



UNIVERSITE ABDELMALEK ESSAADI  
FACULTE DES SCIENCES ET TECHNIQUES  
DE TANGER



DEPARTEMENT GENIE INFORMATIQUE

**Cycle Master: SIM (Semestre III)**

**Module : Architectures distribuées**

## **Rapport de Projet :**

# Réalisation d'une Application Web distribuée basée sur l'architecture micro-services

---

**Réalisé Par :**

EL MSAOURI Tarik

EL RHARROUBI Mohamed Amine

**Encadré Par :**

Pr. Lotfi EL AACHAK

**Année Universitaire 2021/2022**

---

---

## Table des Matières

Table des Matières.....	1
I. Introduction .....	3
II. Objectif.....	3
III. Les Technologies Utilisées.....	4
1. Jupyter notebook .....	4
2. Spring boot .....	5
3. MongoDB.....	5
4. Angular .....	6
5. Flutter .....	6
IV. Réalisation du projet.....	6
1. Méthodologie de Travail .....	6
2. Data Scaping .....	6
3. Nettoyage des données .....	6
4. Exportation des données vers MongoDB .....	6
5. Démarrage du serveur Spring boot .....	7
6. Partie Administration (Angular) .....	7
7. Partie Virtualisation (Flutter) .....	7
V. Conclusion.....	8

---

## I. Introduction

Les microservices sont un style d'architecture utilisé par de nombreuses organisations pour le développement de logiciels. Par le passé, l'industrie informatique utilisait des solutions monolithiques ou basées sur l'architecture orientée services (ou SOA pour Service-Oriented Architecture) comme standard.

Cependant, le manque d'évolutivité dynamique de ce genre de système architectural n'était plus adapté à la complexité croissante des infrastructures actuelles. C'est là qu'intervient le microservice qui est finalement est une évolution logique du système SOA conçu pour atteindre un degré élevé d'agilité, de distribution rapide et d'évolutivité.

Le besoin de méthodes d'indexation et de recherche directement basée sur le contenu de l'image n'est donc plus à démontrer. Le premier prototype de système a été proposé en 1970 et ce système a attiré l'attention de beaucoup de chercheurs. Quelques systèmes deviennent des systèmes commerciaux tels que QBIC (Query By Image Content), CIREs (Content Based Image Retrieval System), ...

Dans notre projet on a utilisé plusieurs technologies par exemple Spring boot Angular MongoDB Flutter etc.

## II. Objectif

L'objectif principal du projet est la réalisation d'une application web distribuée basée sur la Technologie Spring architecture micro-services, et flutter pour l'analyse, la visualisation et le Clustering des données.

La mise en place d'une architecture distribuée basée sur des micro-services consommables depuis une application mobile hybride flutter, et une application d'administration Angular « SPA ».

L'application mobile doit afficher les données sous forme des graphes, des mini rapport statistique « Data Analysis, Data Visualization & Clustering », donc les données doivent être enregistrer dans une BDD Nosql « MongoDB », puis traitées et visualisées.

Vous pouvez effectuer le paramétrage des propriétés puis charger le Dataset ou bien lancer un processus de Scraping en arrière plan pour collecter les informations depuis plusieurs sites web.

---

### III. Les Technologies Utilisées

#### 1. Jupyter notebook

**Jupyter** est une application web utilisée pour programmer dans plus de 40 langages de programmation, dont Python, Julia, Ruby, R, ou encore Scala<sup>2</sup>. C'est un projet communautaire dont l'objectif est de développer des logiciels libres, des formats ouverts et des services pour l'informatique interactive. Jupyter est une évolution du projet IPython. Jupyter permet de réaliser des calepins ou notebooks, c'est-à-dire des programmes contenant à la fois du texte en markdown et du code. Ces calepins sont utilisés en science des données pour explorer et analyser des données.



#### 2. Spring boot

En informatique, Spring est un framework open source pour construire et définir l'infrastructure d'une application Java<sup>3</sup>, dont il facilite le développement et les tests. En 2004, Rod Johnson a écrit le livre Expert One-on-One J2EE Design and Development<sup>4</sup> qui explique les raisons de la création de Spring.



#### 3. MongoDB

**MongoDB** (de l'anglais *humongous* qui peut être traduit par « énorme ») est un système de gestion de base de données orienté documents, répartitionnable sur un nombre quelconque d'ordinateurs et ne nécessitant pas de schéma prédéfini des données. Il est écrit en C++. Le serveur et les outils sont distribués sous licence SSPL, les pilotes sous licence Apache et la documentation sous licence Creative Commons<sup>4</sup>. Il fait partie de la mouvance NoSQL.



---

## 4. Angular

**Angular** (communément appelé "**Angular 2+**" ou "**Angular v2 et plus**")<sup>2,3</sup> est un framework côté client, open source, basé sur TypeScript, et co-dirigé par l'équipe du projet « Angular » à Google et par une communauté de particuliers et de sociétés. Angular est une réécriture complète d'AngularJS, cadriciel construit par la même équipe. Il permet la création d'applications Web et plus particulièrement de ce qu'on appelle des « *Single Page Applications* » : des applications web accessibles via une page web unique qui permet de fluidifier l'expérience utilisateur et d'éviter les chargements de pages à chaque nouvelle action. Le Framework est basé sur une architecture du type MVC et permet donc de séparer les données, le visuel et les actions pour une meilleure gestion des responsabilités. Un type d'architecture qui a largement fait ses preuves et qui permet une forte maintenabilité et une amélioration du travail collaboratif.



