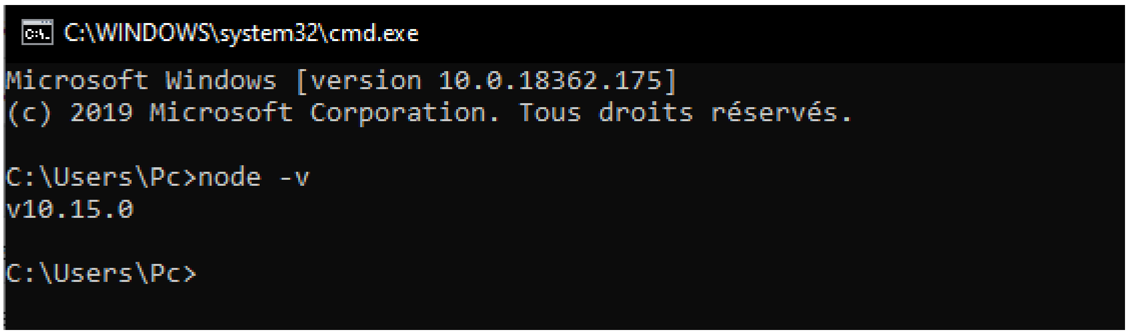
Latest LTS Version: **10.16.0** (includes npm 6.9.0)

Download the Node.js source code or a pre-built installer for your platform, and start developing today.

LTS Recommended For Most Users		Current Latest Features
Windows Installer node-v10.16.0-x64.msi	macOS Installer node-v10.16.0.pkg	Source Code node-v10.16.0.tar.gz

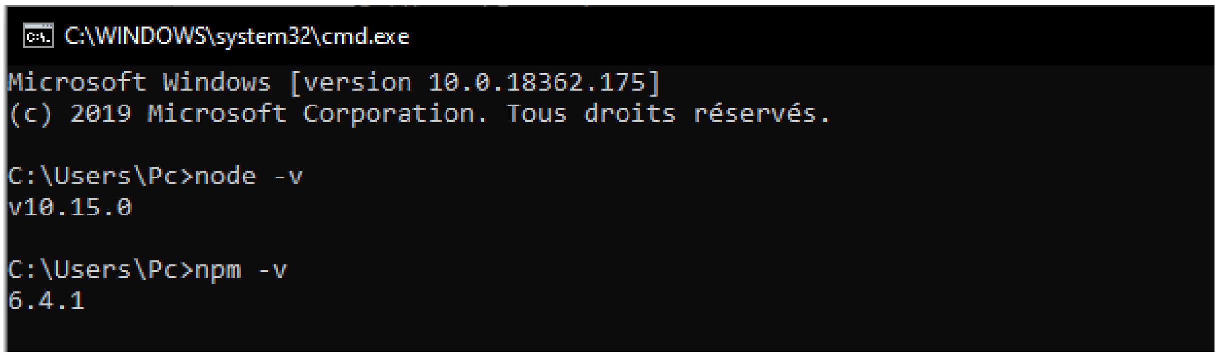
Windows Installer (.msi)	32-bit	64-bit
Windows Binary (.zip)	32-bit	64-bit
macOS Installer (.pkg)	64-bit	
macOS Binary (.tar.gz)	64-bit	

Based on your OS, install the required package. To check if NodeJS is successfully installed run the following command node -v on command line:



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [version 10.0.18362.175]
(c) 2019 Microsoft Corporation. Tous droits réservés.

C:\Users\Pc>node -v
v10.15.0

C:\Users\Pc>
```

Once nodejs is installed, **npm** will also get installed along with it. To check if npm is installed or not, type npm -v in the terminal. It should display the version of the npm.



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [version 10.0.18362.175]
(c) 2019 Microsoft Corporation. Tous droits réservés.

C:\Users\Pc>node -v
v10.15.0

C:\Users\Pc>npm -v
6.4.1
```

2.1.2 – Angular CLI:

```
> npm install -g angular-cli
> ng new my-dream-app
> cd my-dream-app
> ng serve
```

Until now we have configured a global environment , Now, we will move to the next step which it's configuring our special environment for installing angular.

We will need to install **Angular CLI** (Angular Command Line Interface) which it is a command line tool for creating angular apps. It is recommended to use angular cli for creating angular apps as you don't need to spend time installing and configuring all the required dependencies and wiring everything together.

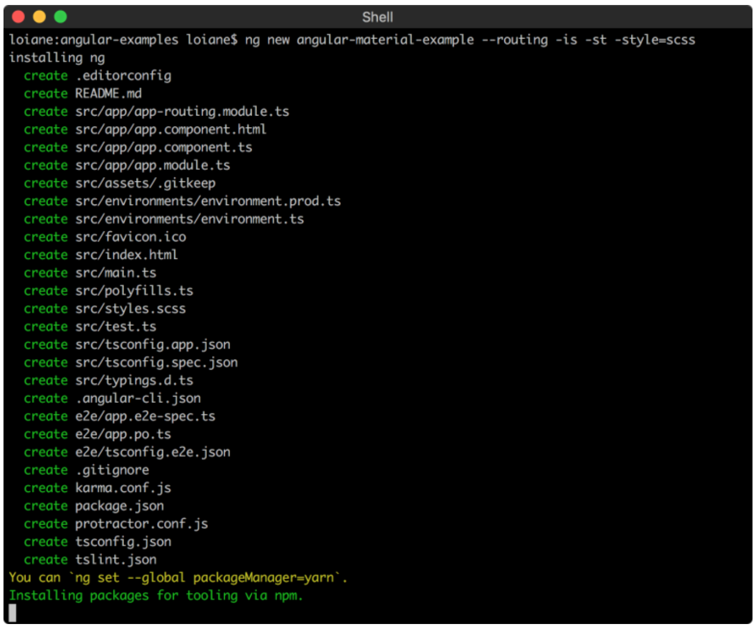
To install Angular CLI type the following commands:

npm install -g @angular/cli

After that the installation is completed, we will create our angular project using the follow command with angular CLI

ng new Angular-MongoDB

the result of the execution of the command must look the same as follows



```
loiane:angular-examples loiane$ ng new angular-material-example --routing -is -st -style=scss
installing ng
  create .editorconfig
  create README.md
  create src/app/app-routing.module.ts
  create src/app/app.component.html
  create src/app/app.component.ts
  create src/app/app.module.ts
  create src/assets/.gitkeep
  create src/environments/environment.prod.ts
  create src/environments/environment.ts
  create src/favicon.ico
  create src/index.html
  create src/main.ts
  create src/polyfills.ts
  create src/styles.scss
  create src/test.ts
  create src/tsconfig.app.json
  create src/tsconfig.spec.json
  create src/typings.d.ts
  create .angular-cli.json
  create e2e/app.e2e-spec.ts
  create e2e/app.po.ts
  create e2e/tsconfig.e2e.json
  create .gitignore
  create karma.conf.js
  create package.json
  create protractor.conf.js
  create tsconfig.json
  create tslint.json
You can `ng set --global packageManager=yarn`.
Installing packages for tooling via npm.
```

2.1.3 – Highcharts:

To install Highcharts in the angular project we need to run the following commands:

```
npm install highcharts –save
```

```
npm install highcharts-angular --save
```

Now we have configured all of our frontend environment.

2.2 – Back End environment:

In this section we will demonstrate how to configure the packages that are necessary for retrieving data from MongoDb, so we will create a RestApi application.

To configure our back-End environment, we need these packages:

- ✓ Express.
- ✓ Mongoose.
- ✓ Body parser.

2.2.1 Express:

Express is a routing and middleware web framework that has minimal functionality of its own and is essentially a series of middleware function calls.

To install Express run the follow command:

```
npm install express—save
```

2.2.2 Body-Parser:

Body-Parser is a body-parsing middleware which allows us to parse the incoming request bodies in a middleware before your handlers, available under the req.body property come into effect.

Installation of body-Parser:

```
npm install body-parser --save
```

2.2.3 Mongoose:

Mongoose is an object data modeling (ODM) library that provides a rigorous modeling environment for your data, enforcing structure as needed while still maintaining the flexibility that makes MongoDB powerful.
Mongoose helps us connect from NodeJS app to the MongoDB database.

