



Sagesse Devoir
Ecole Polytechnique Thiès



Sagesse Devoir
Ecole Polytechnique Thiès



République du Sénégal
Un Peuple – Un But – Une Foi
Ministère l'Enseignement Supérieur de la Recherche et
de l'Innovation
École Polytechnique de Thiès
B.P.A 10 Thiès
Tél: (221) 76 223 61 74 – Fax: (221) 33 951 14 67

RAPPORT SUR CLIENT MOBILE

Présenté par :

Mohamed Massamba SENE

Professeur
Dr. Samba Sidibé

Matière
Développement Web 3

Table des matières

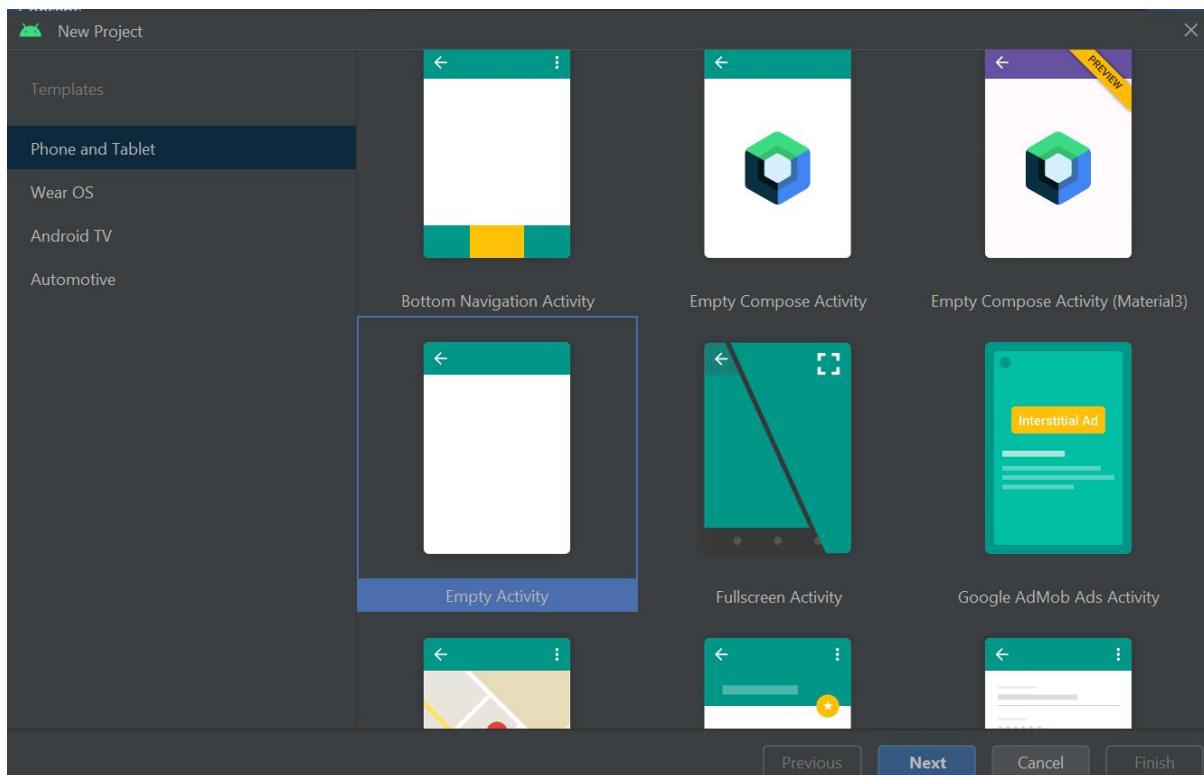
I.	Développement d'un client pour le service web	3
II.	Gestion offline des stocks.....	37
III.	Envoi de notifications	48
IV.	Traçage du téléphone.....	52

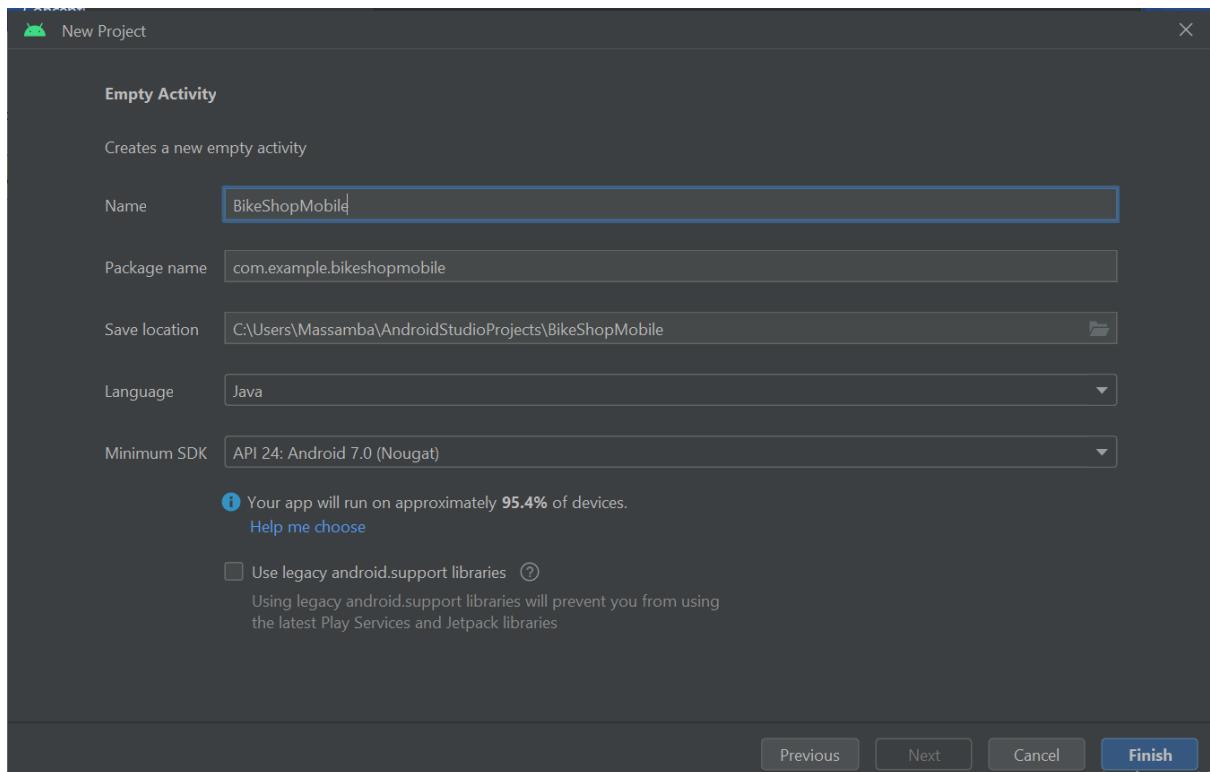
I. Développement d'un client pour le service web

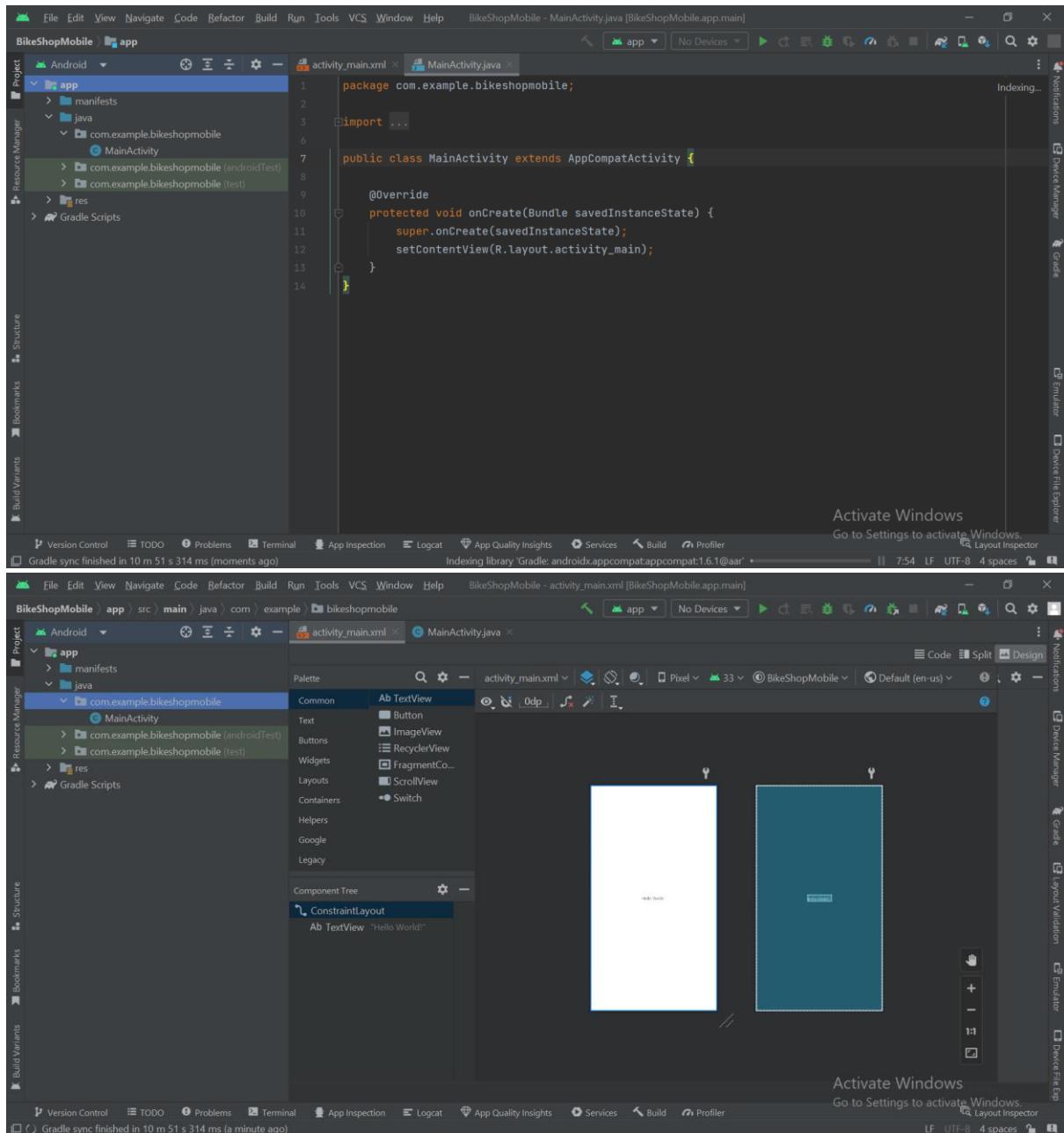
Pour l'application mobile, nous avons décidé de reprendre les fonctionnalités CRUD disponible sur l'application JakartaEE. Ainsi l'utilisateur pourra se connecter et gérer tous les entités disponibles dans la base de données.

Nous ajouterons plus également la possibilité d'envoyer des notifications sur l'état des stocks faibles ainsi que de traquer le téléphone du client en envoyant les données GPS au serveur toutes les 15 minutes.

Création d'un nouveau projet



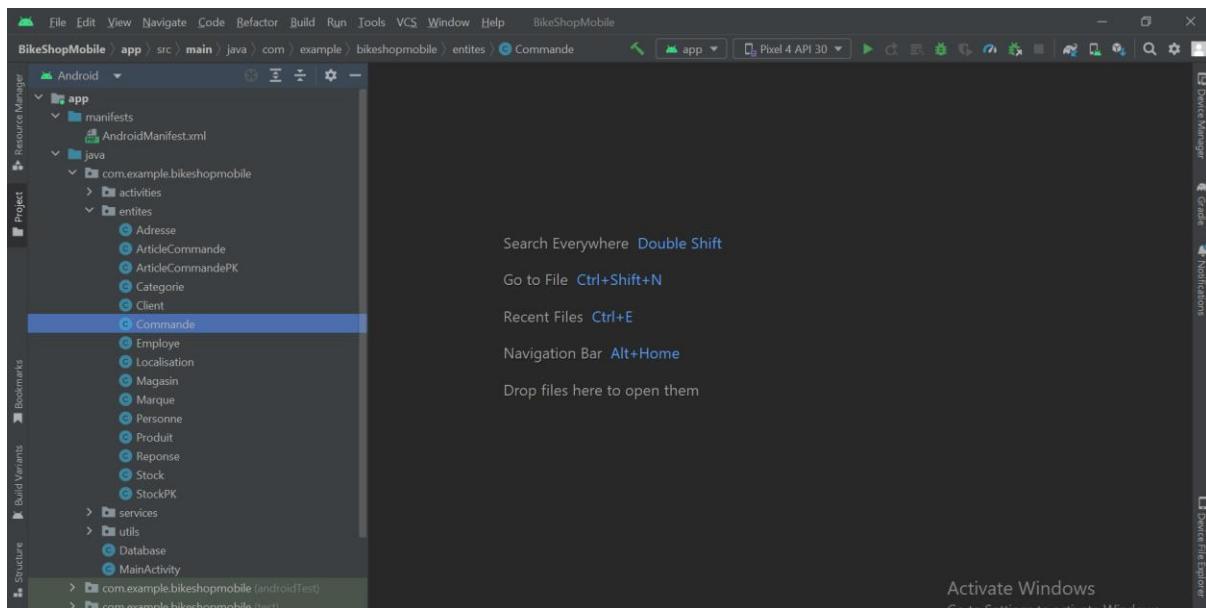




Création des classes pour nos différentes entités

Pour chacune des entités présentes dans notre base de données, nous allons créer une classe Java représentant cette entité. Contrairement au JakartaEE il s'agit ici de tout simplement recréer la classe en renseignant les attributs dont elle dispose, les constructeurs, ... donc nous n'aurons pas à utiliser d'annotations JPA.