## 5. Flutter

**Flutter** est un kit de développement logiciel (SDK) d'interface utilisateur open-source créé par Google. Il est utilisé pour développer des applications pour Android, iOS, Linux, Mac, Windows, Google Fuchsia et le web à partir d'une seule base de code.

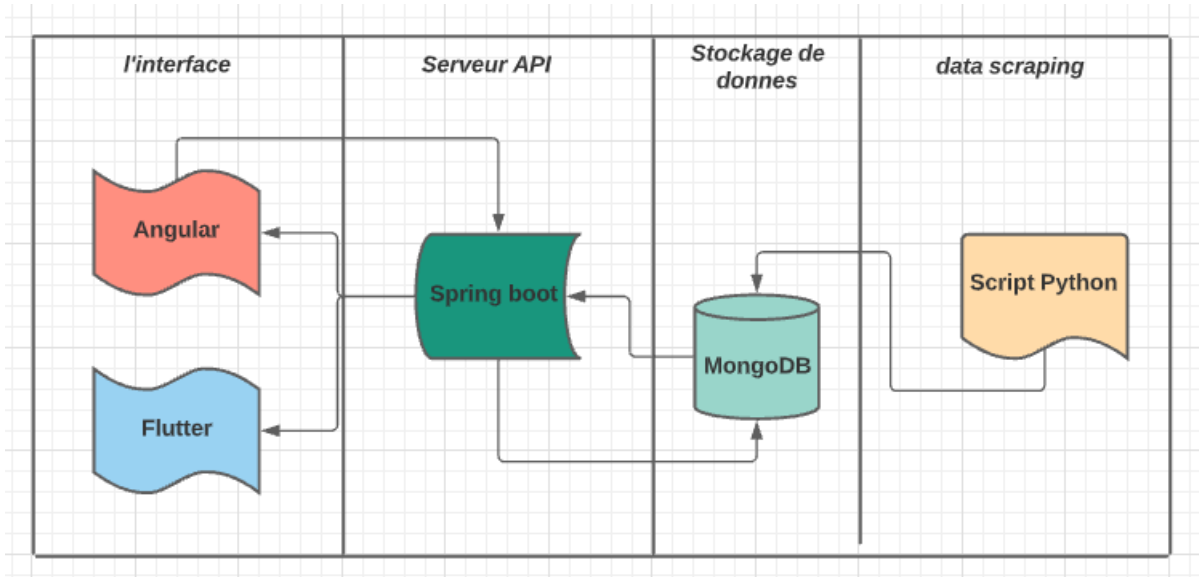
La première version de Flutter était connue sous le nom de code "Sky" et fonctionnait sur le système d'exploitation Android. Elle a été dévoilée lors du sommet des développeurs Dart de 2015, avec l'intention déclarée de pouvoir effectuer un rendu cohérent à 120 images par seconde. Lors du discours d'ouverture des Google Developer Days à Shanghai, Google a annoncé la sortie de Flutter Release Preview 2 qui est la dernière grande version avant Flutter 1.0. Le 4 décembre 2018, Flutter 1.0 a été publié lors de l'événement Flutter Live, ce qui représente la première version "stable" du Framework. Le 11 décembre 2019, Flutter 1.12 a été publié lors de l'événement Flutter Interactive.



# I. Réalisation du projet

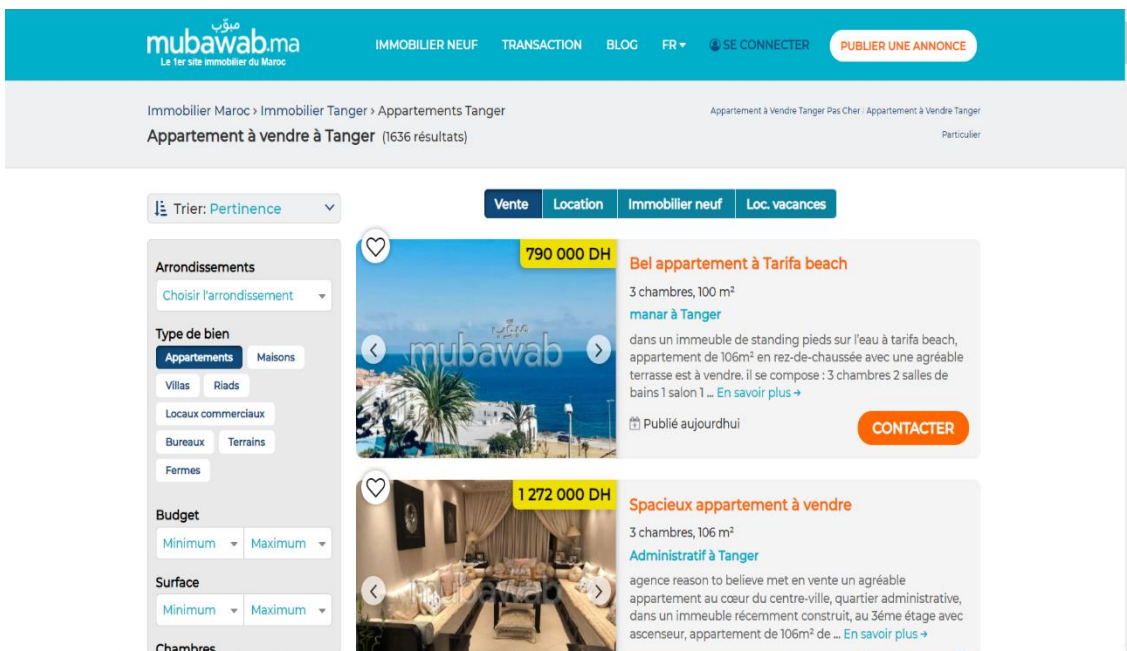
## 1. Méthodologie de Travail.