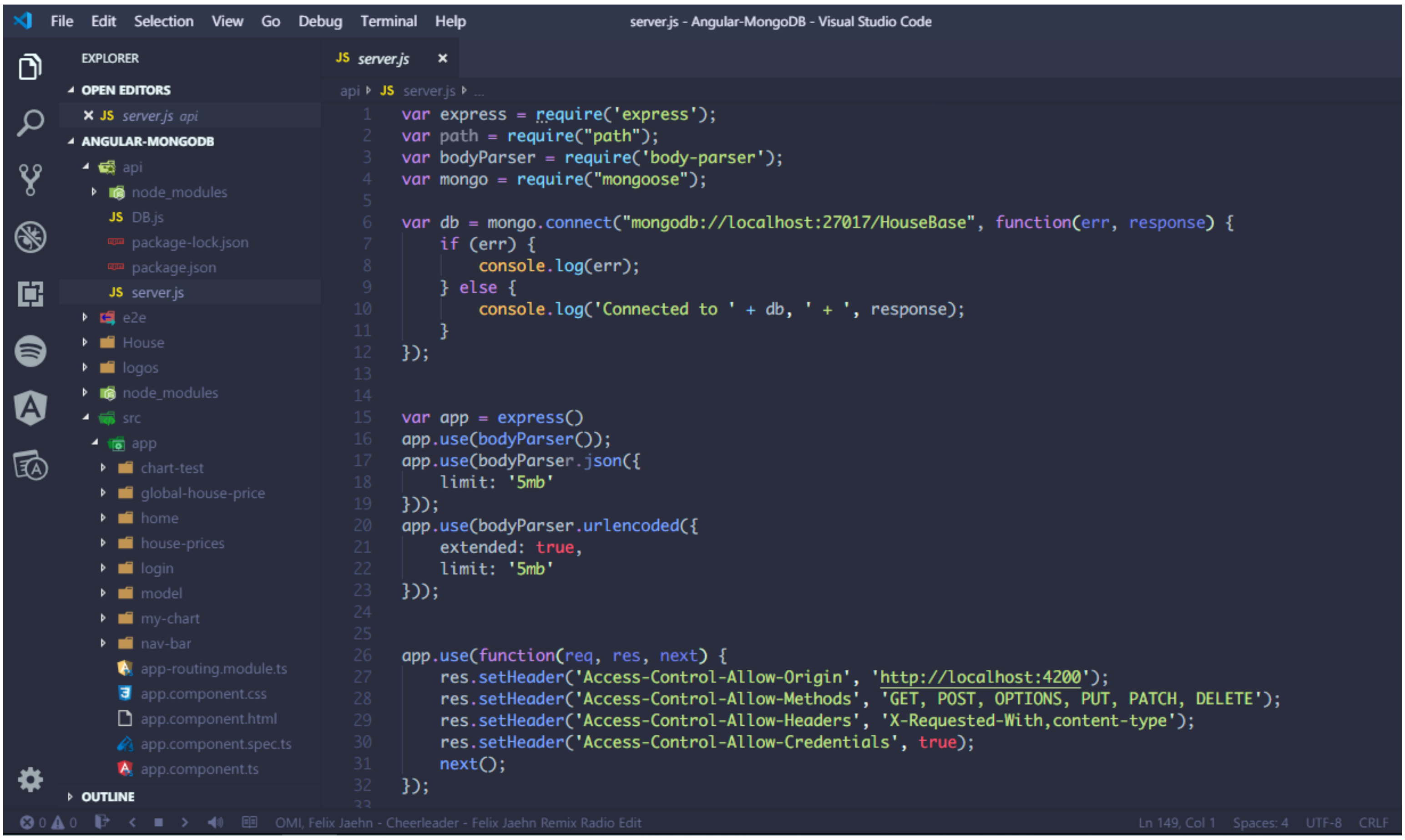
To install it :

```
npm install mongoose—save
```

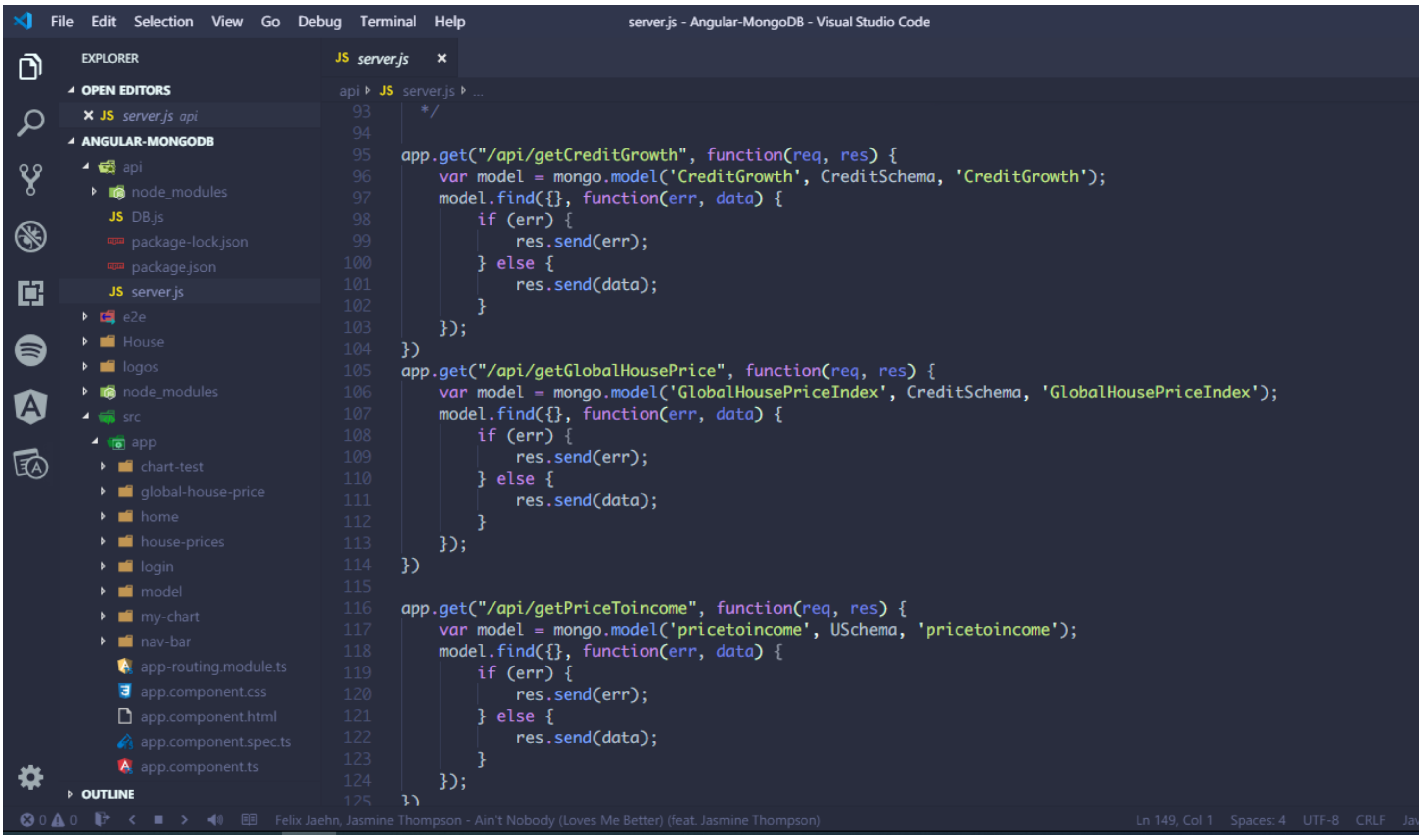
3. Creating a RestApi with MongoDB database – Back-End:

In this section we will build a simple API using Nodejs, Express and MongoDB NoSQL database.
We have already installed the necessary packages, now we will move to the coding part.
We will create a folder with name “api” inside of our angular project that will contains our RestApi files and folders.