Comme nous pouvons le voir ici on crée des classes pour chacune de nos entités



Nous pouvons voir ici la création des classes Commande et Catégorie plus en détails.

Categorie.java

```
1 package com.example.bikeshopmobile.entites;
2
3 import ...
4
5
6 public class Categorie implements Serializable {
7
8     12 usages
9     private Integer id;
10    4 usages
11    private String nom;
12    2 usages
13    private Collection<Produit> produitCollection;
14
15    public Categorie() {
16    }
17
18    public Categorie(Integer id) { this.id = id; }
19
20    public Categorie(Integer id, String nom) {
21        this.id = id;
22        this.nom = nom;
23    }
24
25    public Integer getId() { return id; }
26
27 }
```

Activate Windows
Go to Settings to activate Windows

```
22     }
23
24     public Integer getId() { return id; }
25
26     public void setId(Integer id) { this.id = id; }
27
28     public String getNom() { return nom; }
29
30     public void setNom(String nom) { this.nom = nom; }
31
32     public Collection<Produit> getProduitCollection() { return produitCollection; }
33
34     public void setProduitCollection(Collection<Produit> produitCollection) {
35         this.produitCollection = produitCollection;
36     }
37
38     @Override
39     public int hashCode() {
40         int hash = 0;
41         hash += (id != null ? id.hashCode() : 0);
42         return hash;
43     }
44
45     @Override
46     public boolean equals(Object object) {
47
48         if (this == object)
49             return true;
50
51         if (object instanceof Categorie)
52             return true;
53
54         if (id != null && other.id != null) || (id != null && !id.equals(other.id)))
55             return false;
56
57         if (nom != null && other.nom != null) || (nom != null && !nom.equals(other.nom)))
58             return false;
59
60         return true;
61     }
62
63     @Override
64     public String toString() {
65         return "Categorie{" +
66                 "id=" + id +
67                 ", nom='" + nom +
68                 '}';
69     }
70
71 }
```

Activate Windows
Go to Settings to activate Windows

```
56     public boolean equals(Object object) {
57         if (!(object instanceof Categorie)) {
58             return false;
59         }
60         Categorie other = (Categorie) object;
61         if ((this.id == null && other.id != null) || (this.id != null && !this.id.equals(other.id))) {
62             return false;
63         }
64         return true;
65     }
66
67     @Override
68     public String toString() {
69         return "Categorie{" +
70                 "id=" + id +
71                 ", nom='" + nom +
72                 '}';
73     }
74
75 }
```

Activate Windows
Go to Settings to activate Windows

Commande.java

```
1 package com.example.bikeshopmobile.entities;
2
3 import ...
4
5 public class Commande implements Serializable {
6
7     12 usages
8     private Integer numero;
9     4 usages
10    private short statut;
11    4 usages
12    private String dateCommande;
13    4 usages
14    private String dateLivraisonVoulee;
15    3 usages
16    private String dateLivraison;
17    2 usages
18    private Collection<ArticleCommande> articleCommandeCollection;
19    3 usages
20    private Client clientId;
21    3 usages
22    private Employe vendeurId;
23    3 usages
24    private Magasin magasinId;
```

Activate Windows
Go to Settings to activate Windows

The screenshot shows a Java code editor with the file `Commande.java` open. The code defines a class `Commande` with several methods: `getDateCommande()`, `setDateCommande(String)`, `getDateLivraisonVoule()`, `setDateLivraisonVoule(String)`, `getDateLivraison()`, and `setDateLivraison(String)`. There are also two collection methods: `getArticleCommandeCollection()` and `getCollection()`. The code editor highlights method names in orange and provides usage counts and annotations for each method. The status bar at the bottom right indicates "Activate Windows" and "Go to Settings to activate Windows".

```
1 usage
49  public String getDateCommande() { return dateCommande; }
52
53  2 usages
54  public void setDateCommande(String dateCommande) { this.dateCommande = dateCommande; }
55
56  1 usage
57  public String getDateLivraisonVoule() { return dateLivraisonVoule; }
58
59  2 usages
60  public void setDateLivraisonVoule(String dateLivraisonVoule) {
61      this.dateLivraisonVoule = dateLivraisonVoule;
62  }
63
64  1 usage
65  public String getDateLivraison() { return dateLivraison; }
66
67  2 usages
68  public void setDateLivraison(String dateLivraison) { this.dateLivraison = dateLivraison; }
69
70  public Collection<ArticleCommande> getArticleCommandeCollection() {
71
72      return articleCommandeCollection;
73  }
74
75  public Collection<Commande> getCollection() {
76
77  }
```

The screenshot shows a Java code editor with the file `Commande.java` open. The code defines a class `Commande` with various methods for setting and getting collection, client ID, employee ID, and store ID. It also includes an overridden `hashCode` method. The code editor has a sidebar with navigation links like Device Manager, Grade, Notifications, Device File Explorer, and Device File Explorer. The status bar at the bottom right shows "Activate Windows" and "Go to Settings to activate Windows".

```
77
78     public void setArticleCommandeCollection(Collection<ArticleCommande> articleCommandeCollection) {
79         this.articleCommandeCollection = articleCommandeCollection;
80     }
81
82     3 usages
83     public Client getClientId() { return clientId; }
84
85     2 usages
86     public void setClientId(Client clientId) { this.clientId = clientId; }
87
88     2 usages
89     public Employe getVendeurId() { return vendeurId; }
90
91     2 usages
92     public void setVendeurId(Employe vendeurId) { this.vendeurId = vendeurId; }
93
94     2 usages
95     public Magasin getMagasinId() { return magasinId; }
96
97     public void setMagasinId(Magasin magasinId) { this.magasinId = magasinId; }
98
99
100    @Override
101    public int hashCode() {
102        int hash = 0;
103        hash += (numero != null ? numero.hashCode() : 0);
104    }
105
106
107    6 52 ^
```

```
101     public void setMagasinId(Magasin magasinId) { this.magasinId = magasinId; }
105
106     @Override
107     public int hashCode() {
108         int hash = 0;
109         hash += (numero != null ? numero.hashCode() : 0);
110         return hash;
111     }
112
113     @Override
114     public boolean equals(Object object) {
115         // TODO: Warning - this method won't work in the case the id fields are not set
116         if (!(object instanceof Commande)) {
117             return false;
118         }
119         Commande other = (Commande) object;
120         if ((this.numero == null && other.numero != null) || (this.numero != null && !this.numero.equals(other.numero))) {
121             return false;
122         }
123         return true;
124     }
125
126     @Override
127     public String toString() {
```

```
118     }
119     Commande other = (Commande) object;
120     if ((this.numero == null && other.numero != null) || (this.numero != null && !this.numero.equals(other.numero))) {
121         return false;
122     }
123     return true;
124
125
126     @Override
127     public String toString() {
128
129         return "Commande{" +
130             "numero=" + numero +
131             ", statut=" + statut +
132             ", dateCommande=" + dateCommande +
133             ", dateLivraisonVoulee=" + dateLivraisonVoulee +
134             ", dateLivraison=" + dateLivraison +
135             ", clientId=" + clientId +
136             ", vendeurId=" + vendeurId +
137             ", magasinId=" + magasinId +
138             '}';
139     }
140
141 }
142 }
```

Gestion de la communication avec les services web

Pour récupérer les données et reporter les modifications apportées au niveau du serveur, il nous faut communiquer avec les services web. Nous utilisons donc Retrofit qui est une bibliothèque populaire d'Android Studio facilitant la communication avec les services web.

Nous commençons donc par télécharger les dépendances nécessaires à son utilisation dans le fichier build.gradle de notre application et ajouter la permission pour Internet au niveau du Manifest.

The screenshot shows two tabs in the code editor: `AndroidManifest.xml` and `build.gradle (app)`.

AndroidManifest.xml:

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.POST_NOTIFICATIONS" />
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

    <application
        android:name=".utils.MyApplication"
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="BikeShopMobile"
        android:supportsRtl="true"
        android:theme="@style/Theme.AppCompat.NoActionBar"
        android:usesCleartextTraffic="true"
        tools:targetApi="31">
    
```

build.gradle (app):

```

targetCompatibility JavaVersion.VERSION_1_8

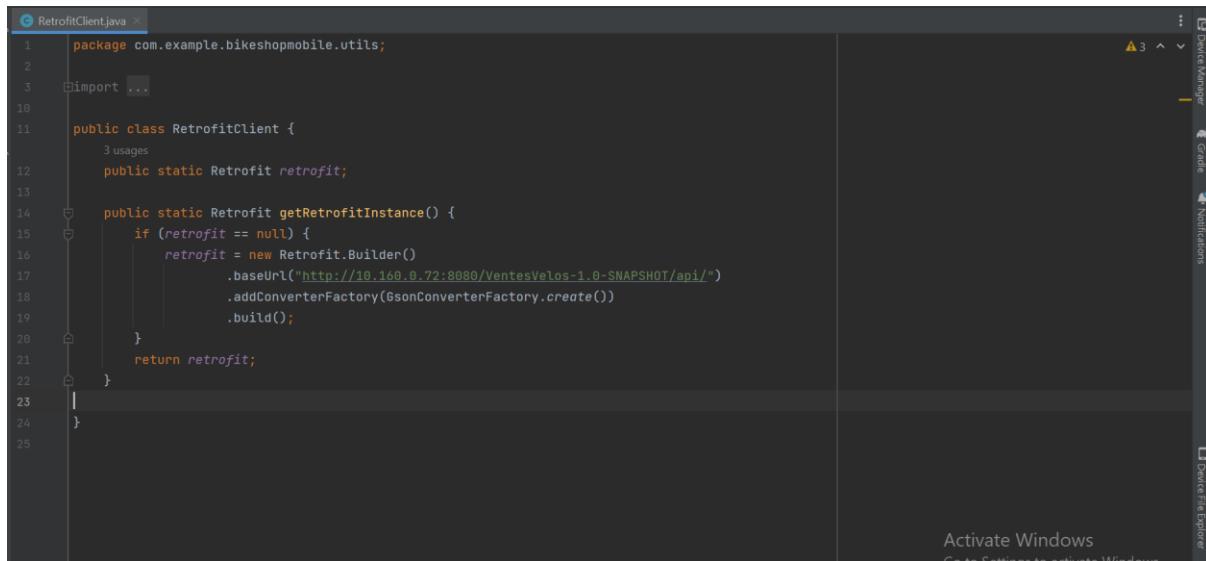
dependencies {
    implementation 'androidx.appcompat:appcompat:1.6.1'
    implementation 'com.google.android.material:material:1.9.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.3'
    implementation 'com.squareup.retrofit2:retrofit:2.9.0'
    implementation 'com.squareup.retrofit2:converter-gson:2.9.0'
    implementation 'androidx.cardview:cardview:1.0.0'
    implementation 'androidx.work:work-runtime:2.8.1'
    implementation 'com.google.android.gms:play-services-location:21.0.1'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.5'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'
}

```

Nous créons ensuite une classe ***RetrofitClient*** qui encapsule la configuration et la création de l'instance de Retrofit dans notre application. Ce qui nous permet de centraliser la configuration pour éviter la duplication du code.

Nous définissons donc une variable statique retrofit qui stocke l'instance de Retrofit une fois créé ainsi qu'une méthode statique `getRetrofitInstance()` qui sera accessible sans instance et nous permettra de vérifier si une instance existe, le cas contraire on en crée une.

Dans le `Retrofit.Builder` nous configurons les aspects de Retrofit comme l'URL de notre service web et le convertisseur avant de retourner l'instance créée.



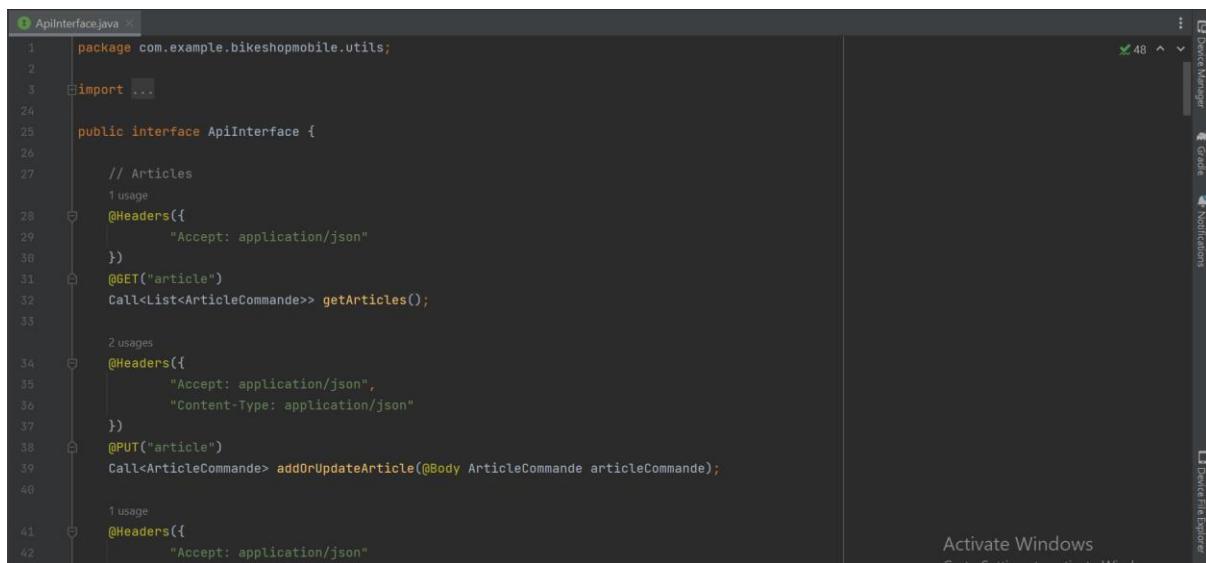
```
1 package com.example.bikeshopmobile.utils;
2
3 import ...
4
5 public class RetrofitClient {
6     3 usages
7     public static Retrofit retrofit;
8
9     public static Retrofit getRetrofitInstance() {
10         if (retrofit == null) {
11             retrofit = new Retrofit.Builder()
12                 .baseUrl("http://10.160.0.72:8080/VentesVélos-1.0-SNAPSHOT/api/")
13                 .addConverterFactory(GsonConverterFactory.create())
14                 .build();
15         }
16         return retrofit;
17     }
18 }
19
20
21
22
23
24
25
```

Activate Windows
Go to Settings to activate Windows

Device Manager Grade Notifications Device File Explorer

Nous créons également une interface **ApiInterface** dans laquelle nous définissons les méthodes correspondantes aux différentes actions que nous pouvons effectuer au niveau de notre service web. Dans notre cas il s'agira de créer des méthodes pour récupérer, supprimer, créer et modifier les entités.

Voici un aperçu de l'interface ainsi créée