D'après le site web **www.mubawab.ma** on a scrappé les données c'est données est nettoyer avec **python** puis exporter vers la base de données **MongoDB** on démarre un serveur Spring boot pour communiquer notre donnée vers la cotée administration **Angular** et pour la virtualisation on **Flutter**.



## 2. Data Scaping.

Ouvrons le site **www.mubawab.ma**.



## On va scrapper les donnes de Tanger avec le Script **Mubawab\_Scrapping.ipynb**

(Scrapping)

Site: <https://www.mubawab.ma> # Ville : Tanger

```
Entrée [2]: from urllib.request import urlopen
from bs4 import BeautifulSoup
import datetime
import random
import re
import pandas as pd
import numpy as np

def Get_detail_mubawab_ma(URL):
    html = urlopen(URL)
    bs_ = BeautifulSoup(html, 'html.parser')
    title = bs_.find('h1', {'class': 'searchTitle'}).get_text().strip()
    try:
        price = bs_.find('h3', {'class': 'orangeTit'}).get_text().strip()
    except AttributeError:
        price = np.nan

    localisation = bs_.find('h3', {'class': 'greyTit'}).get_text().strip()
    size, piece, nb_chambre, salles_de_bains, etat, old, etage, Caracteristiques_supplementaires = ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '
    try:
        desc = bs_.findAll('span', {'class': 'tagProp'})
        if desc[0].get_text().split()[1] == "m²":
            size = desc[0].get_text().strip()
        else:
            size = np.nan
        if desc[1].get_text().split()[1] == "Pièces" or desc[1].get_text().split()[1] == "Pièce":
            piece = desc[1].get_text().strip()
        else:
            piece = np.nan
        if desc[2].get_text().split()[1] == "Chambres" or desc[2].get_text().split()[1] == "Chambre":
            nb_chambre = desc[2].get_text().strip()
        else:
            nb_chambre = np.nan
        if desc[3].get_text().split()[2] == "de":
            salles_de_bains = desc[3].get_text().strip()
        else:
            salles_de_bains = np.nan
        etat = desc[4].get_text().strip()
        liste = desc[5].get_text().split()
```

### 3. Nettoyage des donnes.

La preprocessing and data cleaning de notre donnes ce fait avec le script **Mubawab\_cleaning.ipynb**.

Cleaning

site: <https://www.mubawab.ma>

Import Libraries

```
Entrée [54]: import pandas as pd
```

```
Entrée [55]: data = pd.read_excel('MubawabMa.xlsx')
data.head()
```

```
Out[55]:
```

	Titre	Price	Localisation	Size	Nb_pieces	Nb_chambre	Nb_Salles_bain	Etat	Old	Etage	Caracteristi
0	Appartement Meublé à Louer Place Mozart Tanger	10 000 DH	à l'intersection de la route de Tanger	125 m²	4 Pièces	2 Chambres	2 Salles de bains	Nouveau	1-5 ans	6ème étage	[Meublé]GarageAsc
1	APPART MEUBLE DE LUXE	4 500 DH	Moujahidine à l'intersection de la route de Tanger	65 m²	3 Pièces	2 Chambres	1 Salle de bain	Nouveau	NaN	1er étage	[Meublé]Ascenseur
2	Appartement meublé à la location longue durée	13 000 DH	Malabata à l'intersection de la route de Tanger	124 m²	5 Pièces	2 Chambres	2 Salles de bains	Bon état	NaN	NaN	[Meublé]Terrasse[Ga