Next step is to create a file with name server.js.
The server.js file content will be as follows:

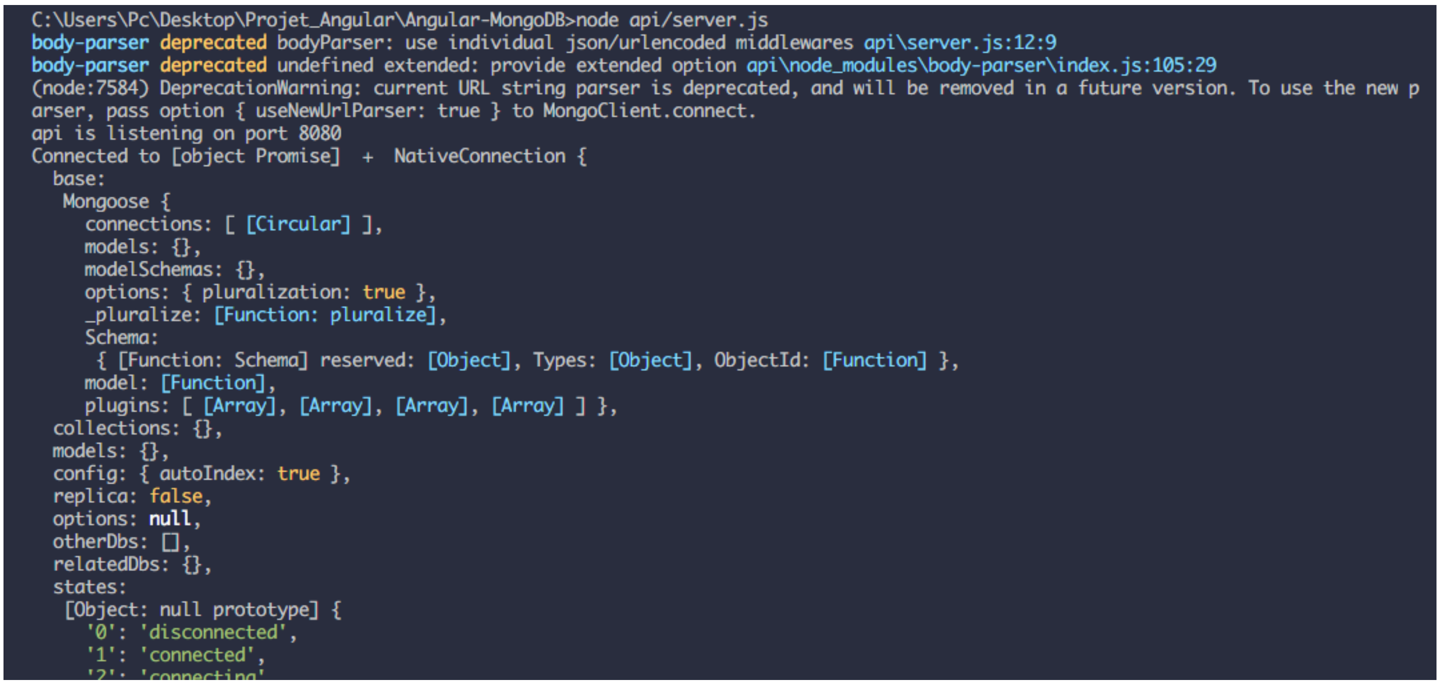


```javascript
var express = require('express');
var path = require("path");
var bodyParser = require('body-parser');
var mongo = require("mongoose");

var db = mongo.connect("mongodb://localhost:27017/HouseBase", function(err, response) {
    if (err) {
        console.log(err);
    } else {
        console.log('Connected to ' + db, ' + ', response);
    }
});


var app = express()
app.use(bodyParser());
app.use(bodyParser.json({
    limit: '5mb'
}));
app.use(bodyParser.urlencoded({
    extended: true,
    limit: '5mb'
}));


app.use(function(req, res, next) {
    res.setHeader('Access-Control-Allow-Origin', 'http://localhost:4200');
    res.setHeader('Access-Control-Allow-Methods', 'GET, POST, OPTIONS, PUT, PATCH, DELETE');
    res.setHeader('Access-Control-Allow-Headers', 'X-Requested-With,content-type');
    res.setHeader('Access-Control-Allow-Credentials', true);
    next();
});
```



```
    */

app.get("/api/getCreditGrowth", function(req, res) {
    var model = mongo.model('CreditGrowth', CreditSchema, 'CreditGrowth');
    model.find({}, function(err, data) {
        if (err) {
            res.send(err);
        } else {
            res.send(data);
        }
    });
})
app.get("/api/getGlobalHousePrice", function(req, res) {
    var model = mongo.model('GlobalHousePriceIndex', CreditSchema, 'GlobalHousePriceIndex');
    model.find({}, function(err, data) {
        if (err) {
            res.send(err);
        } else {
            res.send(data);
        }
    });
})

app.get("/api/getPriceToincome", function(req, res) {
    var model = mongo.model('pricetoincome', USchema, 'pricetoincome');
    model.find({}, function(err, data) {
        if (err) {
            res.send(err);
        } else {
            res.send(data);
        }
    });
})
```

to run our RestApi, we run the following command:
node api/server.js



```
C:\Users\Pc\Desktop\Projet_Angular\Angular-MongoDB>node api/server.js
body-parser deprecated bodyParser: use individual json/urlencoded middlewares api\server.js:12:9
body-parser deprecated undefined extended: provide extended option api\node_modules\body-parser\index.js:105:29
(node:7584) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version. To use the new p
arser, pass option { useNewUrlParser: true } to MongoClient.connect.
api is listening on port 8080
Connected to [object Promise]  +  NativeConnection {
  base:
   Mongoose {
     connections: [ [Circular] ],
     models: {},
     modelSchemas: {},
     options: { pluralization: true },
     _pluralize: [Function: pluralize],
     Schema:
      { [Function: Schema] reserved: [Object], Types: [Object], ObjectId: [Function] },
     model: [Function],
     plugins: [ [Array], [Array], [Array], [Array] ] },
  collections: {},
  models: {},
  config: { autoIndex: true },
  replica: false,
  options: null,
  otherDbs: [],
  relatedDbs: {},
  states:
   [Object: null prototype] {
     '0': 'disconnected',
     '1': 'connected',
```

now, we have successfully created our RestApi which it’s connected to our MongoDB database.

4. Building charts using Highcharts, Angular – Front-End

In this section we will discuss how to get data from the RestApi services and use these data to build charts in the angular project.

<u>**Step 1**</u>: creating a service to get data from the RestApi

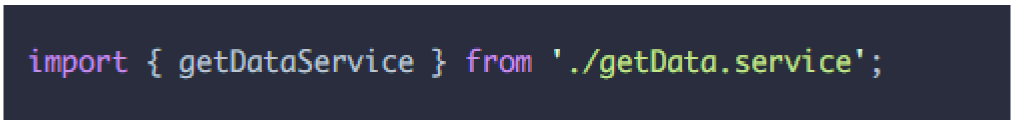
To create a service, we will use angular cli, we will just type the follow command:

ng generate service getData (or just: ng g s getData)

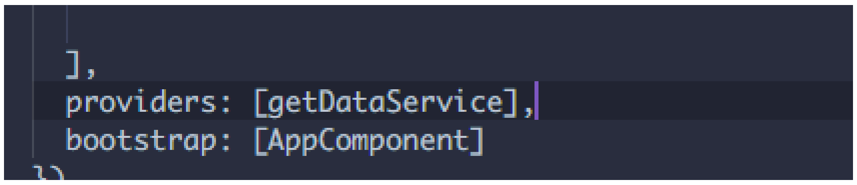
where getData is the name of the service.

We will find a file with name **getData.service.ts** inside of our project.