```
1 package com.example.bikeshopmobile.utils;
2
3 import ...
4
5 public interface ApiInterface {
6
7     // Articles
8     1 usage
9     @Headers({
10         "Accept: application/json"
11     })
12     @GET("article")
13     Call<List<ArticleCommande>> getArticles();
14
15     2 usages
16     @Headers({
17         "Accept: application/json",
18         "Content-Type: application/json"
19     })
20     @PUT("article")
21     Call<ArticleCommande> addOrUpdateArticle(@Body ArticleCommande articleCommande);
22
23     1 usage
24     @Headers({
25         "Accept: application/json"
26     })
27 }
```

Activate Windows
Go to Settings to activate Windows

Device Manager Grade Notifications Device File Explorer

```

1 ApiInterface.java ✘
40
41     1 usage
42     @Headers({
43         "Accept: application/json"
44     })
45     @DELETE("article/{numero}/{ligne}")
46     Call<Reponse> deleteArticle(@Path("numero") Integer numero, @Path("ligne") Integer ligne);
47
48     // Categories
49     1 usage
50     @Headers({
51         "Accept: application/json"
52     })
53     @GET("categorie")
54     Call<List<Categorie>> getCategories();
55
56     2 usages
57     @Headers({
58         "Accept: application/json",
59         "Content-Type: application/json"
60     })
61     @PUT("categorie")
62     Call<Categorie> addOrUpdateCategorie(@Body Categorie categorie);
63
64     1 usage

```

Activate Windows
Go to Settings to activate Windows

```

1 ApiInterface.java ✘
63
64     @DELETE("categorie/{id}")
65     Call<Reponse> deleteCategorie(@Path("id") Integer id);
66
67     // Clients
68     1 usage
69     @Headers({
70         "Accept: application/json"
71     })
72     @GET("client")
73     Call<List<Client>> getClients();
74
75     2 usages
76     @Headers({
77         "Accept: application/json",
78         "Content-Type: application/json"
79     })
80     @PUT("client")
81     Call<Client> addOrUpdateClient(@Body Client client);
82
83     1 usage
84     @Headers({
85         "Accept: application/json"
86     })
87     @DELETE("client/{id}")

```

Activate Windows
Go to Settings to activate Windows

```

1 ApiInterface.java ✘
87     // Commandes
88     1 usage
89     @Headers({
90         "Accept: application/json"
91     })
92     @GET("commande")
93     Call<List<Commande>> getCommandes();
94
95     2 usages
96     @Headers({
97         "Accept: application/json",
98         "Content-Type: application/json"
99     })
100    @PUT("commande")
101    Call<Commande> addOrUpdateCommande(@Body Commande commande);
102
103    1 usage
104    @Headers({
105        "Accept: application/json"
106    })
107    @DELETE("commande/{numero}")
108    Call<Reponse> deleteCommande(@Path("numero") Integer numero);
109
110    // Employes
111    1 usage

```

Activate Windows
Go to Settings to activate Windows

Pour finir nous créons un service **ApiService** qui étend IntentService qui permet de faciliter l'exécution des tâches en arrière-plan dans un thread dédié.

L'objectif est d'utiliser une instance de RetrofitClient pour effectuer les appels vers le service web dans le but de récupérer les données au niveau des différentes endpoints. Une fois les données récupérées, on utilise un LocalBroadcastManager qui envoie des messages de diffusion locaux afin de notifier l'activité concernée que les données ont été récupérées et sont prêtes à être utilisées.

Voici un aperçu de ce service.

```
ApiService.java x
1 package com.example.bikeshopmobile.services;
2
3 import ...
28
29 public class ApiService extends IntentService {
30
31     10 usages
32     private static final String TAG = "ApiService";
33     10 usages
34     private ApiInterface apiInterface;
35
36     public ApiService() { super( name: "ApiService" ); }
37
38     @Override
39     public void onCreate() {
40         super.onCreate();
41         apiInterface = RetrofitClient.getRetrofitInstance().create(ApiInterface.class);
42     }
43
44
45     @Override
46     protected void onHandleIntent(Intent intent) {
47         if (intent != null) {
48             final String action = intent.getAction();
49             if (action != null ) {
```

The screenshot shows a Java code editor with the file `ApiService.java` open. The code implements a `onHandleIntent` method that handles various intent actions by calling corresponding methods like `getArticles`, `getCategories`, etc. Several breakpoints are set in the code, indicated by red circles with the number '1'. Annotations are present above the code, such as 'Override' and 'D'. The right side of the screen features a vertical toolbar with icons for Device Manager, Grade, Notifications, and Device File Explorer. A status bar at the bottom displays 'Activate Windows' and other system information.

```
43
44
45
46 @Override
47 protected void onHandleIntent(Intent intent) {
48     if (intent != null) {
49         final String action = intent.getAction();
50         if (action != null ) {
51             switch (action) {
52                 case "getArticles":
53                     getArticles();
54                     break;
55                 case "getCategories":
56                     getCategories();
57                     break;
58                 case "getClients":
59                     getClients();
60                     break;
61                 case "getCommandes":
62                     getCommandes();
63                     break;
64                 case "getEmployes":
65                     getEmployes();
66                     break;
67                 case "getMagasins":
68                     getMagasins();
```

The screenshot shows a Java file named `ApiService.java` in the Android Studio code editor. The code contains a switch statement with several cases for different actions:

```
    case "getCommandes":  
        getCommandes();  
        break;  
    case "getEmployes":  
        getEmployes();  
        break;  
    case "getMagasins":  
        getMagasins();  
        break;  
    case "getMarques":  
        getMarques();  
        break;  
    case "getProduits":  
        getProduits();  
        break;  
    case "getStocks":  
        getStocks();  
        break;  
    default:  
        Log.d(TAG, msg: "Action inconnue"+action);  
        break;  
    }  
}
```

```
1 usage
2
3 private void getArticles() {
4     Call<List<ArticleCommande>> call = apiInterface.getArticles();
5     call.enqueue(new Callback<List<ArticleCommande>>() {
6         @Override
7         public void onResponse(Call<List<ArticleCommande>> call, Response<List<ArticleCommande>> response) {
8             if (response.isSuccessful()) {
9                 Intent broadcastIntent = new Intent(action: "ACTION_ARTICLES_LOADED");
10                broadcastIntent.putExtra(name: "articleCommandeList", (Serializable) response.body());
11                LocalBroadcastManager.getInstance(getApplicationContext()).sendBroadcast(broadcastIntent);
12            } else {
13                Toast.makeText(getApplicationContext(), text: "La récupération des données a échoué", Toast.LENGTH_SHORT).show();
14            }
15        }
16
17        @Override
18        public void onFailure(Call<List<ArticleCommande>> call, Throwable t) {
19            Log.d(TAG, msg: "Error" + t);
20        }
21    });
22 }
23
24 1 usage
25 private void getCategories() {
```

```

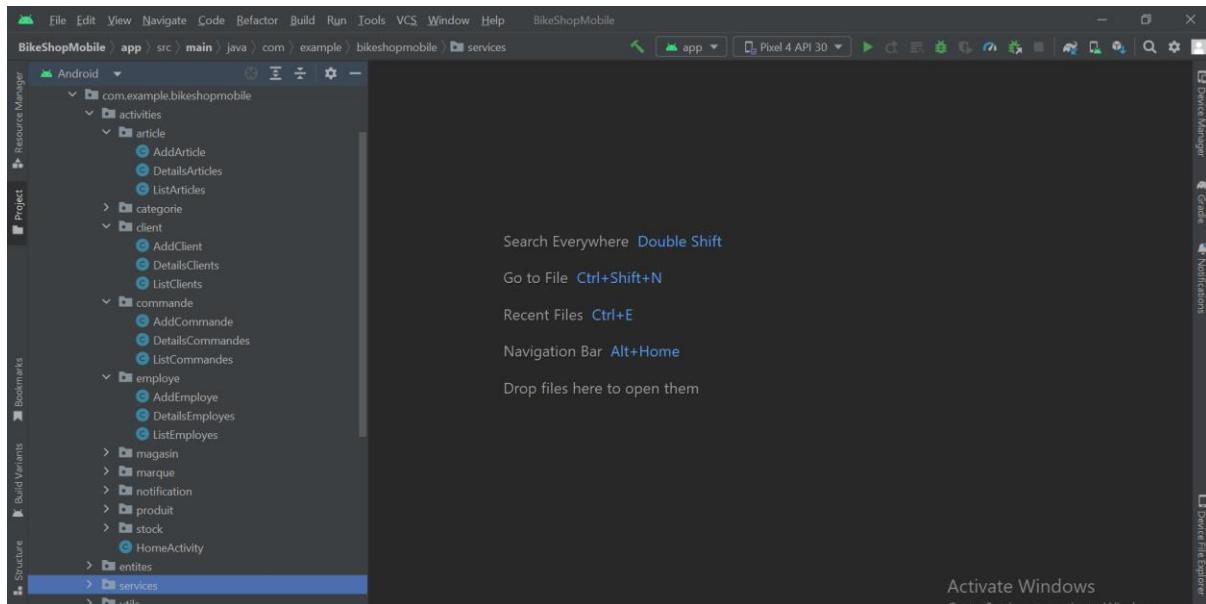
130     private void getClients() {
131         Call<List<Client>> call = apiInterface.getClients();
132         call.enqueue(
133             new Callback<List<Client>>() {
134                 @Override
135                 public void onResponse(Call<List<Client>> call, Response<List<Client>> response) {
136                     if (response.isSuccessful()) {
137                         Intent broadcastIntent = new Intent(action: "ACTION_CLIENTS_LOADED");
138                         broadcastIntent.putExtra(name: "clientlist", Serializable(response.body()));
139                         LocalBroadcastManager.getInstance(getApplicationContext()).sendBroadcast(broadcastIntent);
140                     } else {
141                         Toast.makeText(getApplicationContext(), text: "La récupération des données a échoué", LENGTH_SHORT).show();
142                     }
143                 }
144             });
145         @Override
146         public void onFailure(Call<List<Client>> call, Throwable t) {
147             Log.d(TAG, msg: "Error " + t);
148         }
149     );
150 }
151
152     1 usage
153     private void getCommandes() {

```

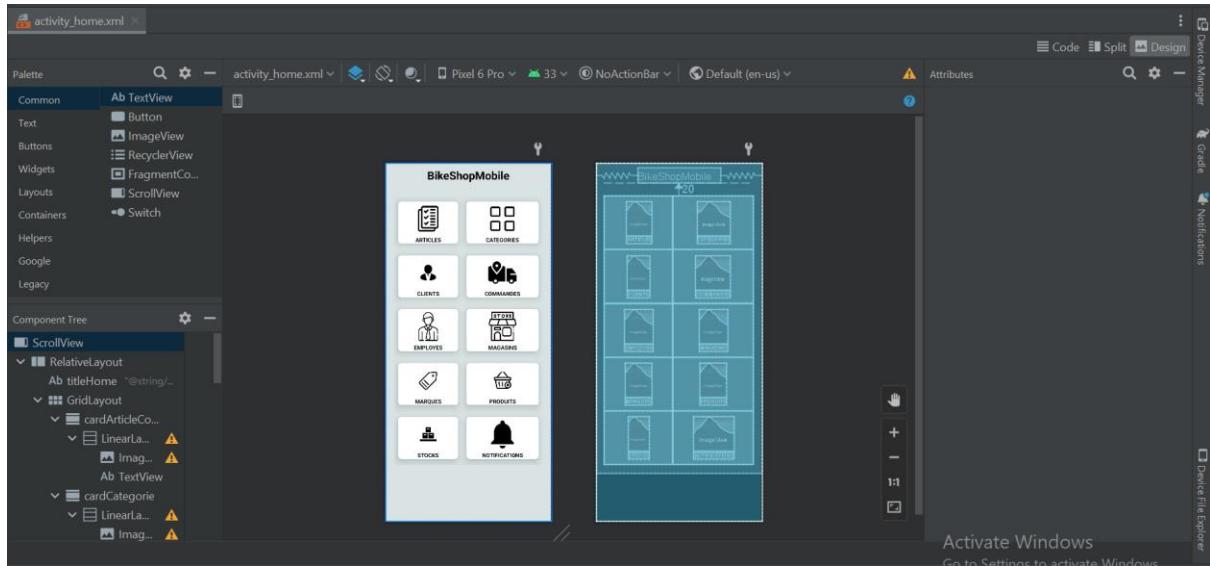
Créations des activités

Pour chacune de nos entités, nous allons donc créer les activités pour ajouter, modifier et lister les données récupérées du service web. A noter qu'au niveau de l'activité pour lister nous avons également gérer la suppression.

Comme nous pouvons le voir sur cette image, on a les trois activités pour chacune de nos entités ainsi qu'une activité HomeActivity pour permettre la navigation en choisissant quelle entité, nous souhaitons manipuler.



Nous allons d'abord commencer par voir l'implémentation pour HomeActivity qui assure la navigation entre nos différentes activités de listing.



Comme nous le voyons au niveau du layout, on définit ici un GridLayout avec pour chaque case un CardView qui lorsque cliqué nous redirige vers l'activité de listing pour l'entité en question. Il y a également un pour les notifications mais nous y reviendrons plus tard.

Au niveau du Java, nous voyons donc qu'il nous suffit de récupérer chaque carte et d'ajouter un listener pour la redirection.