Le résultat final.

```
Entrée [66]: data.head()
```

```
Out[66]:
```

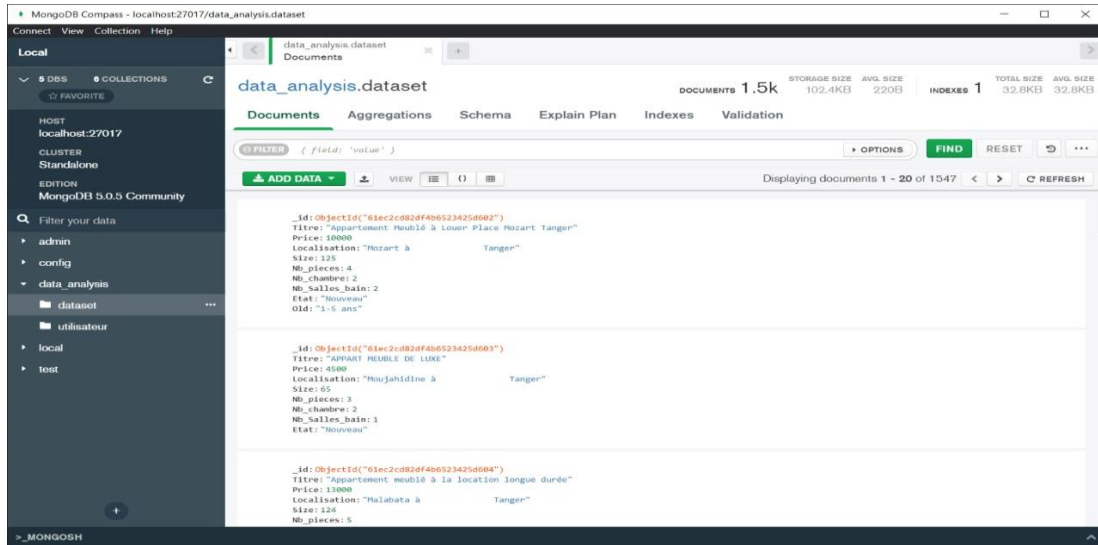
	Titre	Price	Localisation	Size	Nb_pieces	Nb_chambre	Nb_Salles_bain	Etat	Old
0	Appartement Meublé à Louer Place Mozart Tanger	10000.0	Mozart à Tanger	125.0	4.0	2.0	2.0	Nouveau	1-5 ans
1	APPART MEUBLE DE LUXE	4500.0	Moujahidine à Tanger	65.0	3.0	2.0	1.0	Nouveau	NaN
2	Appartement meublé à la location longue durée	13000.0	Malabata à Tanger	124.0	5.0	2.0	2.0	Bon état	NaN
3	Appartement Meublé à Louer – Place Mozart – Ta...	5000.0	Mozart à Tanger	80.0	2.0	2.0	1.0	Bon état	5-10 ans
4	Coquet Appartement à la location Malabata	11500.0	Malabata à Tanger	120.0	4.0	2.0	NaN	NaN	NaN

Exporter le fichier nettoyé avec .to\_excel()

```
Entrée [67]: data.to_excel('MubawabMa_clean.xlsx', index=False, encoding='utf-8')
```

## 4. Exportation des données vers MongoDB.

Les données sont transformées du format CSV vers JSON.



## 5. Démarrage du serveur Spring boot.

On a créé un class **utilisateur**.

```
1 package net.mongo.api.model;
2 import java.util.UUID;
10 @Getter
11 @Setter
12 @ToString
13 @Document(collection="utilisateur")
14 public class Utilisateur {
15     @Id
16     private String id;
17     private String nom;
18     private String username;
19     private String email;
20     private String pass;
21     private boolean admin;
22 }
```

Puis **utilisateurRepository**

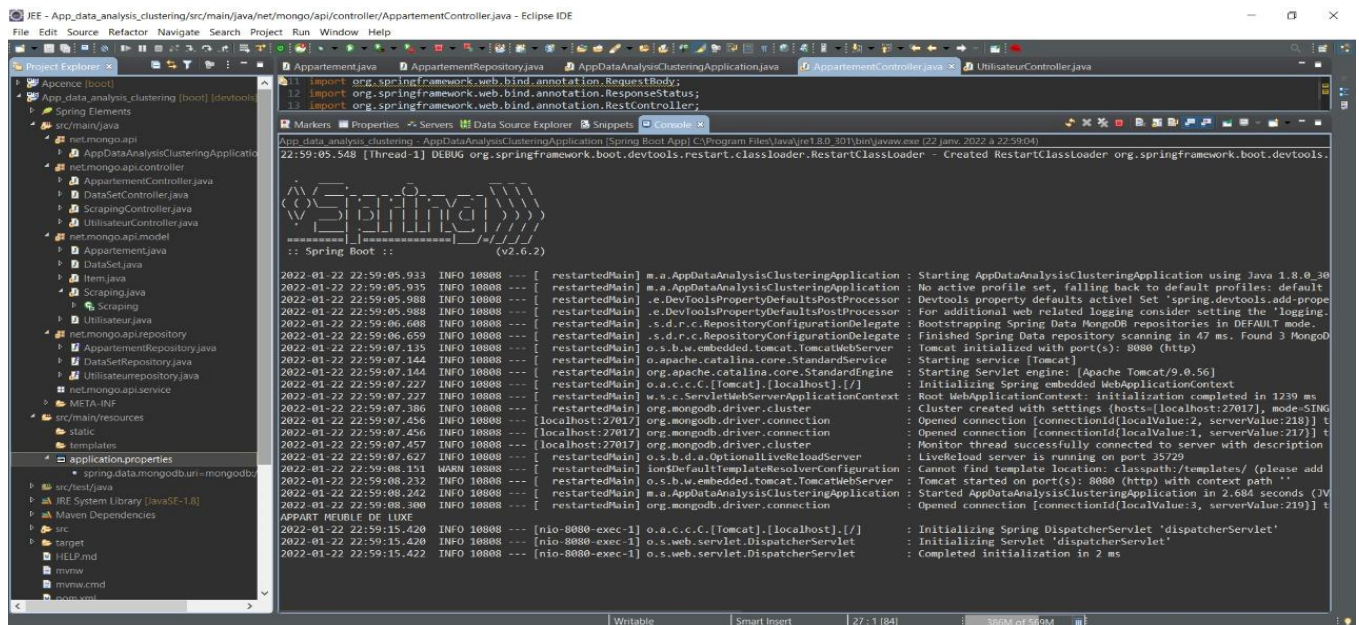
```
Appartement.java AppartementRepository.java AppDataAnalysisSystem AppartementController.java
1 package net.mongo.api.repository;
2 import java.util.Optional;
7
8 public interface UtilisateurRepository extends MongoRepository<Utilisateur,String> {
9
10
11 }
12 }
```