We need to add the service to our providers in the file **app.module.ts**



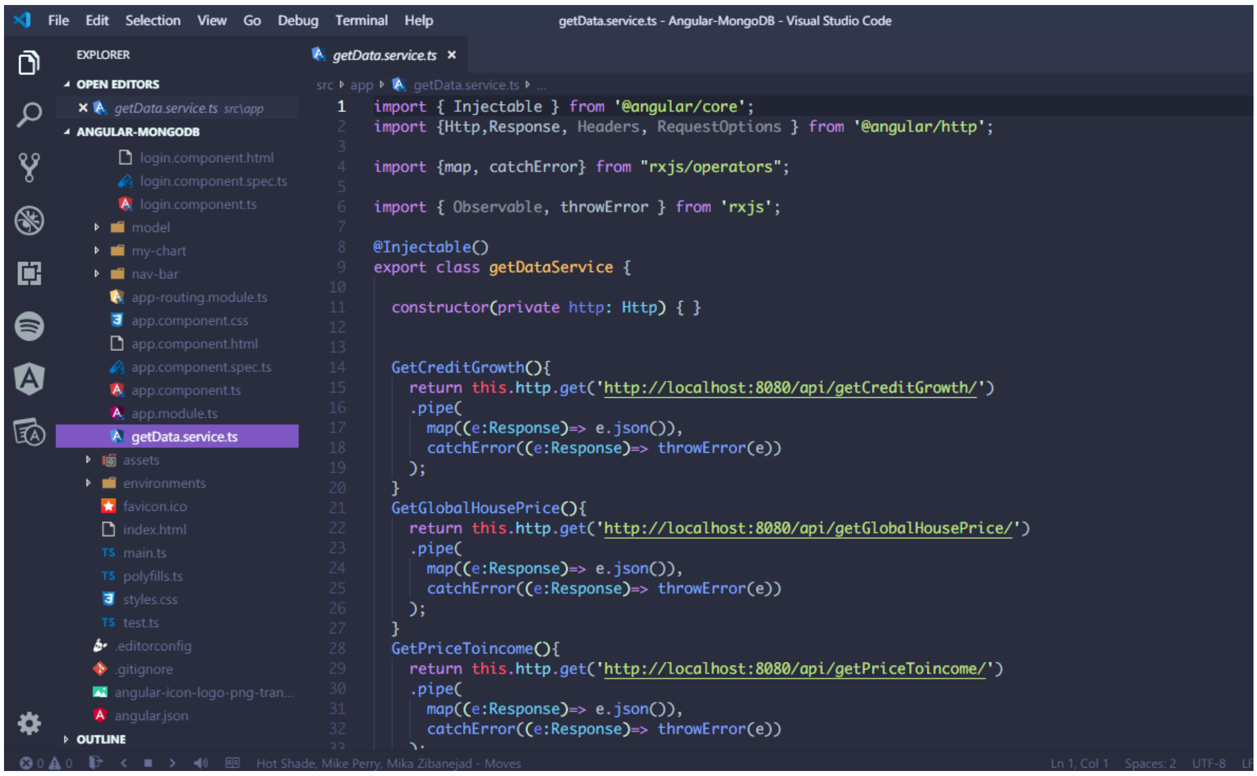
```
import { getDataService } from './getData.service';
```



```
  ],
  providers: [getDataService],
  bootstrap: [AppComponent]
```

and inside of our service we will define our functions of retrieving data using HTTP modules in Angular.

the content of the service getData:



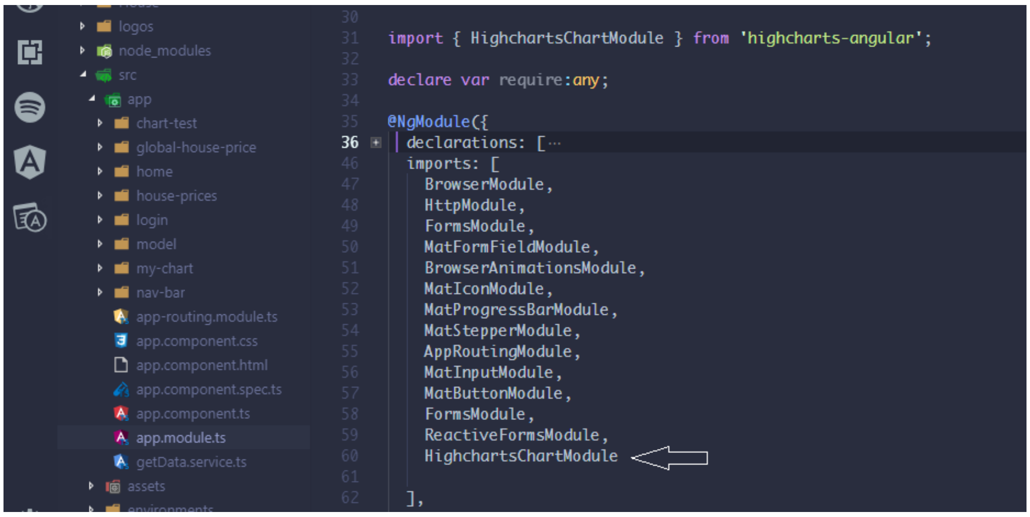
```
import { Injectable } from '@angular/core';
import {Http,Response, Headers, RequestOptions } from '@angular/http';

import {map, catchError} from "rxjs/operators";

import { Observable, throwError } from 'rxjs';

@Injectable()
export class getDataService {

  constructor(private http: Http) { }


  GetCreditGrowth(){
    return this.http.get('http://localhost:8080/api/getCreditGrowth/')
    .pipe(
      map((e:Response)=> e.json()),
      catchError((e:Response)=> throwError(e))
    );
  }
  GetGlobalHousePrice(){
    return this.http.get('http://localhost:8080/api/getGlobalHousePrice/')
    .pipe(
      map((e:Response)=> e.json()),
      catchError((e:Response)=> throwError(e))
    );
  }
  GetPriceToincome(){
    return this.http.get('http://localhost:8080/api/getPriceToincome/')
    .pipe(
      map((e:Response)=> e.json()),
      catchError((e:Response)=> throwError(e))
```

Here we have implemented some function that retrieve data from our RestApi which it's running on port 8080.

Step 2: Implementing Highcharts on the project.

In this step we will see how to import Highcharts library to the angular project, and exploit it with the retrieved data to show charts.

We must declare the Highcharts library in the app.module.ts as follow :



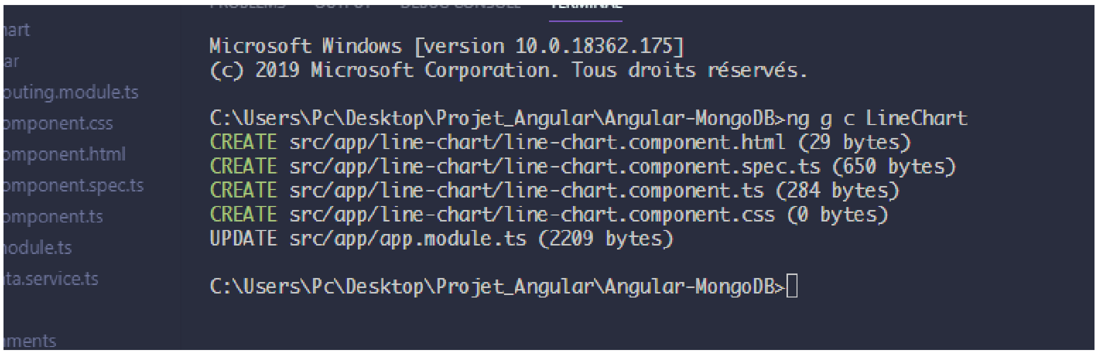
```
import { HighchartsChartModule } from 'highcharts-angular';

declare var require:any;

@NgModule({
  declarations: […
  imports: [
    BrowserModule,
    HttpModule,
    FormsModule,
    MatFormFieldModule,
    BrowserAnimationsModule,
    MatIconModule,
    MatProgressBarModule,
    MatStepperModule,
    AppRoutingModule,
    MatInputModule,
    MatButtonModule,
    FormsModule,
    ReactiveFormsModule,
    HighchartsChartModule

  ],
```

Step 3: Creating a component that will contain the chart and add it to the routing file.

To create a component, we can use Angular CLI and type the follow command:

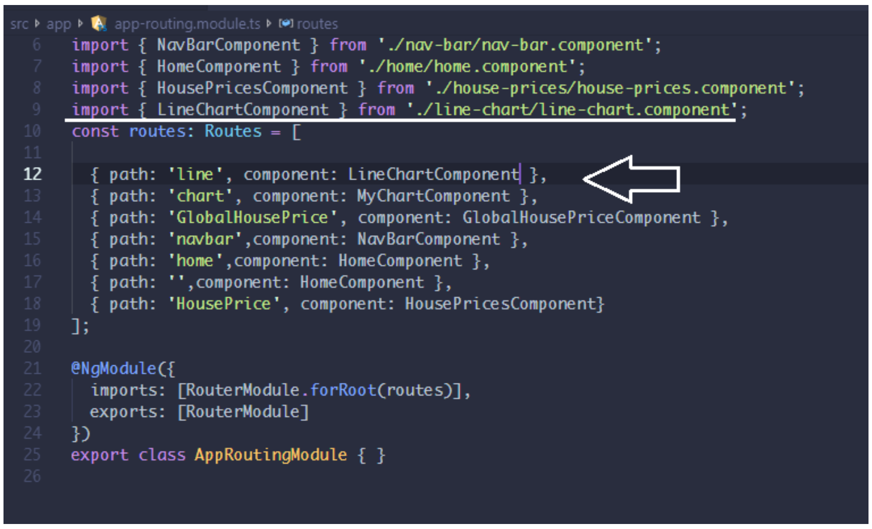
ng generate component ComponentName (or just: ng g c ComponentName)

the result is as follows:



```
Microsoft Windows [version 10.0.18362.175]
(c) 2019 Microsoft Corporation. Tous droits réservés.

C:\Users\Pc\Desktop\Projet_Angular\Angular-MongoDB>ng g c LineChart
CREATE src/app/line-chart/line-chart.component.html (29 bytes)
CREATE src/app/line-chart/line-chart.component.spec.ts (650 bytes)
CREATE src/app/line-chart/line-chart.component.ts (284 bytes)
CREATE src/app/line-chart/line-chart.component.css (0 bytes)
UPDATE src/app/app.module.ts (2209 bytes)

C:\Users\Pc\Desktop\Projet_Angular\Angular-MongoDB>
```