```

1 package com.example.bikeshopmobile.activities;
2
3 import ...
4
5
6 public class HomeActivity extends AppCompatActivity {
7
8     public CardView article,categorie,client,commande,employe,magasin,marque,produit,stock,notif;
9
10    private static final int LOCATION_PERMISSION_REQUEST_CODE = 100;
11
12
13    @Override
14    protected void onCreate(Bundle savedInstanceState) {
15        super.onCreate(savedInstanceState);
16        setContentView(R.layout.activity_home);
17
18        article = findViewById(R.id.cardArticleCommande);
19        categorie = findViewById(R.id.cardCategorie);
20        client = findViewById(R.id.cardClient);
21        commande = findViewById(R.id.cardCommande);
22        employe = findViewById(R.id.cardEmploye);
23        magasin = findViewById(R.id.cardMagasin);
24        marque = findViewById(R.id.cardMarque);
25        produit = findViewById(R.id.cardProduit);
26        stock = findViewById(R.id.cardStock);
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46

```

```
HomeActivity.java
49     article.setOnClickListener(new View.OnClickListener() {
50
51         @Override
52         public void onClick(View view) {
53             startActivity(new Intent(packageContext: HomeActivity.this, ListArticles.class));
54         }
55     );
56
57     categorie.setOnClickListener(new View.OnClickListener() {
58
59         @Override
60         public void onClick(View view) {
61             startActivity(new Intent(packageContext: HomeActivity.this, ListCategories.class));
62         }
63     );
64
65     client.setOnClickListener(new View.OnClickListener() {
66
67         @Override
68         public void onClick(View view) {
69             startActivity(new Intent(packageContext: HomeActivity.this, ListClients.class));
70         }
71     );
72
73     commande.setOnClickListener(new View.OnClickListener() {
74
75         @Override
76         public void onClick(View view) {
77             startActivity(new Intent(packageContext: HomeActivity.this, ListCommandes.class));
78         }
79     );
80
81     employe.setOnClickListener(new View.OnClickListener() {
82
83         @Override
84         public void onClick(View view) {
85             startActivity(new Intent(packageContext: HomeActivity.this, ListEmployes.class));
86         }
87     );
88
89     magasin.setOnClickListener(new View.OnClickListener() {
90
91         @Override
92         public void onClick(View view) {
93             startActivity(new Intent(packageContext: HomeActivity.this, ListMagasins.class));
94         }
95     );
96
97     marque.setOnClickListener(new View.OnClickListener() {
98
99         @Override
100        public void onClick(View view) {
101            startActivity(new Intent(packageContext: HomeActivity.this, ListMarques.class));
102        }
103    );
104}
```

Activate Windows
Go to Settings to activate Windows

```
HomeActivity.java
73     startActivity(new Intent(packageContext: HomeActivity.this, ListCommandes.class));
74
75 );
76
77 employe.setOnClickListener(new View.OnClickListener() {
78
79         @Override
80         public void onClick(View view) {
81             startActivity(new Intent(packageContext: HomeActivity.this, ListEmployes.class));
82         }
83     );
84
85 magasin.setOnClickListener(new View.OnClickListener() {
86
87         @Override
88         public void onClick(View view) {
89             startActivity(new Intent(packageContext: HomeActivity.this, ListMagasins.class));
90         }
91     );
92
93 marque.setOnClickListener(new View.OnClickListener() {
94
95         @Override
96         public void onClick(View view) {
97             startActivity(new Intent(packageContext: HomeActivity.this, ListMarques.class));
98         }
99     );
100 }
```

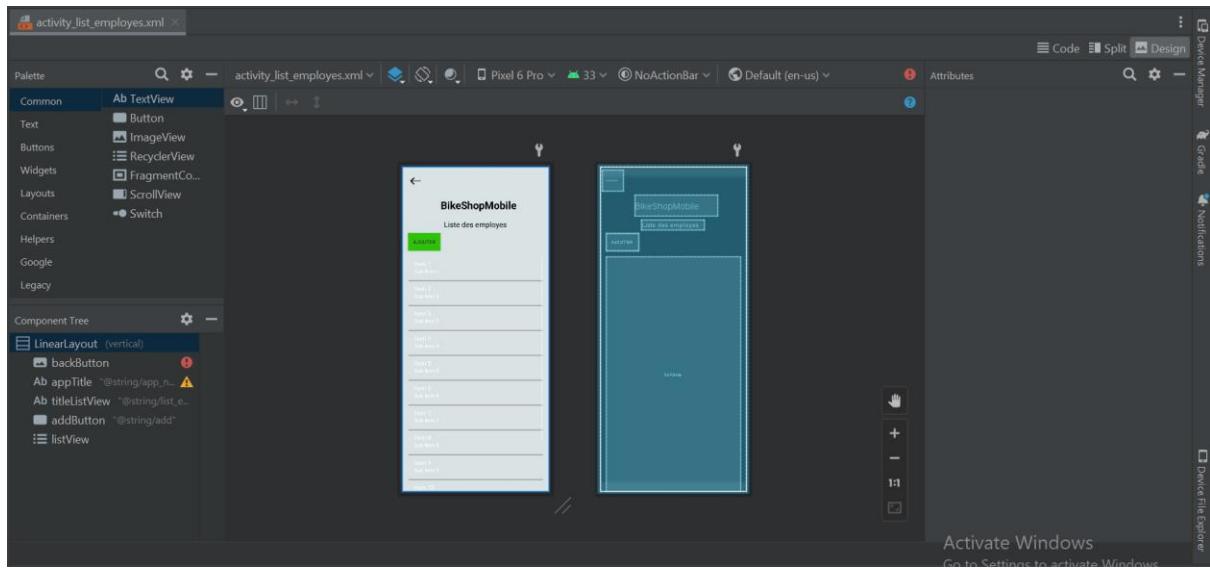
Activate Windows
Go to Settings to activate Windows

Voici le rendu que l'on obtient sur le téléphone :



Etant donné que le principe est le même pour toutes les entités avec pour seule différence les layouts et attributs, nous allons expliquer l'implémentation en utilisant l'exemple de l'entité Employe. Il sera possible de parcourir plus en détail chaque entité en regardant le code source.

ListEmployes



Comme nous le voyons au niveau du layout, on définit ici un `LinearLayout` avec une flèche de retour en arrière, le nom de l'application, le nom de la page et un bouton ajouter qui lorsque cliqué démarre l'activité d'ajout.

Pour lister nos entités, nous utilisons une `ListView`, au niveau de chaque élément de cette `ListView` on affiche le nom de l'employé ainsi que le magasin où il travaille et une icône de corbeille qui lorsque cliqué supprime l'employé. Lorsque l'élément est cliqué on est redirigé vers l'activité de modification.

Au niveau du code Java, nous récupérons donc chaque élément du layout et nous créons également des attributs pour la liste des employés et un `BroadcastReceiver`.

Lorsque l'activité démarre, nous lançons l'`ApiService` précédemment créé en choisissant comme action `getEmployes()` pour récupérer l'ensemble des employés à partir de notre service web. On initialise notre `BroadcastReceiver` de telle sorte que lorsque les données arrivent, il les stocke dans la liste des employés et créer un item dans la `ListView` pour chaque employé. On prend soin d'utiliser un `IntentFilter` afin que le `BroadcastReceiver` ne reçoive que les updates par rapport aux employés. On ajoute des listener sur l'élément et l'icône de corbeille pour gérer la modification et la suppression. Pour la modification, il s'agit de rediriger vers l'activité qui en est chargé et pour la suppression, nous avons une méthode `deleteEmploye()` qui communique avec le service web pour supprimer l'employé puis rafraîchit l'activité pour mettre à jour la liste.

On ajoute également un listener sur le `addButton` qui permet de rediriger vers l'activité d'ajout lorsqu'il est cliqué.

```

ListEmployees.java
1 package com.example.bikeshopmobile.activities.employee;
2
3 import ...
35
36 public class ListEmployees extends AppCompatActivity {
37
38     2 usages
39     List<Employee> employeelist;
40     5 usages
41     BroadcastReceiver broadcastReceiver;
42     2 usages
43     ListView listView;
44     2 usages
45     ApiInterface apiInterface;
46     2 usages
47     Button btn1;
48     2 usages
49     ImageButton btn2;
50
51     @Override
52     protected void onCreate(Bundle savedInstanceState) {
53         super.onCreate(savedInstanceState);
54         setContentView(R.layout.activity_list_employees);
55
56         listView = findViewById(R.id.listView);
57         btn1 = findViewById(R.id.addButton);
58         btn2 = findViewById(R.id.backButton);
59         apiInterface = RetrofitClient.getRetrofitInstance().create(ApiInterface.class);
60
61         broadcastReceiver = (BroadcastReceiver) (context, intent) -> {
62             if (intent != null && "ACTION_EMPLOYEES_LOADED".equals(intent.getAction())) {
63                 List<Employee> data = (List<Employee>) intent.getSerializableExtra("employeelist");
64                 if (data != null) {
65                     employeelist = data;
66                     ArrayAdapter<Employee> adapter = new ArrayAdapter<>( context: ListEmployees.this, R.layout.list_tile_layout, R.id.title, employeelist );
67                     @Override
68                     public View getView(int position, View convertView, ViewGroup parent) {
69                         View view = super.getView(position, convertView, parent);
70                         if (view == null) {
71                             LayoutInflator inflater = LayoutInflater.from(getContext());
72                             view = inflater.inflate(R.layout.list_tile_layout, parent, attachToRoot: false);
73                         }
74
75                         Employe item = getItem(position);
76                         ...
77                         return view;
78                     }
79                     listView.setAdapter(adapter);
80                 }
81             });
82
83             view.setOnClickListener(new View.OnClickListener() {
84                 @Override
85                 public void onClick(View v) {
86                     Intent intent = new Intent( packageContext: ListEmployees.this, DetailsEmployee.class );
87                     intent.putExtra( name: "selectedEmployee", item );
88                     startActivity(intent);
89                 }
90             });
91
92             trashcanIcon.setOnClickListener(new View.OnClickListener() {
93                 @Override
94                 public void onClick(View v) { deleteEmployee(item.getId()); }
95             });
96
97             ...
98         }
99     }
100
101     ...
102
103     ...
104

```

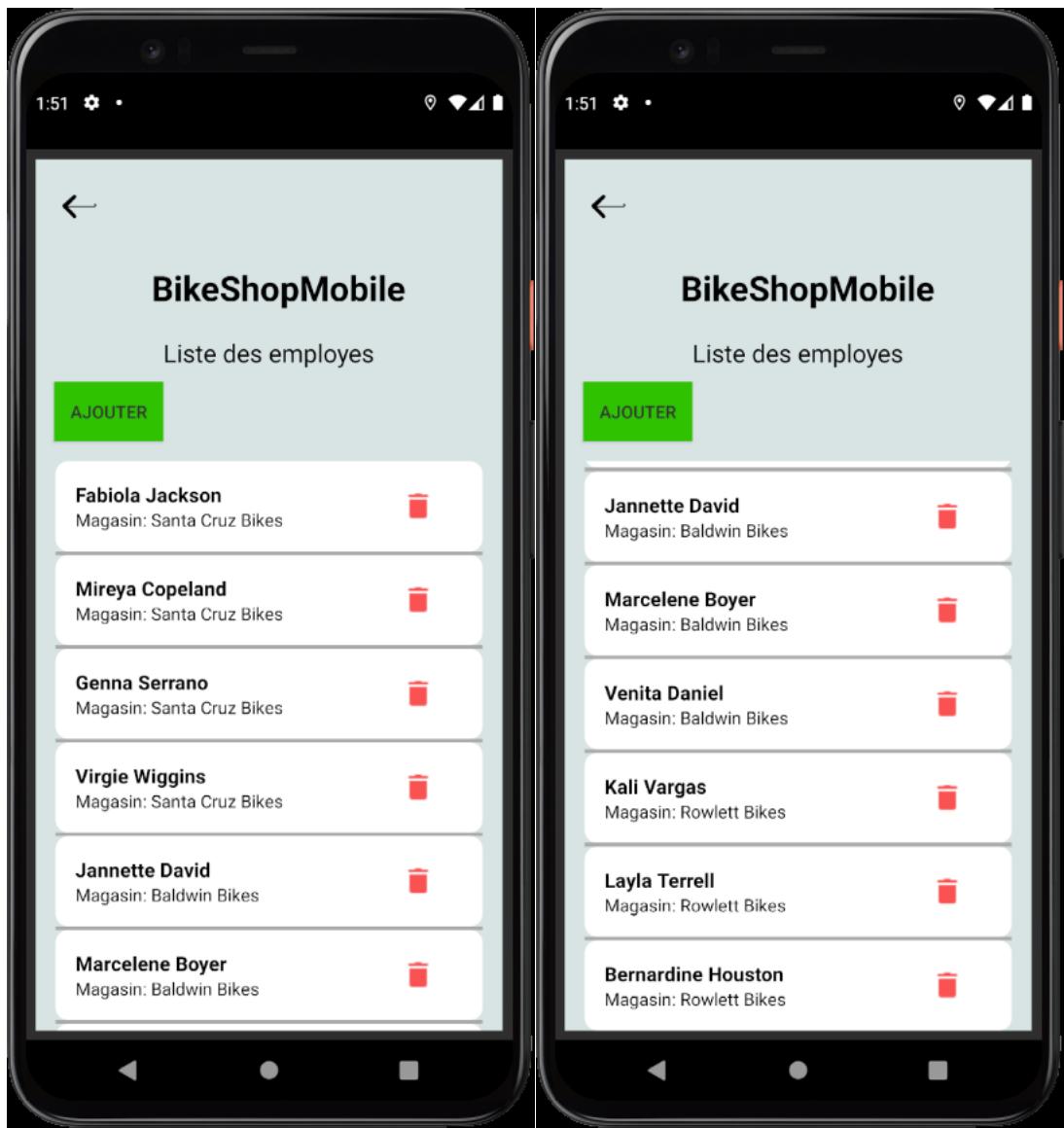
Activate Windows
Go to Settings to activate Windows