## Le RestController avec des methode @GetMapping @PostMapping @DeleteMapping ...

```
29 @CrossOrigin
30 @RestController
31 public class UtilisateurController {
32     @Autowired
33     private UtilisateurRepository repository;
34
35     @CrossOrigin(origins = "http://localhost:4200")
36     @ResponseStatus(HttpStatus.CREATED)
37     @PostMapping("/addUtilisateur")
38     public Utilisateur saveUtilisateur(@RequestBody Utilisateur u) {
39
40         return (repository.save(u));
41     }
42
43     @ResponseStatus(HttpStatus.CREATED)
44     @PostMapping("/login")
45     public Utilisateur log(@RequestBody Utilisateur utilisateur) {
46         Utilisateur a = new Utilisateur();
47         List<Utilisateur> users = repository.findAll();
48         a = utilisateur;
49         for (Utilisateur user : users) {
50             if(a.getPass().equals(user.getPass()) && a.getEmail().equals(user.getEmail()) ) {
51                 return user;
52             }
53         }
54         return a;
55     }
56
57     @GetMapping("/findAllUtilisateurs")
58     public List<Utilisateur> getUtilisateurs(){
59         return repository.findAll();
60     }
61
62     @GetMapping("/findAll")
63     public String getdata(){
64         return "OK";
65     }
66 }
```

## Finalement on démarre notre Serveur Spring boot.



The screenshot shows the Eclipse IDE interface with the Spring Boot application running. The console output displays the Spring Boot logo and various startup logs, including the following information:

- Spring Boot (v2.6.2)
- Restarted Main: m.a.AppDataAnalysisClusteringApplication : Starting AppDataAnalysisClusteringApplication using Java 1.8.0\_30
- Restarted Main: m.a.AppDataAnalysisClusteringApplication : No active profile set, falling back to default profiles: default
- Restarted Main: e.DevToolsPropertyDefaultsPostProcessor : DevTools property defaults active! Set 'spring.devtools.add-props' for additional web related logging consider setting the 'logging'.
- Restarted Main: s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data MongoDB repositories in DEFAULT mode.
- Restarted Main: s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 47 ms. Found 3 MongoDB
- Restarted Main: o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
- Restarted Main: org.apache.catalina.core.StandardService : Starting service [Tomcat]
- Restarted Main: org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.56]
- Restarted Main: o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
- Restarted Main: Root WebApplicationContext: initialization completed in 1239 ms
- Restarted Main: Cluster created with settings (hosts=[localhost:27017], mode=SHARDING)
- Restarted Main: org.mongodb.driver.connection : Opened connection [connectionId{localValue:2, serverValue:218}] t
- Restarted Main: org.mongodb.driver.connection : Opened connection [connectionId{localValue:1, serverValue:217}] t
- Restarted Main: o.s.b.d.a.OptionalLiveReloadServer : Monitor thread successfully connected to server with description
- Restarted Main: org.mongodb.driver.connection : LiveReload server is running on port 35729
- Restarted Main: org.mongodb.driver.connection : Cannot find template location: classpath:/templates/ (please add
- Restarted Main: m.a.AppDataAnalysisClusteringApplication : Started AppDataAnalysisClusteringApplication in 2.684 seconds (VM
- Restarted Main: org.mongodb.driver.connection : Opened connection [connectionId{localValue:3, serverValue:219}] t
- Restarted Main: o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
- Restarted Main: o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
- Restarted Main: o.s.web.servlet.DispatcherServlet : Completed initialization in 2 ms

## Création des class pour la communication des données

```

}
@ResponseStatus(HttpStatus.CREATED)
@GetMapping("/FlutterDataPrices")
public List<HashMap> getData(){
    HashMap<String, Double> prices = new HashMap<String, Double>();
    List<HashMap> list=new ArrayList<HashMap>();
    List<String> adresses=new ArrayList<String>();
    adresses.add("Tanger");
    adresses.add("Centre à          Tanger");
    adresses.add("Tanger City Center à          Tanger");
    adresses.add("Médina à          Tanger");
    adresses.add("Administratif à          Tanger");
    adresses.add("De La Plage à          Tanger");
    adresses.add("Marjane à          Tanger");
    adresses.add("Mozart à          Tanger");
    adresses.add("Malabata à          Tanger");
    int i=0;
    for(String ad: adresses) {
        ArrayList<Double> pieceTanger = new ArrayList<Double>();
        List<Appartement> apps=rep.findAll();
        for (Appartement ap : apps) {
            if(ap.getLocalisation().equals(ad)) {
                pieceTanger.add(ap.getPrice());
            }
        }
        Double som=0.0;
        for(Double p: pieceTanger) {
            som=som+p;
        }
        i++;
        prices.put("item"+i,som/pieceTanger.size());
    }
    list.add(prices);

    //prices.clear();
    return list;
}
}
```

## Et le RestControler

```


26 public class AppartementController {
27     @Autowired
28     private AppartementRepository rep;
29     @ResponseStatus(HttpStatus.CREATED)
30     @GetMapping("/FlutterGetAll")
31     public List<Appartement> getAllAppartement(){
32         return rep.findAll();
33     }
34
35     @ResponseStatus(HttpStatus.CREATED)
36     @GetMapping("/FlutterTest")
37     public List<HashMap> getAppartement(){
38         ArrayList<String> aList = new ArrayList<String>();
39         ArrayList<String> pieces = new ArrayList<String>();
40         List<Appartement> apps=rep.findAll();
41         for (Appartement ap : apps) {
42             String s=ap.getLocalisation();
43             aList.add(s);
44         }
45     }
}
```

## 6. Partie Administration (Angular).

Premièrement la création d'interface **Angular**

Toggle Sidebar

Login



Email

admin@gmail.com

Password

...