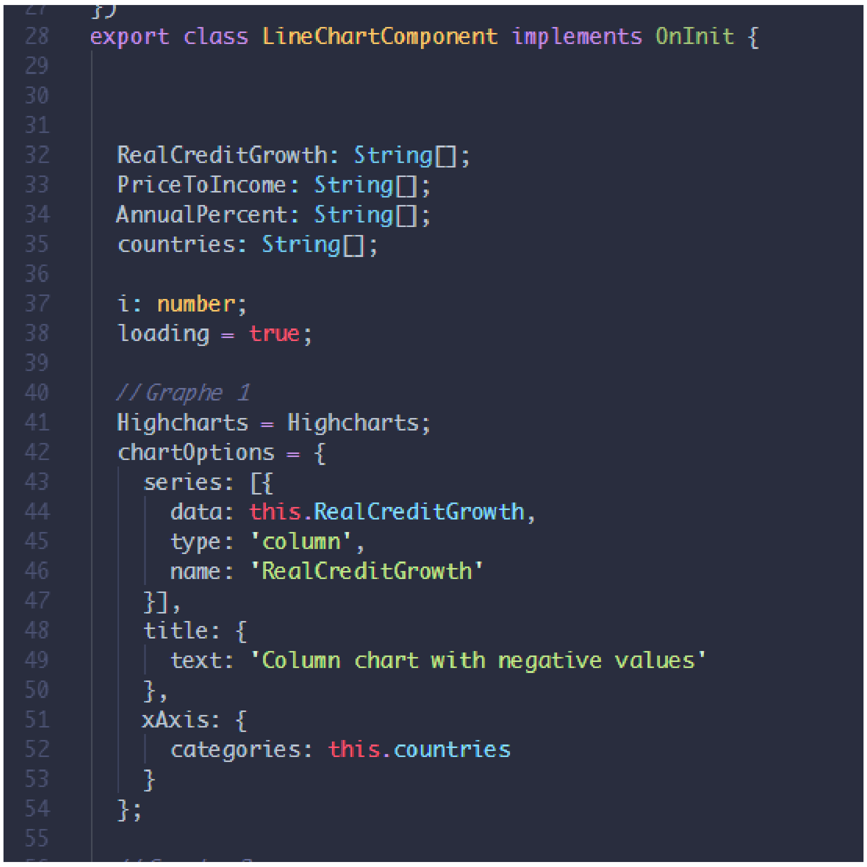
We need to add the component to the routing file app-routing.module.ts as follow :



```
import { NavBarComponent } from './nav-bar/nav-bar.component';
import { HomeComponent } from './home/home.component';
import { HousePricesComponent } from './house-prices/house-prices.component';
import { LineChartComponent } from './line-chart/line-chart.component';
const routes: Routes = [

  { path: 'line', component: LineChartComponent },
  { path: 'chart', component: MyChartComponent },
  { path: 'GlobalHousePrice', component: GlobalHousePriceComponent },
  { path: 'navbar',component: NavBarComponent },
  { path: 'home',component: HomeComponent },
  { path: '',component: HomeComponent },
  { path: 'HousePrice', component: HousePricesComponent}
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

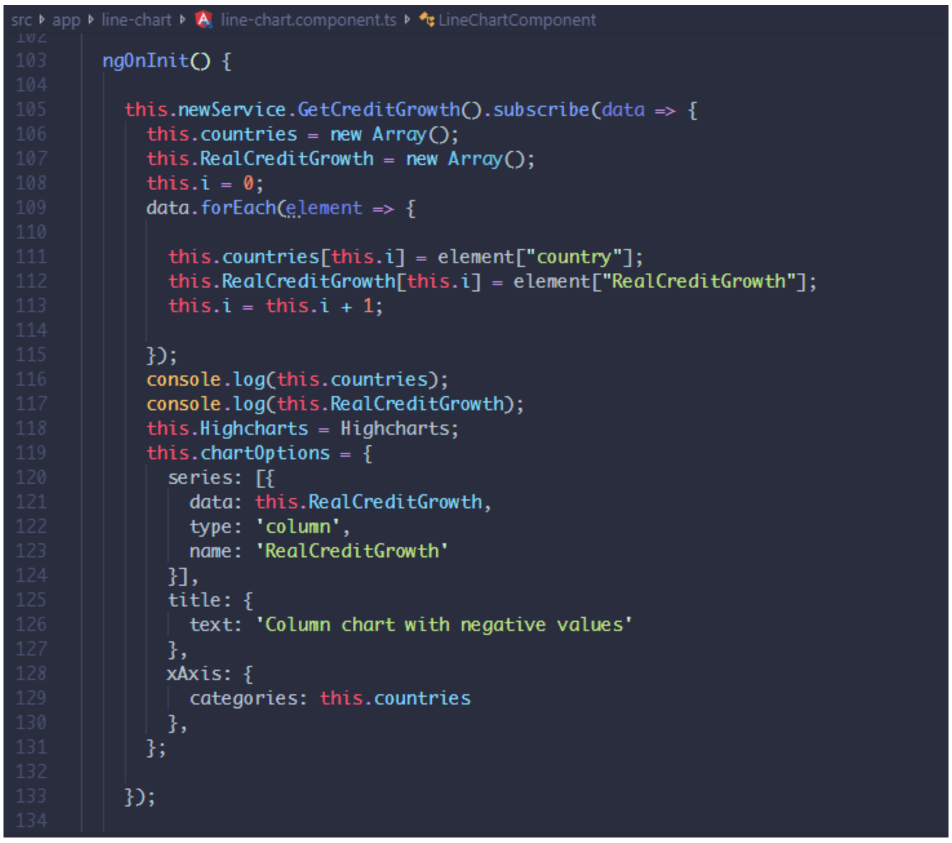
Step 4: receive data inside the component and passing data to the chart.

We will instantiate some array variables to store our data in arrays, and two principal variables for Highcharts (Highcharts and chartOptions), that they are managing the chart.



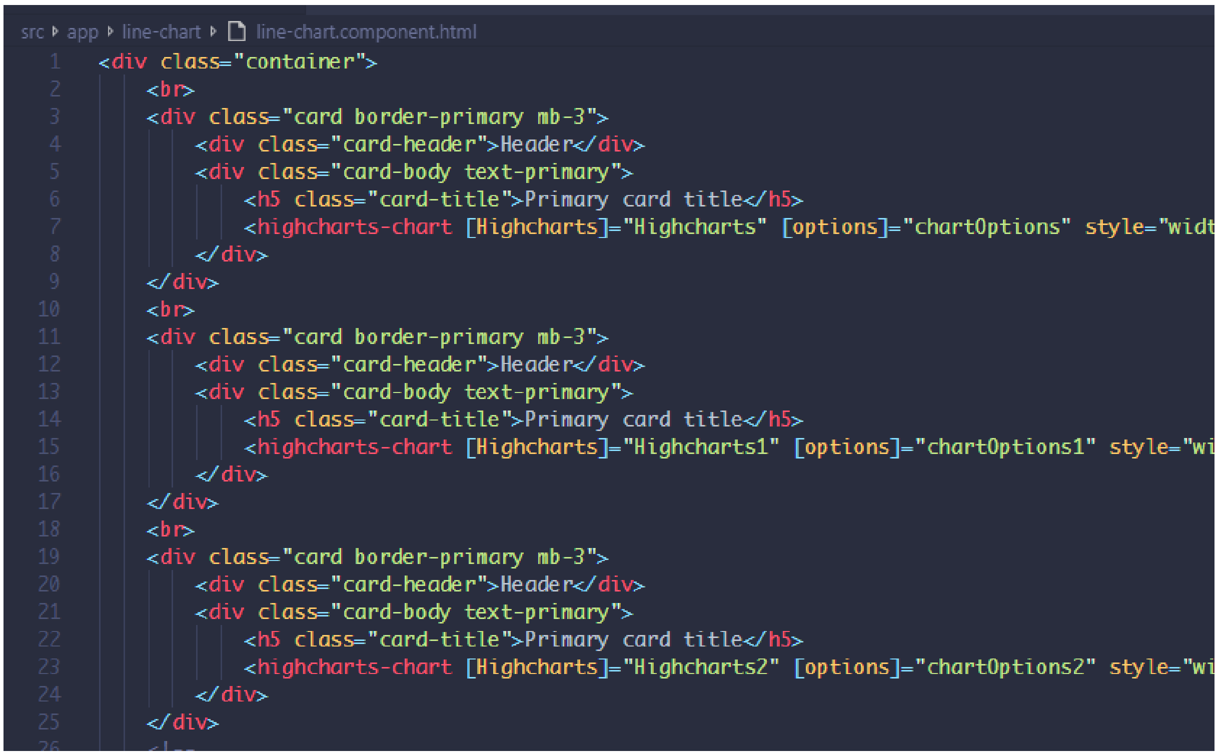
```typescript
})
export class LineChartComponent implements OnInit {