```

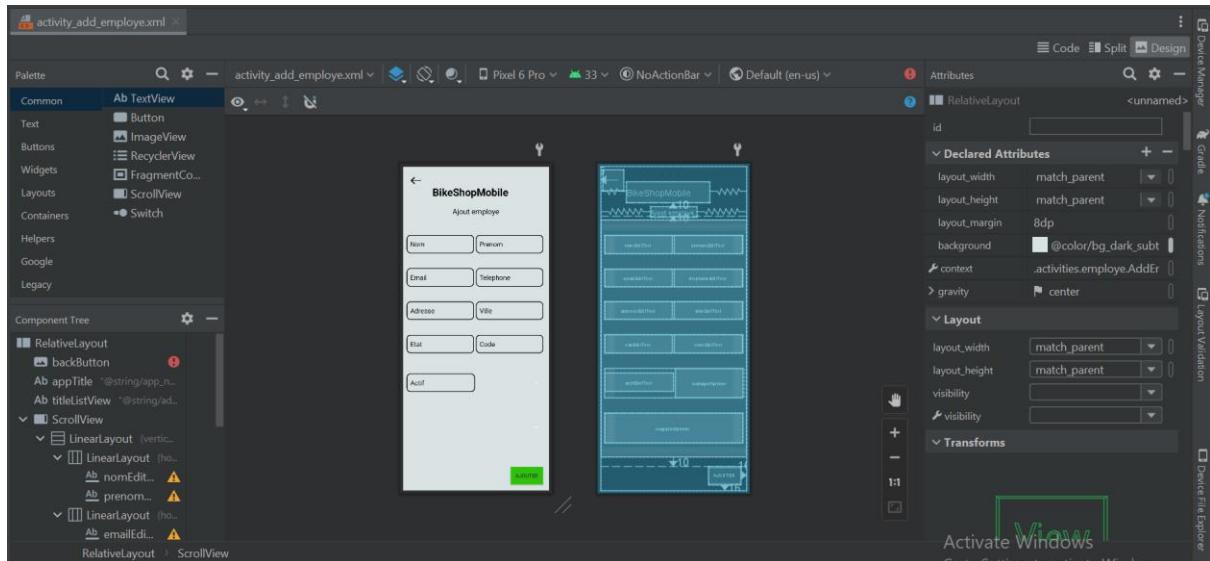
108     LocalBroadcastManager.getInstance( context: this).registerReceiver(broadcastReceiver, filter);
109
110     getEmployees();
111
112     btn1.setOnClickListener(new View.OnClickListener() {
113         @Override
114         public void onClick(View view) {
115             startActivity(new Intent( packageContext: ListEmployees.this, AddEmploye.class));
116         }
117     });
118
119     btn2.setOnClickListener(new View.OnClickListener() {
120         @Override
121         public void onClick(View view) {
122             startActivity(new Intent( packageContext: ListEmployees.this, HomeActivity.class));
123         }
124     });
125 }
126
127 @Override
128 public void onDestroy() {
129     super.onDestroy();
130     if (broadcastReceiver != null) {
131         LocalBroadcastManager.getInstance( context: this).unregisterReceiver(broadcastReceiver);
132         broadcastReceiver = null;
133     }
134 }
135
136 2 usages
137 private void getEmployees() {
138     Intent serviceIntent = new Intent( packageContext: this, ApiService.class);
139     serviceIntent.setAction("getEmployees");
140     startService(serviceIntent);
141 }
142
143 1 usage
144 private void deleteEmployee(Integer id) {
145
146     Call<Reponse> call = apiInterface.deleteEmploye(id);
147     call.enqueue(
148         new Callback<Reponse>() {
149             @Override
150             public void onResponse(Call<Reponse> call, Response<Reponse> response) {
151                 if (response.isSuccessful()) {
152                     getEmployees();
153                     Toast.makeText( context: ListEmployees.this, text: "L'employé a été supprimé et les données rechargées", Toast.LENGTH_SHORT).show();
154                 } else {
155                     Toast.makeText( context: ListEmployees.this, text: "La récupération des données a échoué", Toast.LENGTH_SHORT).show();
156                 }
157             }
158
159             @Override
160             public void onFailure(Call<Reponse> call, Throwable t) {
161                 Log.d( tag: "ListCommandes", msg: "Error "+t);
162             }
163         });
164 }

```

Voici donc le rendu sur téléphone, nous montrerons la suppression et les redirections plus en détails dans la démo :



AddEmploye

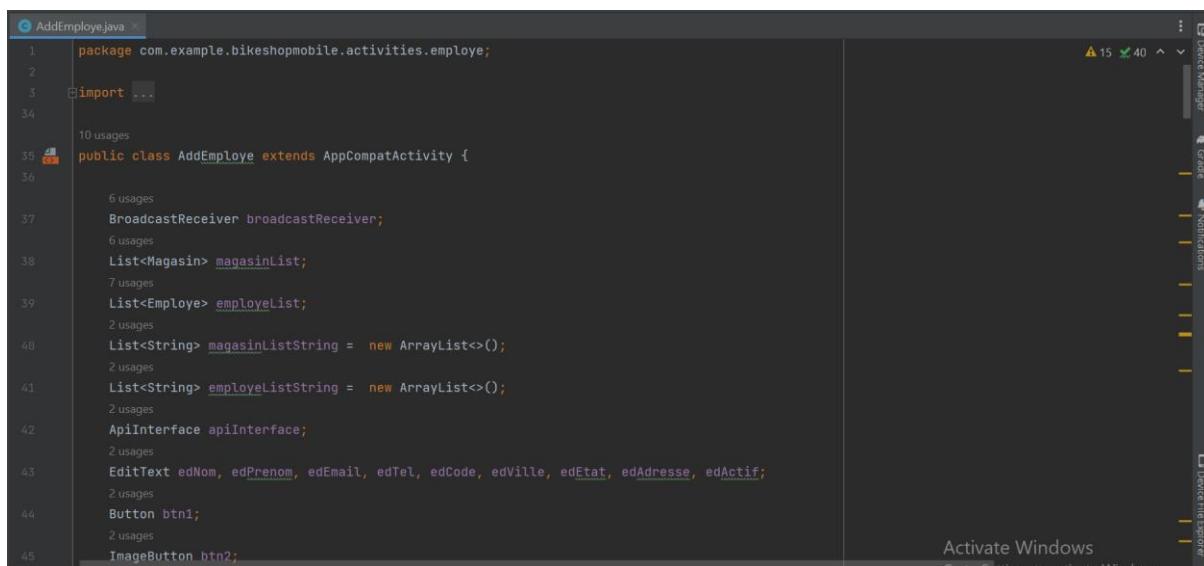


Comme nous le voyons au niveau du layout, il s'agit d'un formulaire avec les différents champs nécessaires à la création d'un employé. Nous avons également une flèche de retour, le titre de l'application et le nom de la page. Ainsi qu'un bouton ajouter qui lorsque cliqué ajoute le client et nous redirige vers la page de listing.

Au niveau du code Java, on récupérer les différents éléments présents au niveau de notre layout, on crée des listes pour les magasins et les employés pour la sélection au niveau du formulaire et un BroadcastReceiver.

Lorsque l'activité démarre, nous lançons l'ApiService avec les actions getEmployes() et getMagasins() pour récupérer la liste des employés et des magasins pour les besoins des options select. On initialise le BroadcastReceiver de façon à ce qu'à l'arrivée des données, il crée un spinner dans lequel seront placés les différents éléments. On ajoute un IntentFilter de telle sorte que seuls les données des employés et des magasins ne soient envoyés vers lui.

On ajoute un listener sur la flèche retour afin qu'il renvoie à l'activité de listing et un listener sur le bouton ajouter afin de vérifier que le formulaire est valide, en l'occurrence tous les champs obligatoires ont été remplies et si tel est le cas on exécute la méthode addEmploye() qui fait appel au service web pour enregistrer le client au niveau du serveur et redirige vers l'activité de listing.



```
1 package com.example.bikeshopmobile.activities.employe;
2
3 import ...
4
5 10 usages
6
7 public class AddEmploye extends AppCompatActivity {
8
9     6 usages
10    BroadcastReceiver broadcastReceiver;
11
12    6 usages
13    List<Magasin> magasinList;
14
15    7 usages
16    List<Employe> employeList;
17
18    2 usages
19    List<String> magasinListString = new ArrayList<>();
20
21    2 usages
22    List<String> employeListString = new ArrayList<>();
23
24    2 usages
25    ApiInterface apiInterface;
26
27    2 usages
28    EditText edNom, edPrenom, edEmail, edTel, edCode, edVille, edEtat, edAdresse, edActif;
29
30    2 usages
31    Button btn1;
32
33    2 usages
34    ImageButton btn2;
```

```
if (intent != null && "ACTION_MAGASINS_LOADED".equals(intent.getAction())) {
    List<Magasin> data = (List<Magasin>) intent.getSerializableExtra("magasin_list");
    if (data != null) {
        magasinList = data;
        for (Magasin m: magasinList) {
            magasinListString.add(m.getNom());
        }
    }
    Spinner magasinSpinner = findViewById(R.id.magasinSpinner);
    CustomSpinnerAdapter spinnerAdapter = new CustomSpinnerAdapter(context, R.layout.spinner_item, magasinList);
    spinnerAdapter.setDropDownViewResource(R.layout.spinner_item);
    magasinSpinner.setAdapter(spinnerAdapter);
    if (employe != null) {
        selectedMagasinPosition = getMagasinPosition(employe.getMagasinId().getNom());
        magasinSpinner.setSelection(selectedMagasinPosition);
    }
    magasinSpinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
        @Override
        public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
            View selectedItem = parent.getSelectedView();
            if (selectedItem != null) {
                selectedItem.setBackgroundColor(getResources().getColor(R.color.bg_dark_subtle));
                String selectedMagasin = (String) parent.getItemAtPosition(position);
                selectedMagasinPosition= getMagasinPosition(selectedMagasin);
                magasinSpinner.setSelection(selectedMagasinPosition);
            }
        }
    });
}
```