Mot de passe oublié?

Login

[Vous n'avez pas de compte ? Register](#)

La consommation de serveur API pour les utilisateurs

```
app.service.ts x
src > app > app.service.ts > AppService > constructor
1  import { HttpClient } from '@angular/common/http';
2  import { Injectable } from '@angular/core';
3  import { Observable } from 'rxjs';
4  import { Utilisateur } from './utilisateur';
5  @Injectable({
6    providedIn: 'root'
7  })
8  export class AppService {
9
10   constructor(
11     private http:HttpClient,
12   ) { }
13   getAll(){
14     return this.http.get(`http://localhost:8080/findAllUtilisateurs`);
15   }
16   update(id: string,utilisateur: any){
17     return this.http.put(`http://localhost:8080/updateUtilisateur/${id}`,utilisateur);
18   }
19   create(Utilisateur: any){
20     return this.http.post(`http://localhost:8080/addUtilisateur`,Utilisateur);
21   }
22   delete(id: string){
23     return this.http.delete(`http://localhost:8080/deleteUtilisateur/${id}`);
24   }
25   DataHtml(url: string){
26     return this.http.put(`http://localhost:8080/datahtml`,url);
27   }
28   Donnee(item: any){
29     return this.http.put(`http://localhost:8080/donne`,item);
30   }
31 }
```

## Composant de Gestion des utilisateurs

```
export class AppService {
  constructor(
    private http:HttpClient,
  ) { }
  getAll(){
    return this.http.get(`http://localhost:8080/findAllUtilisateurs`);
  }
  update(id: string,utilisateur: any){
    return this.http.put(`http://localhost:8080/updateUtilisateur/${id}`,utilisateur);
  }
  create(Utilisateur: any){
    return this.http.post(`http://localhost:8080/addUtilisateur`,Utilisateur);
  }
  delete(id: string){
    return this.http.delete(`http://localhost:8080/deleteUtilisateur/${id}`);
  }

  DataHtml(url: string){
    return this.http.put(`http://localhost:8080/datahtml`,url);
  }
  Donnee(item: any){
    return this.http.put(`http://localhost:8080/donne`,item);
  }

  getData(){
    return this.http.get(`http://localhost:8080/findAllDataSet`);
  }







  Datadelete(id: string){
    return this.http.delete(`http://localhost:8080/deleteDataLine/${id}`);
  }

  Dataedit(id: string,Data: any){
    return this.http.put(`http://localhost:8080/updateData/${id}`,Data);
  }
}
```



### Gestion d'utilisateur

## User Management

### Users

<b>tarik : tarik@gmail.com</b> ✓ Admin	 
<b>Amine@gmail.com : Amine@gmail.com</b> ✓ Admin	 
<b>Testeur : test@gmail.com2</b> ⊖ Utilisateur	 

### Update user

Nom Testeur2	
Username Testeur	
Email test@gmail.com2	
Password ..... 	
<input type="checkbox"/> Admin	<button>Modifier</button>

## Ajouter des donnes

[Toggle Sidebar](#)[Logout](#)[Users management](#)[Data management](#)

### Data Management

Titre <input type="text"/>		Localisation <input type="text"/>	Price <input type="text" value="0"/>
Size <input type="text" value="0"/>	Nombre de chambre <input type="text" value="0"/>	Nombre de salles bain <input type="text" value="0"/>	Etat <input type="text"/>

Add

## Modification des donnes

[Toggle Sidebar](#)[Logout](#)[Users management](#)[Data management](#)

### Data Management

Titre <input type="text" value="Location appartement à IBERIA 5 chambres"/>		Localisation <input type="text" value="Iberie à Tanger"/>	Price <input type="text" value="9800"/>
Size <input type="text" value="205"/>	Nombre de chambre <input type="text" value="5"/>	Nombre de salles bain <input type="text" value="3"/>	Etat <input type="text" value="Bon état"/>

Update

## L'affichage des donnes

[Toggle Sidebar](#)[Logout](#)[Users management](#)[Data management](#)

### Data Management

Titre <input type="text" value="Location appartement à IBERIA 5 chambres"/>		Localisation <input type="text" value="Iberie à Tanger"/>	Price <input type="text" value="9800"/>
Size <input type="text" value="205"/>	Nombre de chambre <input type="text" value="5"/>	Nombre de salles bain <input type="text" value="3"/>	Etat <input type="text" value="Bon état"/>