  RealCreditGrowth: String[];
  PriceToIncome: String[];
  AnnualPercent: String[];
  countries: String[];

  i: number;
  loading = true;

  //Graphe 1
  Highcharts = Highcharts;
  chartOptions = {
    series: [{
      data: this.RealCreditGrowth,
      type: 'column',
      name: 'RealCreditGrowth'
    }],
    title: {
      text: 'Column chart with negative values'
    },
    xAxis: {
      categories: this.countries
    }
  };

```

In the function ngOnInit() which it's a function executed when the component load, we will retrieve data using the getData service that we have already created and store the json data to the arrays already created , after we retrieve data and store it we will change the Hightchart options by adding our series and categories which it's used on the chartOptions.



src › app › line-chart › line-chart.component.ts › LineChartComponent

```typescript
  ngOnInit() {

    this.newService.GetCreditGrowth().subscribe(data => {
      this.countries = new Array();
      this.RealCreditGrowth = new Array();
      this.i = 0;
      data.forEach(element => {

        this.countries[this.i] = element["country"];
        this.RealCreditGrowth[this.i] = element["RealCreditGrowth"];
        this.i = this.i + 1;

      });
      console.log(this.countries);
      console.log(this.RealCreditGrowth);
      this.Highcharts = Highcharts;
      this.chartOptions = {
        series: [{
          data: this.RealCreditGrowth,
          type: 'column',
          name: 'RealCreditGrowth'
        }],
        title: {
          text: 'Column chart with negative values'
        },
        xAxis: {
          categories: this.countries
        },
      };

    });

```



src ▸ app ▸ line-chart ▸ line-chart.component.html

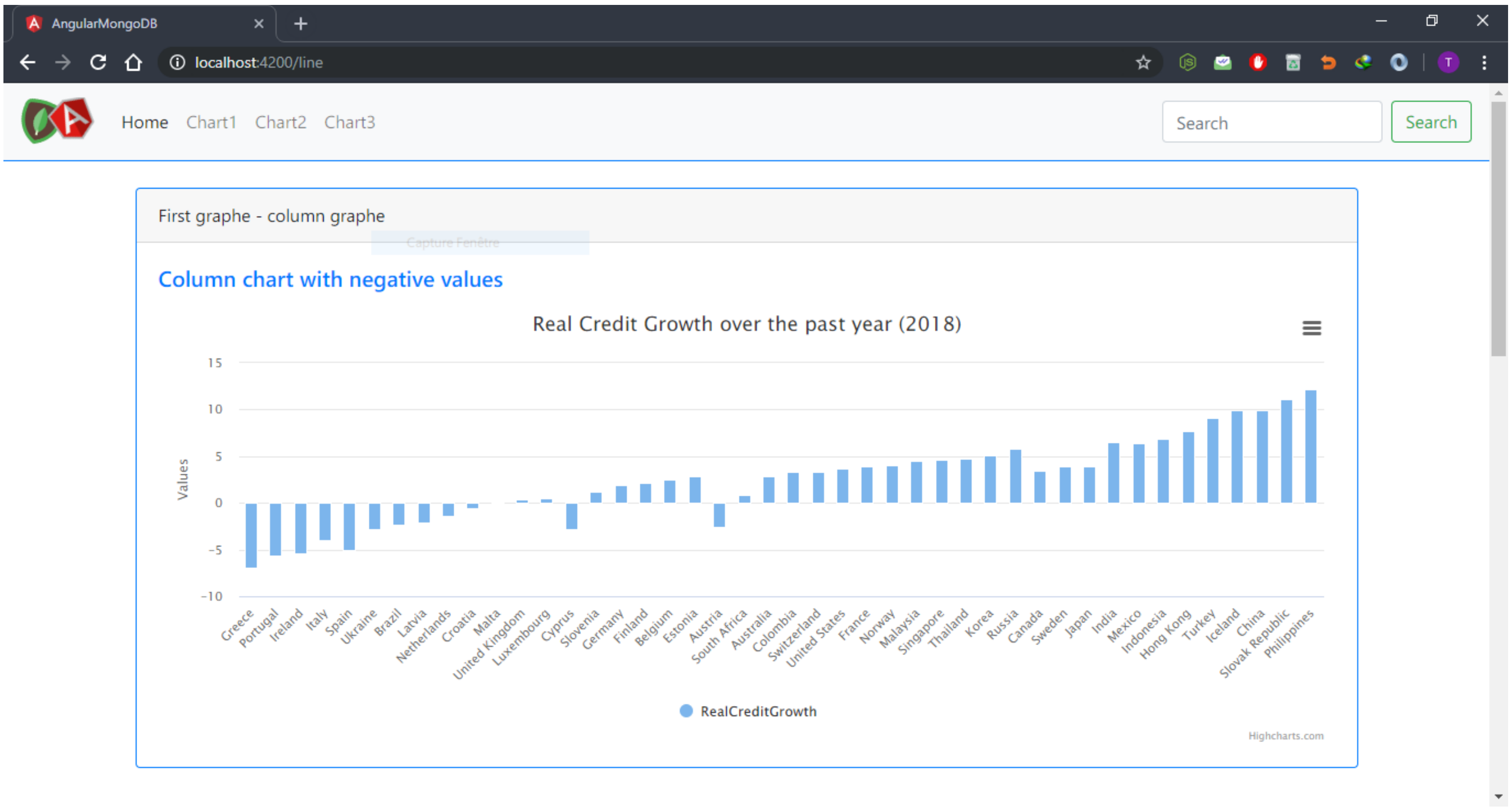
```
<div class="container">
    <br>
    <div class="card border-primary mb-3">
        <div class="card-header">Header</div>
        <div class="card-body text-primary">
            <h5 class="card-title">Primary card title</h5>
            <highcharts-chart [Highcharts]="Highcharts" [options]="chartOptions" style="widt
        </div>
    </div>
    <br>
    <div class="card border-primary mb-3">
        <div class="card-header">Header</div>
        <div class="card-body text-primary">
            <h5 class="card-title">Primary card title</h5>
            <highcharts-chart [Highcharts]="Highcharts1" [options]="chartOptions1" style="wi
        </div>
    </div>
    <br>
    <div class="card border-primary mb-3">
        <div class="card-header">Header</div>
        <div class="card-body text-primary">
            <h5 class="card-title">Primary card title</h5>
            <highcharts-chart [Highcharts]="Highcharts2" [options]="chartOptions2" style="wi
        </div>
    </div>
```

for the HTML we will use the tags

<highcharts-char [Hightcharts]="Highcharts" [options]="chartOptions"></Highcharts-char>

To communicate the html file with the typescript file.

The result chart:



AngularMongoDB

localhost:4200/line

Home Chart1 Chart2 Chart3

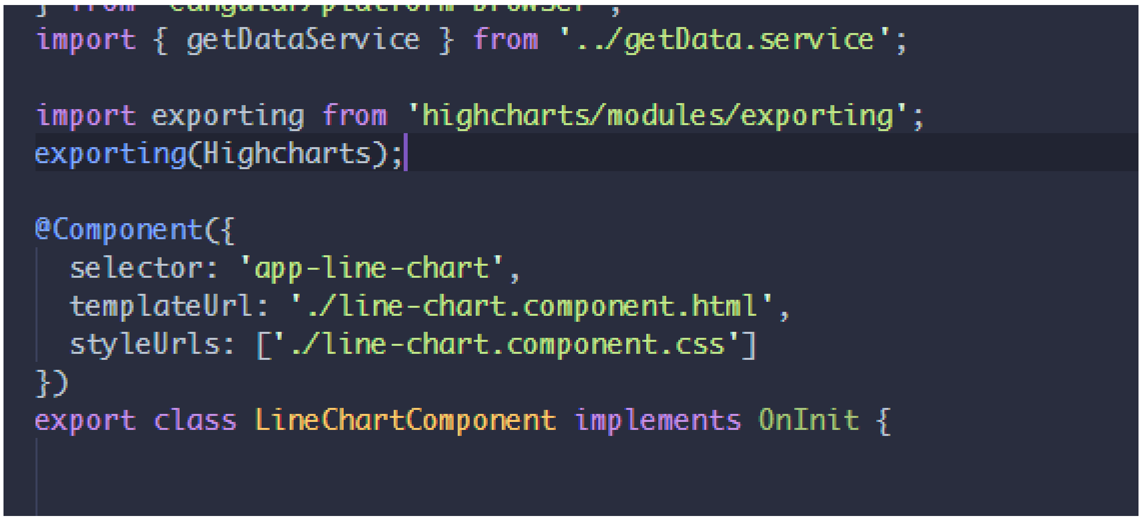
Search

First graphe - column graphe

Column chart with negative values

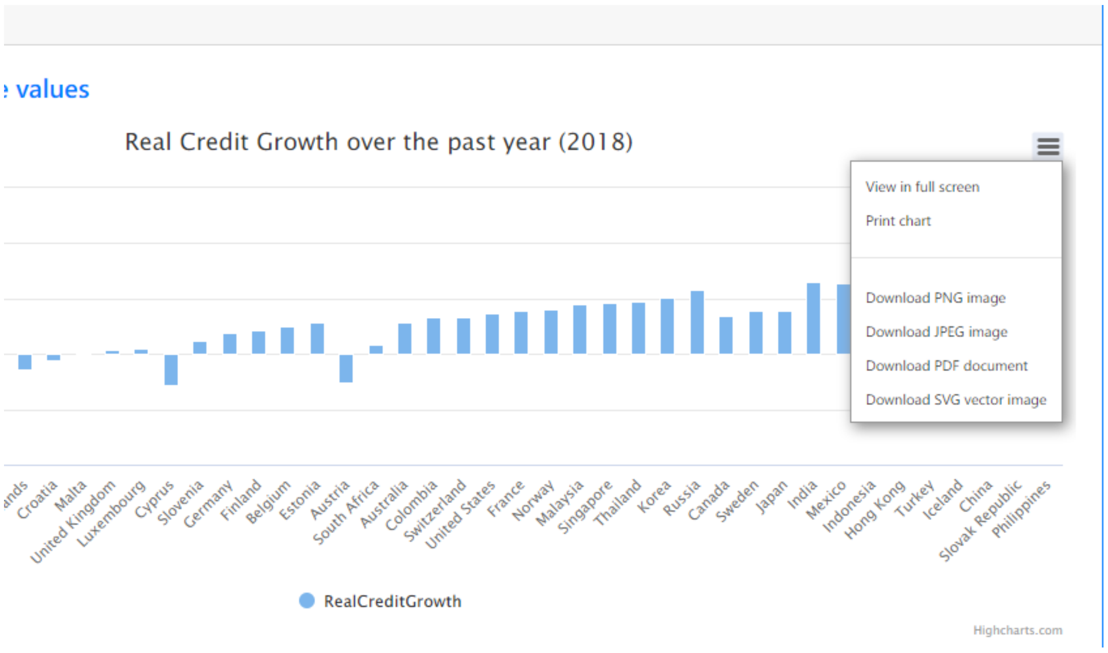
N.B:

To use the export functionality that already defined on Highcharts library to export the charts in different format (PNG, PDF ....), we just need to call the exporting library of Hightcharts in our components as follows