```
110     spinnerAdapter.setDropDownViewResource(R.layout.spinner_item);
111     employeSpinner.setAdapter(spinnerAdapter);
112     if (employe != null) {
113         selectedEmployeePosition= getEmployePosition( <: employe.getNom() + " " + employe.getPrenom());
114         employeSpinner.setSelection(selectedEmployeePosition);
115     }
116     employeSpinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
117         @Override
118         public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
119             View selectedItem = parent.getSelectedView();
120             if (selectedItem != null) {
121                 selectedItem.setBackgroundColor(getResources().getColor(R.color.bg_dark_subtle));
122                 String employe = (String) parent.getItemAtPosition(position);
123                 selectedEmployeePosition= getEmployePosition(employe);
124                 employeSpinner.setSelection(selectedEmployeePosition);
125             }
126         }
127         @Override
128         public void onNothingSelected(AdapterView<?> parent) {
129     }
130     });
131 }
132 }
133 }
134 }
```

```
135
136     btn1.setOnClickListener(new View.OnClickListener() {
137         @Override
138         public void onClick(View view) {
139             String nom =edNom.getText().toString();
140             String prenom=edPrenom.getText().toString();
141             String email=edEmail.getText().toString();
142             Magasin magasin=magasinList.get(selectedMagasinPosition);
143             Short actif=Short.parseShort(edActif.getText().toString());
144             if (nom==null||prenom==null||email==null||actif==null||magasin==null) {
145                 Toast.makeText(context, AddEmploye.this, text: "Le nom, prenom, email, actif et le magasin sont obligatoires", Toast.LENGTH_SHORT)
146             } else {
147                 Employe employe = new Employe();
148                 employe.setNom(nom);
149                 employe.setPrenom(prenom);
150                 employe.setEmail(email);
151                 employe.setTelephone(edTel.getText().toString());
152                 employe.setActif(actif);
153                 employe.setAdresse(new Adresse(edAdresse.getText().toString(),edVille.getText().toString(),edEtat.getText().toString(),edCode.
154                 employe.setMagasinId(magasin);
155                 employe.setManagerId(employeList.get(selectedEmployePosition));
156                 addEmployee(employe);
157             }
158         }
159     });
160 }
```

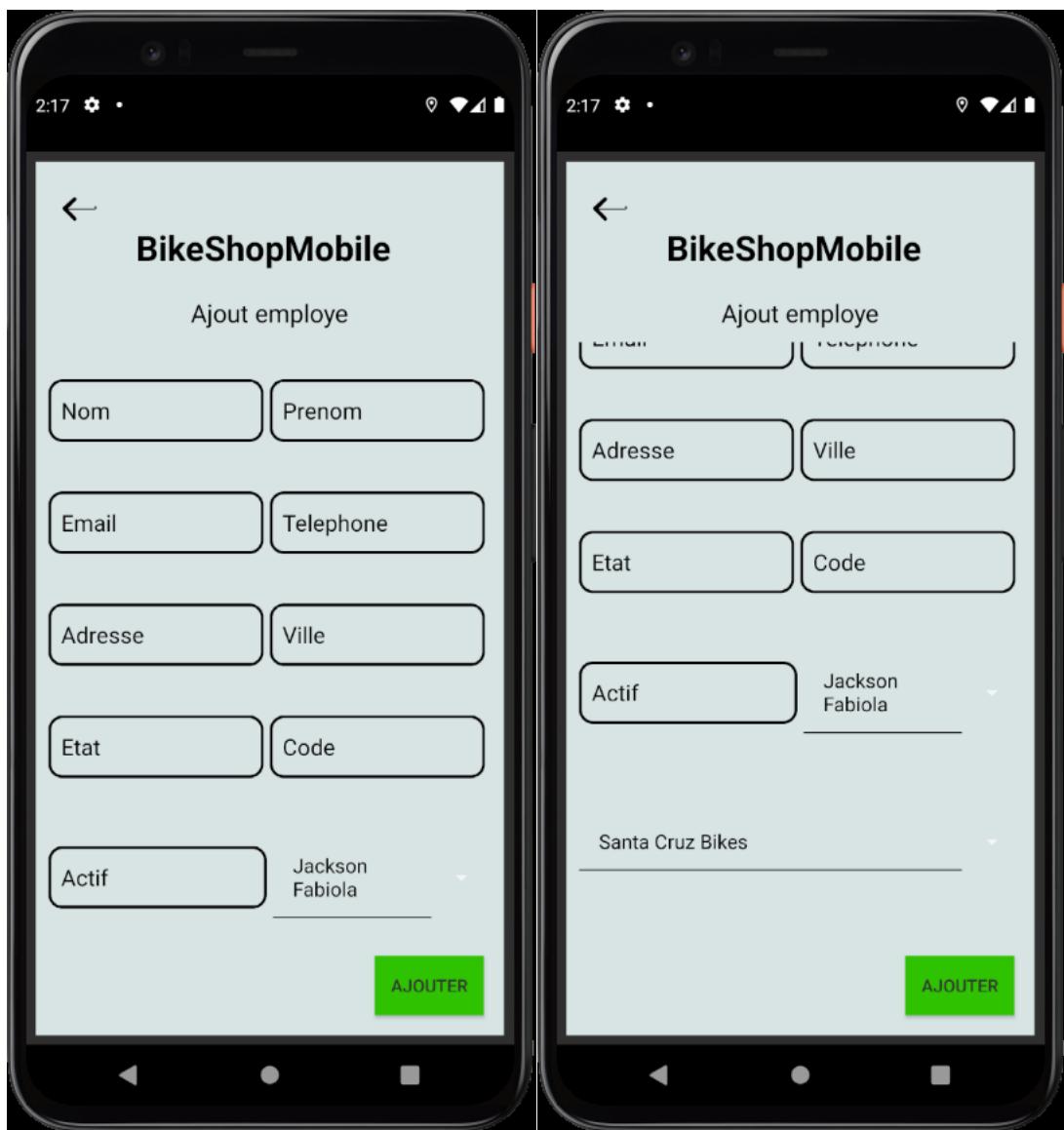
```
162
163     @Override
164     public void onClick(View view) {
165         startActivityForResult(new Intent(packageContext, AddEmploye.this, ListEmployes.class));
166     }
167
168     IntentFilter filter1 = new IntentFilter(action: "ACTION_MAGASINS_LOADED");
169     LocalBroadcastManager.getInstance(context: this).registerReceiver(broadcastReceiver, filter1);
170     IntentFilter filter2 = new IntentFilter(action: "ACTION_EMPLOYES_LOADED");
171     LocalBroadcastManager.getInstance(context: this).registerReceiver(broadcastReceiver, filter2);
172     getMagasins();
173     getEmployees();
174 }
175
176     @Override
177     public void onDestroy() {
178         super.onDestroy();
179         if (broadcastReceiver != null) {
180             LocalBroadcastManager.getInstance(context: this).unregisterReceiver(broadcastReceiver);
181             broadcastReceiver = null;
182         }
183     }
184
185     2 usages
186     private int getMagasinPosition(String s) {
187
188         if (s != null && magasinList != null) {
189             for (int i = 0; i < magasinList.size(); i++) {
190                 if (s.equals(magasinList.get(i).getNom())) {
191                     return i;
192                 }
193             }
194             return 0;
195         }
196
197         2 usages
198         private int getEmployePosition(String s) {
199             if (s != null && employeList != null) {
200                 for (int i = 0; i < employeList.size(); i++) {
201                     if (s.equals(employeList.get(i).getNom()+" "+employeList.get(i).getPrenom())) {
202                         return i;
203                     }
204                 }
205                 return 0;
206             }
207
208             1 usage
209             private void getMagasins() {
210                 Intent serviceIntent = new Intent("com.example.myapp.magasin");
211                 serviceIntent.putExtra("place", "Angevine place");
212             }
213         }
214     }
215 }
```

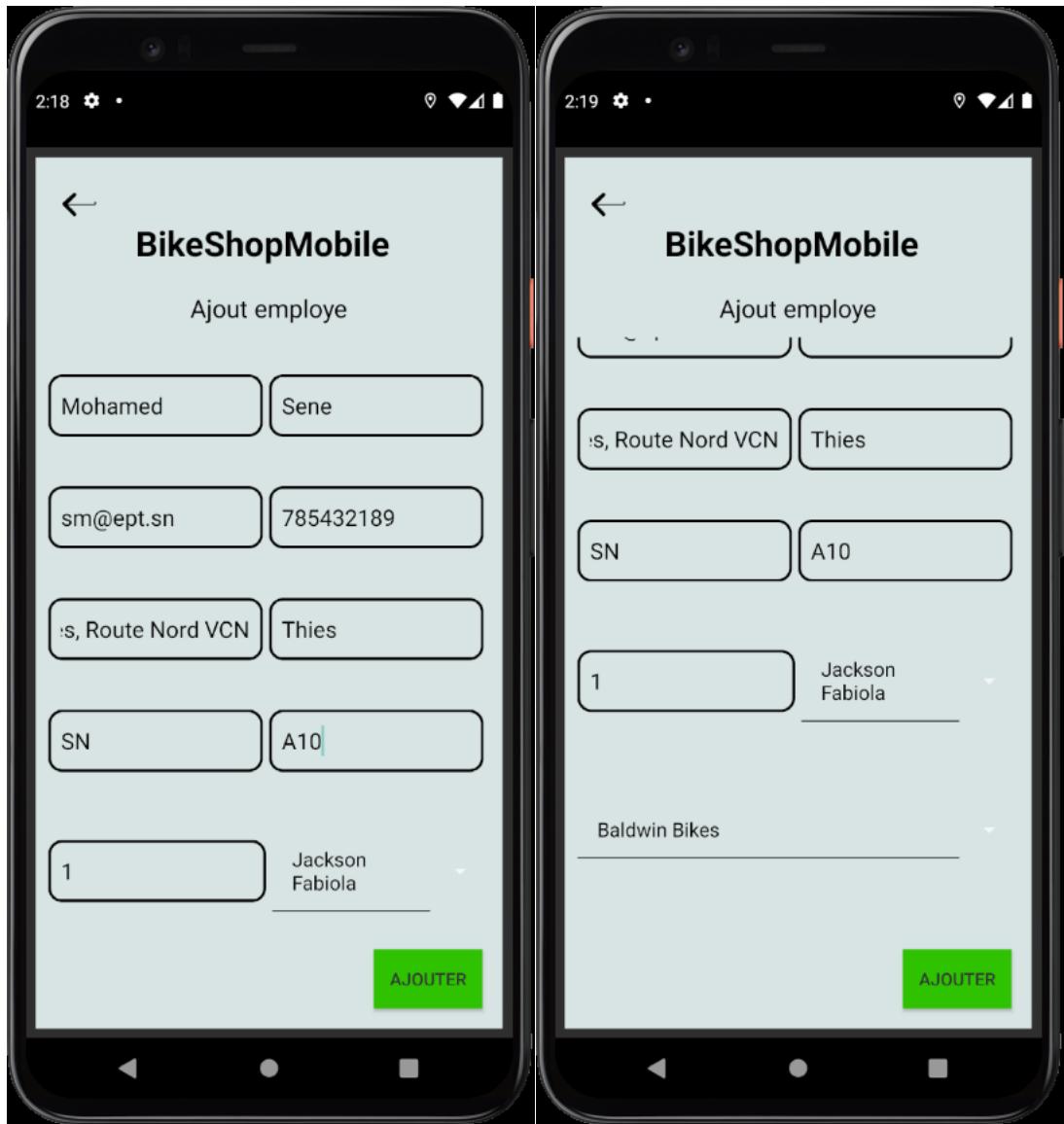
```
186
187         if (s != null && magasinList != null) {
188             for (int i = 0; i < magasinList.size(); i++) {
189                 if (s.equals(magasinList.get(i).getNom())) {
190                     return i;
191                 }
192             }
193             return 0;
194         }
195
196         2 usages
197         private int getEmployePosition(String s) {
198             if (s != null && employeList != null) {
199                 for (int i = 0; i < employeList.size(); i++) {
200                     if (s.equals(employeList.get(i).getNom()+" "+employeList.get(i).getPrenom())) {
201                         return i;
202                     }
203                 }
204                 return 0;
205             }
206
207             1 usage
208             private void getMagasins() {
209                 Intent serviceIntent = new Intent("com.example.myapp.magasin");
210                 serviceIntent.putExtra("place", "Angevine place");
211             }
212         }
213     }
214 }
```

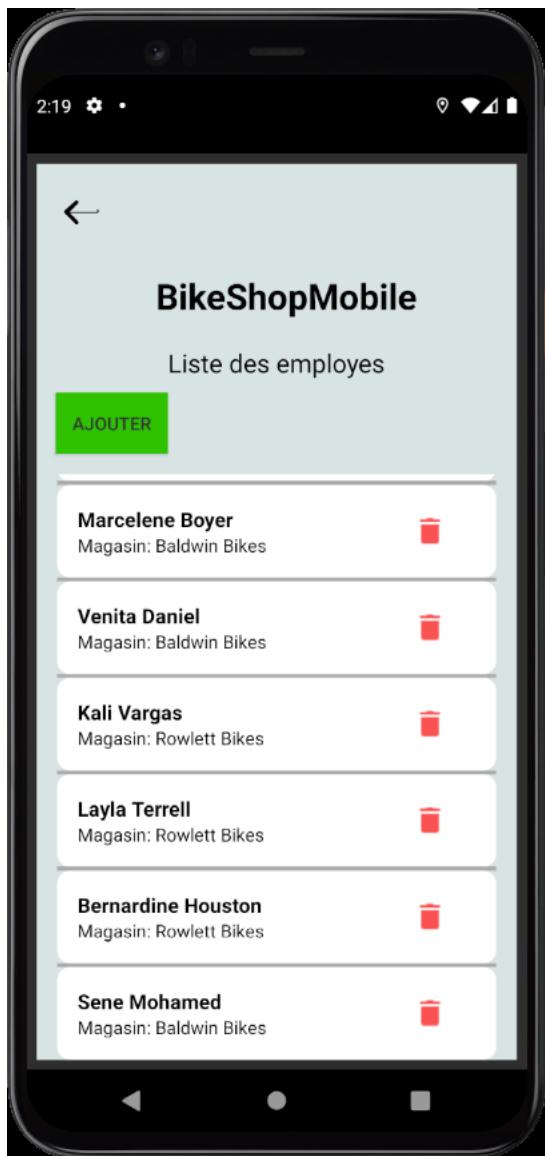
```
private void addEmploye(Employe employe) {
    Call<Employe> call = apiInterface.addOrUpdateEmploye(employe);
    call.enqueue(
        new Callback<Employe>() {
            @Override
            public void onResponse(Call<Employe> call, Response<Employe> response) {
                if (response.isSuccessful()) {
                    startActivity(new Intent(getApplicationContext(), ListEmployes.class));
                } else {
                    Toast.makeText(context, "L'enregistrement des données a échoué", Toast.LENGTH_SHORT).show();
                }
            }