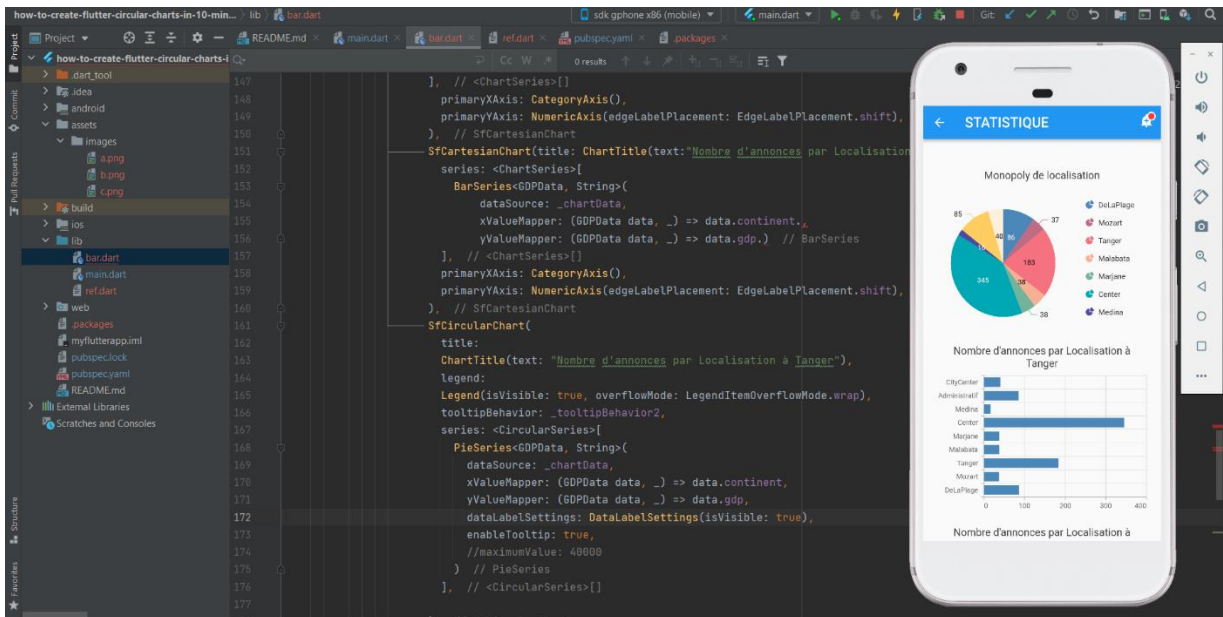
Update

Titre	Localisation	Size	Prix	Nombre de chambre	Nombre de salles bain	Etat	Actions
Studio meublé pour les étudiants"	Boukhalef à Tanger	24 m²	1200 dh	1	1	Nouveau	<div>UpdateDelete</div>
Ben yahia abdellatif"	Msala à Tanger	70 m²	3000 dh	2	1	Bon état	<div>UpdateDelete</div>
Location appartement à IBERIA 5 chambres"	Iberie à Tanger	205 m²	9800 dh	5	3	Bon état	<div>UpdateDelete</div>



## 7. Partie Virtualisation (Flutter).

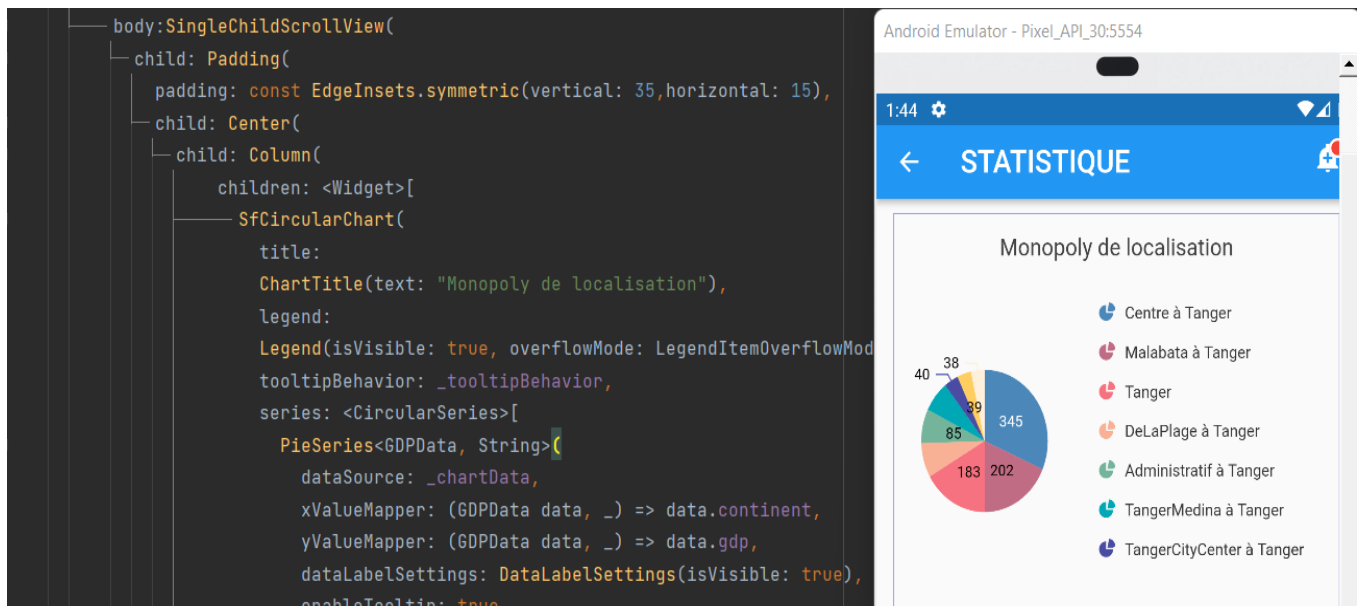
Dans cette partie on a consommé notre API **spring boot** pour générer des graphes.