```
import { getDataService } from '../getData.service';

import exporting from 'highcharts/modules/exporting';
exporting(Highcharts);

@Component({
  selector: 'app-line-chart',
  templateUrl: './line-chart.component.html',
  styleUrls: ['./line-chart.component.css']
})
export class LineChartComponent implements OnInit {
```

The result will be as follows

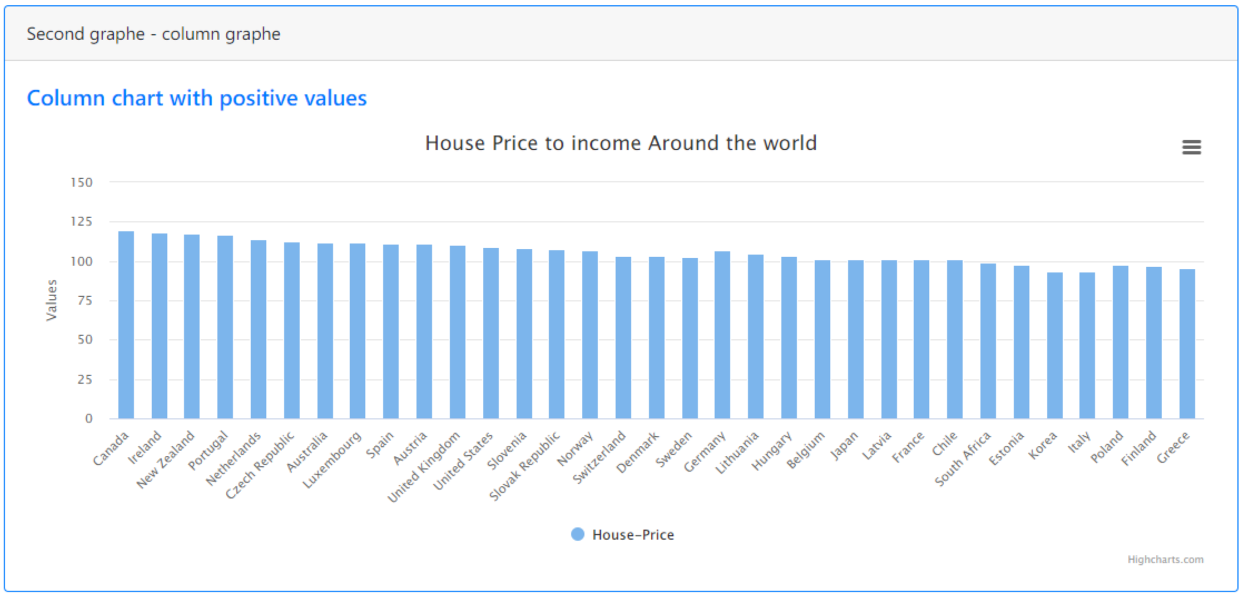


5. Examples of charts created

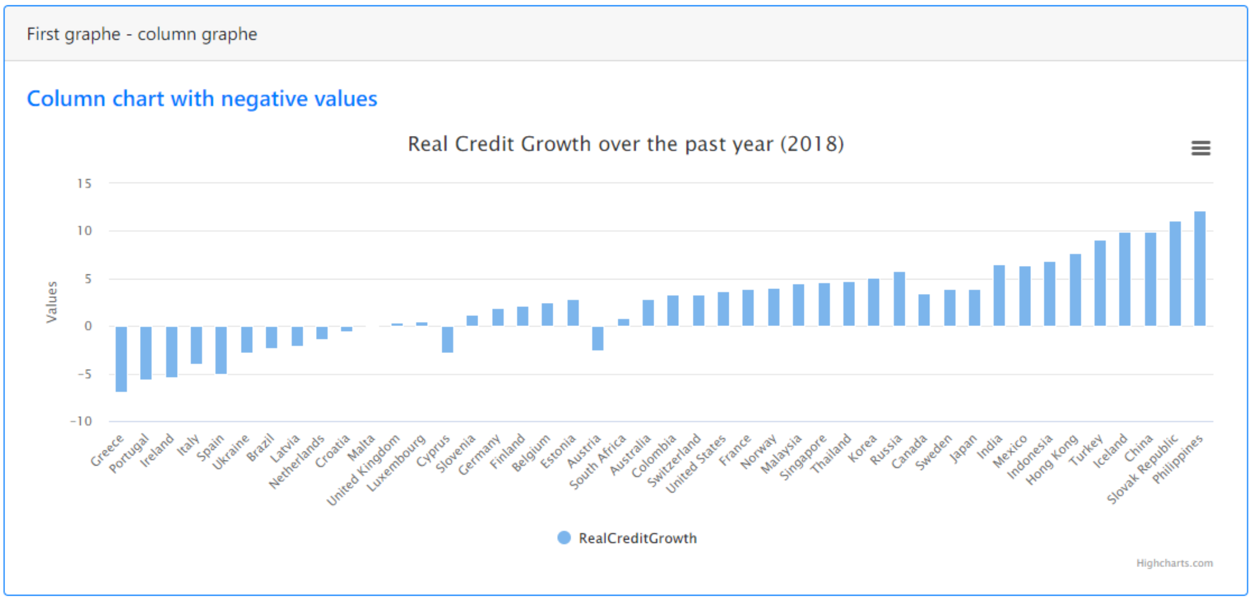
in this section we will display some of the charts created on this project, as it's known Highcharts represent a large amount of charts types, we will show just some of these charts.