            @Override
            public void onFailure(Call<Employe> call, Throwable t) {
                Log.d("AddEmploye", msg: "Error "+t);
            }
        });
}
```

Voici le rendu sur téléphone ainsi que le résultat de l'ajout d'un employé :







employee x personne x

WHERE ORDER BY id DESC

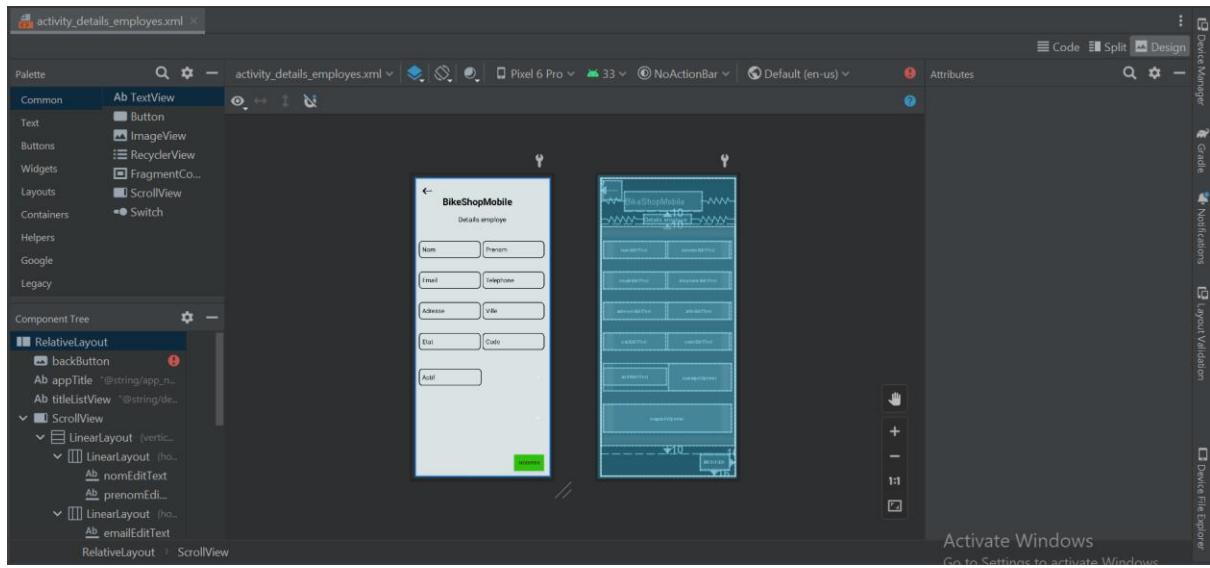
	id	actif	magasin_id	manager_id
1	1458	1	2	1446
2	1457	1	1	<null>
3	1456	1	1	<null>
4	1455	1	3	1452
5	1454	1	3	1452
6	1453	1	3	1446
7	1452	1	2	1450
8	1451	1	2	1450
9	1450	1	2	1446
10	1449	1	1	1447
11	1448	1	1	1447
12	1447	1	1	1446
13	1446	1	1	<null>

employee x personne x

WHERE ORDER BY id DESC

	id	dtype	email	nom	prenom	telephone
1	1458	Employe	sm@ept.sn	Mohamed	Sene	785432189
2	1457	Employe	mobileshop@bike.shop	Shop	Mobile	(972) 530-7891
3	1456	Employe	wwebshop@bike.shop	Shop	Web	(972) 530-7890
4	1455	Employe	bernardine.houston@bikes.shop	Houston	Bernardine	(972) 530-5557
5	1454	Employe	layla.terrell@bikes.shop	Terrell	Layla	(972) 530-5556
6	1453	Employe	kali.vargas@bikes.shop	Vargas	Kali	(972) 530-5555
7	1452	Employe	venita.daniel@bikes.shop	Daniel	Venita	(516) 379-4446
8	1451	Employe	marcelene.boyer@bikes.shop	Boyer	Marcelene	(516) 379-4445
9	1450	Employe	jannette.david@bikes.shop	David	Jannette	(516) 379-4444
10	1449	Employe	virgie.wiggins@bikes.shop	Wiggins	Virgie	(831) 555-5557
11	1448	Employe	genna.serrano@bikes.shop	Serrano	Genna	(831) 555-5556
12	1447	Employe	mireya.copeland@bikes.shop	Copeland	Mireya	(831) 555-5555
13	1446	Employe	fabiola.jackson@bikes.shop	Jackson	Fabiola	(831) 555-5554
14	1445	Client	ester.acevedo@gmail.com	Acevedo	Ester	<null>
15	1444	Client	ivette.estes@gmail.com	Estes	Ivette	<null>
16	1443	Client	lezie.lamb@gmail.com	Lamb	Leenzie	<null>
17	1442	Client	cassie.cline@gmail.com	Cline	Cassie	<null>
18	1441	Client	jamaal.morrison@msn.com	Morrison	Jamaal	<null>
19	1440	Client	ernest.rollins@yahoo.com	Rollins	Ernest	<null>
20	1439	Client	florrie.little@yahoo.com	Little	Florrie	<null>

DetailsEmployees



Comme nous pouvons le voir au niveau du layout, il est similaire à celui d'ajout à l'exception du titre de la page et du texte du bouton.

Au niveau du code Java, il fonctionne de façon similaire à celui de la page d'ajout également. A l'exception que lors de l'affichage de la page nous initialisons les champs du formulaire par leur valeur actuelle dans la base. Le reste fonctionne de façon similaire.

```

1 package com.example.bikeshopmobile.activities.employee;
2
3 import ...
4
5 9 usages
6
7 public class DetailsEmployees extends AppCompatActivity {
8
9     6 usages
10    BroadcastReceiver broadcastReceiver;
11
12    6 usages
13    List<Magasin> magasinList;
14
15    7 usages
16    List<Employe> employeList;
17
18    2 usages
19    List<String> magasinListString = new ArrayList<>();
20
21    2 usages
22    List<String> employeListString = new ArrayList<>();
23
24    2 usages
25    ApiInterface apiInterface;
26
27    3 usages
28    EditText edNom, edPrenom, edEmail, edTel, edCode, edVille, edEtat, edAdresse, edActif;
29
30    2 usages
31    Button btn1;
32
33    2 usages
34    ImageButton btn2;
35
36
37
38
39
40
41
42
43
44
45

```

```
47
48     @Override
49     protected void onCreate(Bundle savedInstanceState) {
50         super.onCreate(savedInstanceState);
51         setContentView(R.layout.activity_details_employes);
52
53         apiInterface = RetrofitClient.getRetrofitInstance().create(ApiInterface.class);
54         Employe selectedEmploye = (Employe) getIntent().getSerializableExtra("selectedEmploye");
55         edNom = findViewById(R.id.nomEditText);
56         edPrenom = findViewById(R.id.prenomEditText);
57         edEmail = findViewById(R.id.emailEditText);
58         edTel = findViewById(R.id.telephoneEditText);
59         edCode = findViewById(R.id.codeEditText);
60         edVille = findViewById(R.id.villeEditText);
61         edEtat = findViewById(R.id.etatEditText);
62         edAdresse = findViewById(R.id.adresseEditText);
63         edActif = findViewById(R.id.actifEditText);
64         btn1 = findViewById(R.id.modifyButton);
65         btn2 = findViewById(R.id.backButton);
66
67         edNom.setText(String.valueOf(selectedEmploye.getNom()));
68         edPrenom.setText(String.valueOf(selectedEmploye.getPrenom()));
69         edEmail.setText(String.valueOf(selectedEmploye.getEmail()));
70         edTel.setText(String.valueOf(selectedEmploye.getTelephone()));
71         edActif.setText(String.valueOf(selectedEmploye.getActif()));


```

```
edPrenom.setText(String.valueOf(selectedEmploye.getPrenom()));
edEmail.setText(String.valueOf(selectedEmploye.getEmail()));
edTel.setText(String.valueOf(selectedEmploye.getTelephone()));
edActif.setText(String.valueOf(selectedEmploye.getActif()));

if (selectedEmploye.getAdresse() != null) {
    edCode.setText(String.valueOf(selectedEmploye.getAdresse().getCodeZip()));
    edVille.setText(String.valueOf(selectedEmploye.getAdresse().getVille()));
    edAdresse.setText(String.valueOf(selectedEmploye.getAdresse().getAdresse()));
    edEtat.setText(String.valueOf(selectedEmploye.getAdresse().getEtat()));
} else {
    edCode.setText(null);
    edVille.setText(null);
    edAdresse.setText(null);
    edEtat.setText(null);
}

broadcastReceiver = (BroadcastReceiver) (context, intent) -> {
    if (intent != null && "ACTION_MAGASINS_LOADED".equals(intent.getAction())) {
        List<Magasin> data = (List<Magasin>) intent.getSerializableExtra("magasinList");
        if (data != null) {
            magasinList = data;
            for (Magasin m: magasinList) {
                magasinListString.add(m.getNom());
            }
        }
        Spinner magasinSpinner = findViewById(R.id.magasinSpinner);
    }
}
```

```
83
84 broadcastReceiver = [BroadcastReceiver] (context, intent) > {
85     if (intent != null && "ACTION_MAGASINS_LOADED".equals(intent.getAction())) {
86         List<Magasin> data = (List<Magasin>) intent.getSerializableExtra("magasinList");
87         if (data != null) {
88             magasinList = data;
89             for (Magasin m: magasinList) {
90                 magasinListString.add(m.getNom());
91             }
92         }
93     }
94     Spinner magasinSpinner = findViewById(R.id.magasinSpinner);
95     CustomSpinnerAdapter spinnerAdapter = new CustomSpinnerAdapter(context, DetailsEmployes.this, R.layout.spinner_item, magasinList);
96     spinnerAdapter.setDropDownViewResource(R.layout.spinner_item);
97     magasinSpinner.setAdapter(spinnerAdapter);
98     if (selectedEmploye != null) {
99         selectedMagasinPosition = getMagasinPosition(selectedEmploye.getMagasinId().getNom());
100        magasinSpinner.setSelection(selectedMagasinPosition);
101    }
102    magasinSpinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
103        @Override
104        public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
105            View selectedItem = parent.getSelectedView();
106            if (selectedItem != null) {
107                selectedItem.setBackgroundColor(getResources().getColor(R.color.bg_dark_subtle));
108                String selectedMagasin = (String) parent.getItemAtPosition(position);
109                selectedMagasinPositions = getMagasinPosition(selectedMagasin);
110            }
111        }
112    });
113 }
```