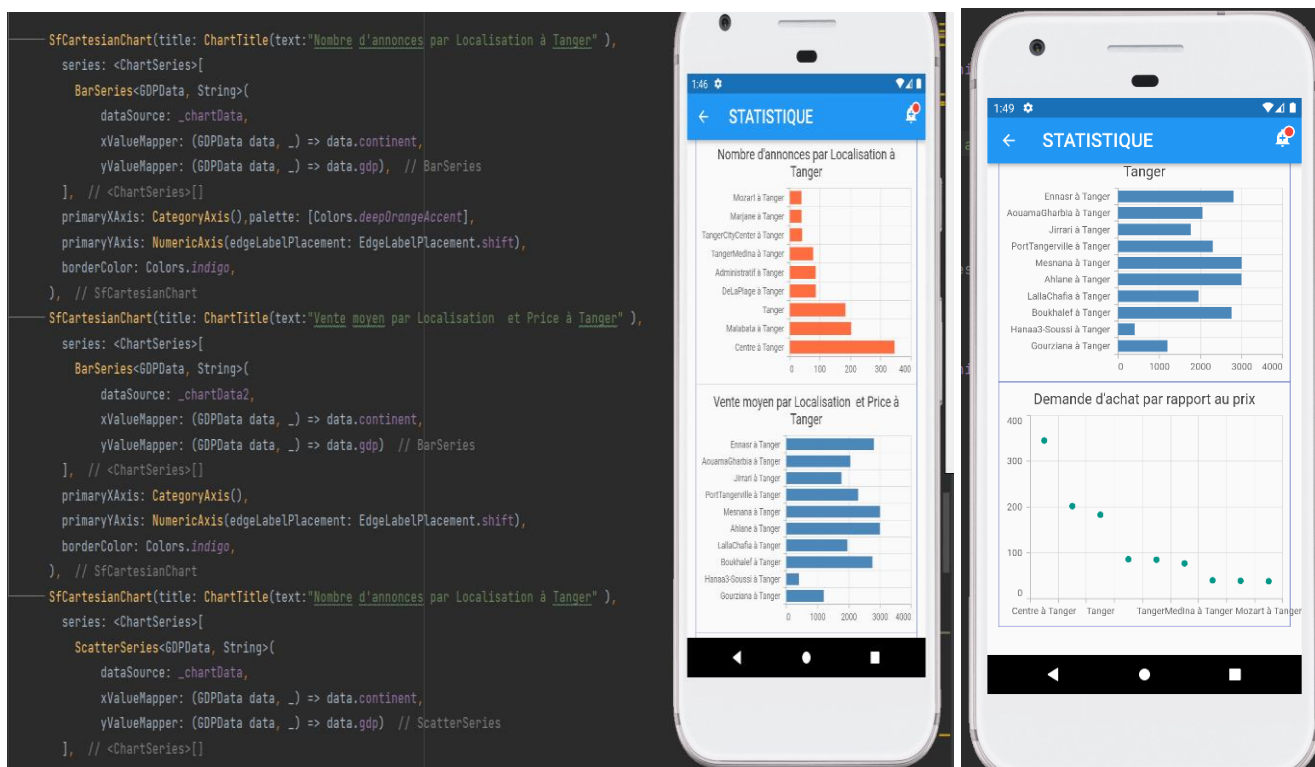
La méthode d'acquière les données à partir Spring boot et notre Graphe.

```
void fetchPoste() async {  
  try{  
    final resp=await get(Uri.parse(url));  
    final jsonData=jsonDecode(resp.body) as List;  
    setState(() {  
      _postsJson=jsonData;  
      _chartData = [  
        GDPData('DeLaPlage', _postsJson[0]['DeLaPlage']),  
        GDPData('Mozart', _postsJson[0]['Mozart']),  
        GDPData('Tanger', _postsJson[0]['Tanger']),  
        GDPData('MaLabata', _postsJson[0]['MaLabata']),  
        GDPData('Marjane', _postsJson[0]['Marjane']),  
        GDPData('Center', _postsJson[0]['Center']),  
        GDPData('Medina', _postsJson[0]['Medina']),  
        GDPData('Administratif', _postsJson[0]['Administratif']),  
        GDPData('CityCenter', _postsJson[0]['CityCenter']),  
      ];  
    });  
  }  
}
```

## Monopoly de localisation



## Nombre d'annonces par Localisation à Tanger Vente moyen par Localisation et Price à Tanger



---

## Conclusion

Ce Projet présente une des plusieurs parties de fonctionnement au niveau de technologie utilisée. En tant que première expérience dans ce domaine, ce projet a été très enrichissant, surtout sur le plan technique. Il nous a permis d'acquérir de nouvelles compétences, dans la découverte, configuration et l'utilisation des différents outils et techniques pour gérer les bases de données NOSQL, création des interfaces mobile et Serveur API. Nous avons eu aussi l'occasion d'approfondir nos connaissances en Spring boot.