5.1 Column charts

5.1.1 with positive value :



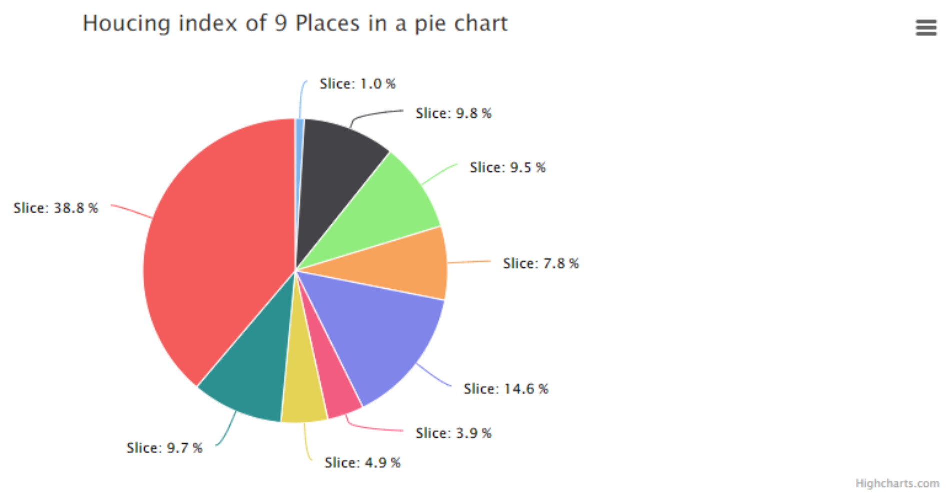
5.1.2 With negative value :



5.2 Pie charts

chart

Pie chart with data labled

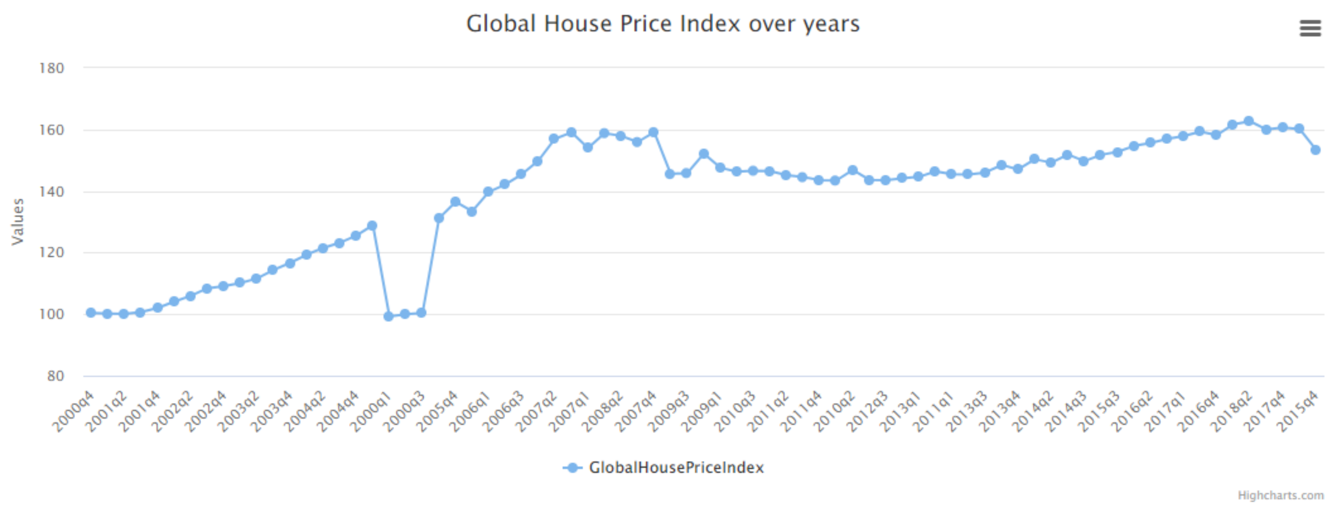


5.3 Linear charts

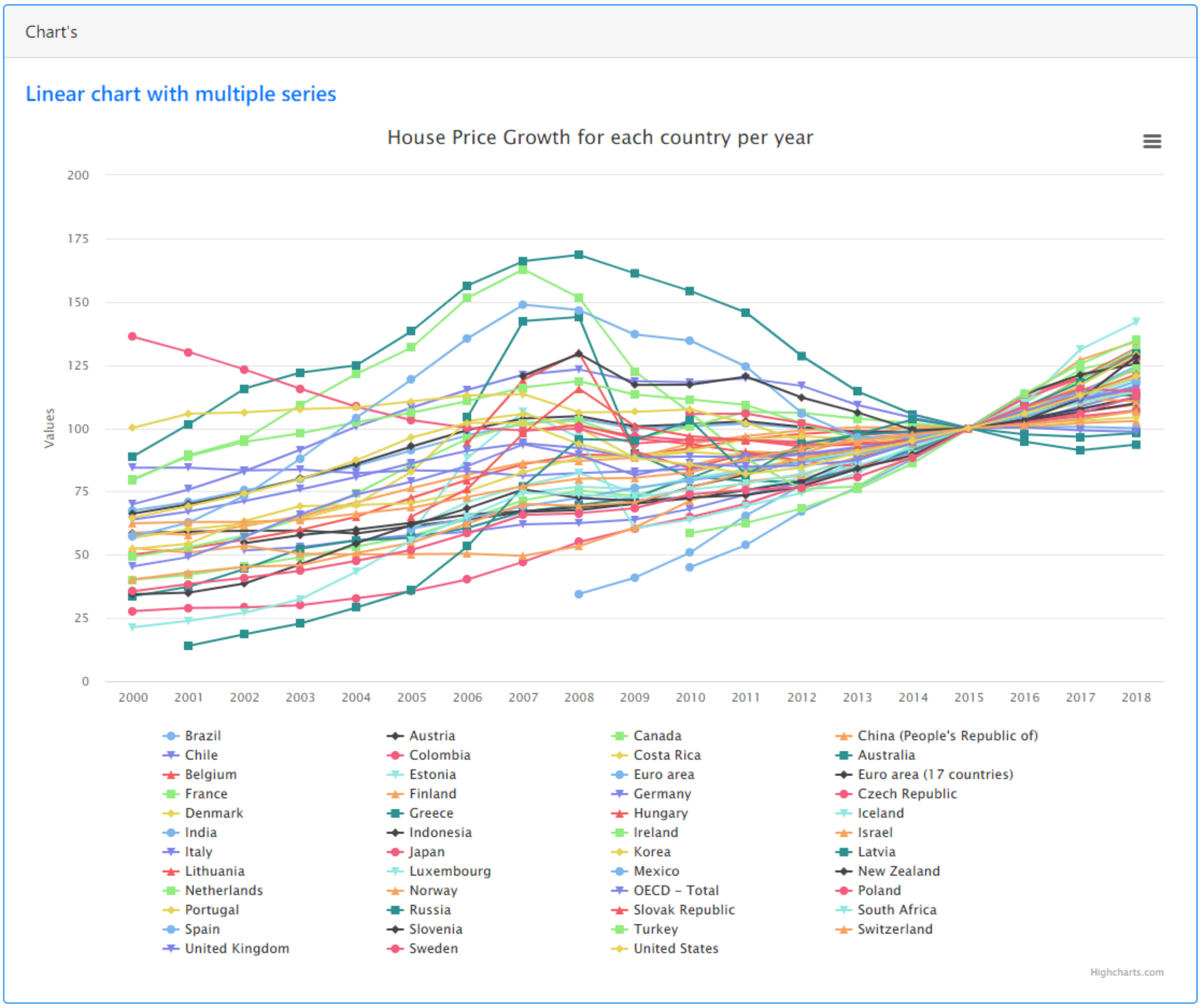
5.3.1 With one series:

chart

Simple linear chart



5.3.2 With multiple series:


Chart's

Linear chart with multiple series

Conclusion:

Angular is one of the most powerful open source frameworks. It is an “all-in-one” solution that aims to provide developers with every possible option out of the box. From Routing to HTTP requests handling and the adoption of TypeScript.