```

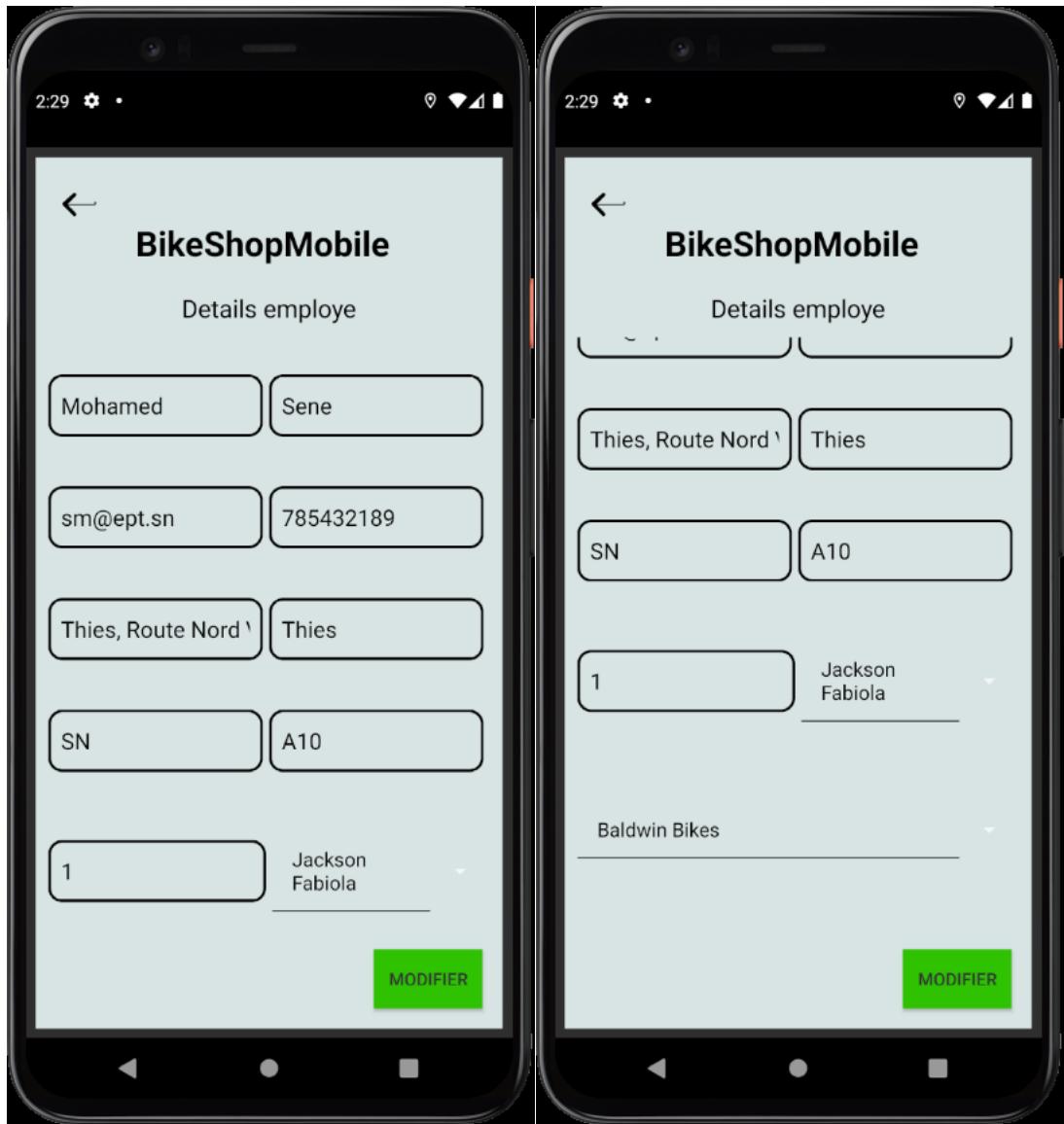
118     } else if (intent != null && "ACTION_EMPLOYES_LOADED".equals(intent.getAction())) {
119         List<Employee> data = (List<Employee>) intent.getSerializableExtra("employeeList");
120         if (data != null) {
121             employeList = data;
122             for (Employee e: employeList) {
123                 employeListString.add(e.getNom() + " " + e.getPrenom());
124             }
125             Spinner employeSpinner = findViewById(R.id.managerSpinner);
126             CustomSpinnerAdapter spinnerAdapter = new CustomSpinnerAdapter(context: DetailsEmployees.this, R.layout.spinner_item, employeList);
127             spinnerAdapter.setDropDownViewResource(R.layout.spinner_item);
128             employeSpinner.setAdapter(spinnerAdapter);
129             if (selectedEmployee != null) {
130                 selectedEmployeePosition = getEmployeePosition(" " + selectedEmployee.getNom() + " " + selectedEmployee.getPrenom());
131                 employeSpinner.setSelection(selectedEmployeePosition);
132             }
133             employeSpinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
134                 @Override
135                 public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
136                     View selectedItem = parent.getSelectedView();
137                     if (selectedItem != null) {
138                         selectedItem.setBackgroundDrawable(getResources().getDrawable(R.drawable.bg_dark_subtle));
139                         String selectedEmploye = (String) parent.getItemAtPosition(position);
140                         selectedEmployeePosition = getEmployeePosition(selectedEmploye);
141                         employeSpinner.setSelection(selectedEmployeePosition);
142                     }
143                 }
144             });
145         }
146     }
147 }
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202

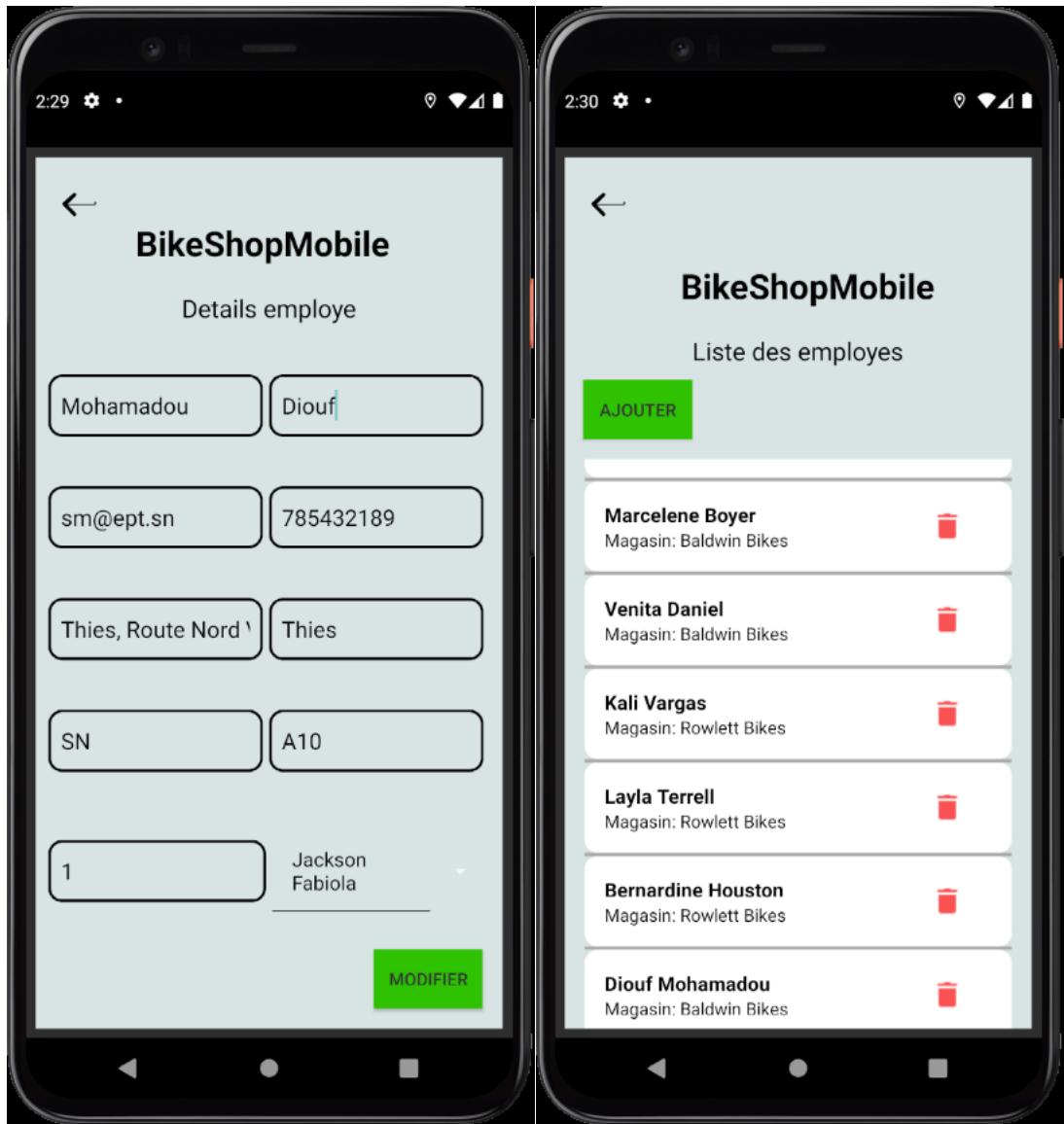
```

```

205     for (int i = 0; i < magasinList.size(); i++) {
206         if (s.equals(magasinList.get(i).getNom())) {
207             return i;
208         }
209     }
210     return -1;
211 }
212
213 /**
214  * @param s
215  * @return
216  */
217 private int getEmployeePosition(String s) {
218     if (s != null && employeList != null) {
219         for (int i = 0; i < employeList.size(); i++) {
220             if (s.equals(employeList.get(i).getNom() + " " + employeList.get(i).getPrenom())) {
221                 return i;
222             }
223         }
224     }
225     return -1;
226 }
227
228 /**
229  * @return
230  */
231 private void getMagasins() {
232     Intent serviceIntent = new Intent(getApplicationContext(), ApiService.class);
233     serviceIntent.setAction("getMagasins");
234 }
235
236 /**
237  * @param employe
238  */
239 private void getEmployees() {
240     Intent serviceIntent = new Intent(getApplicationContext(), ApiService.class);
241     serviceIntent.setAction("getEmployees");
242     startService(serviceIntent);
243 }
244
245 /**
246  * @param employe
247  */
248 private void updateEmployee(Employe employe) {
249     Call<Employe> call = apiInterface.addOrUpdateEmployee(employe);
250     call.enqueue(
251         new Callback<Employe>() {
252             @Override
253             public void onResponse(Call<Employe> call, Response<Employe> response) {
254                 if (response.isSuccessful()) {
255                     startActivity(new Intent(getApplicationContext(), DetailsEmployees.this, ListEmployees.class));
256                 } else {
257                     Toast.makeText(context, "L'enregistrement des données a échoué", Toast.LENGTH_SHORT).show();
258                 }
259             }
260             @Override
261             public void onFailure(Call<Employe> call, Throwable t) {
262                 Log.d("DetailsEmployees", "Error " + t);
263             }
264         });
265 }
266
267 }
268
269 
```

Voici le rendu sur téléphone ainsi que le résultat de la modification :





personne

WHERE ORDER BY id DESC

	id	dtype	email	nom	prenom	telephone
1	1458	Employe	sm@dept.sn	Mohamadou	Diouf	785432189
2	1457	Employe	mobileshop@bike.shop	Shop	Mobile	(972) 530-7891
3	1456	Employe	wwebshop@bike.shop	Shop	Web	(972) 530-7890
4	1455	Employe	bernardine.houston@bikes.shop	Houston	Bernardine	(972) 530-5557
5	1454	Employe	layla.terrell@bikes.shop	Terrell	Layla	(972) 530-5556
6	1453	Employe	kali.vargas@bikes.shop	Vargas	Kali	(972) 530-5555
7	1452	Employe	venita.daniel@bikes.shop	Daniel	Venita	(516) 379-4446
8	1451	Employe	marcelene.boyer@bikes.shop	Boyer	Marcelene	(516) 379-4445
9	1450	Employe	jannette.david@bikes.shop	David	Jannette	(516) 379-4444
10	1449	Employe	virgie.wiggins@bikes.shop	Wiggins	Virgie	(831) 555-5557
11	1448	Employe	genna.serrano@bikes.shop	Serrano	Genna	(831) 555-5556
12	1447	Employe	mireya.copeland@bikes.shop	Copeland	Mireya	(831) 555-5555
13	1446	Employe	fabiola.jackson@bikes.shop	Jackson	Fabiola	(831) 555-5554
14	1445	Client	ester.acevedo@gmail.com	Acevedo	Ester	<null>
15	1444	Client	ivette.estes@gmail.com	Estes	Ivette	<null>
16	1443	Client	lezelie.lamb@gmail.com	Lamb	Lezelie	<null>
17	1442	Client	cassie.cline@gmail.com	Cline	Cassie	<null>
18	1441	Client	jamaal.morrison@msn.com	Morrison	Jamaal	<null>
19	1440	Client	ernest.rollins@yahoo.com	Rollins	Ernest	<null>
20	1439	Client	florrie.little@yahoo.com	Little	Florrie	<null>
21	1438	Client	lee.dunecool.com	Dunn	Lee	(608) 807-6157

II. Gestion offline des stocks

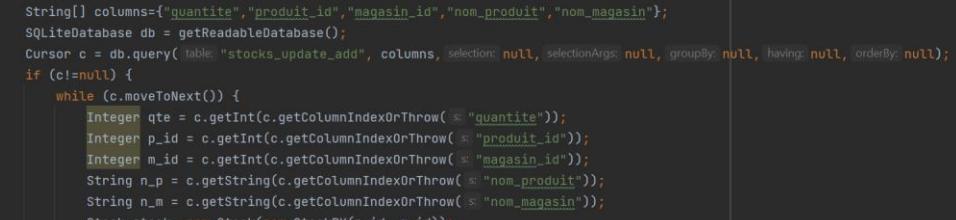
Pour commencer nous créons une class **Database** qui étend **SQLiteOpenHelper** pour la création et la gestion de la base de données SQLite sur Android.

Nous créons trois table une table stocks qui contiendra les données récupérées au niveau du service web ainsi que les tables stocks_update_add et stocks_delete qui contiendront les modifications ou les ajouts effectués hors lignes et les stocks supprimés.

Nous créons des méthodes pour la récupération, l'ajout et la suppression au niveau de la base de données que nous utiliserons plus tard dans nos activités

```
1 package com.example.bikeshopmobile;
2
3 import ...
4
5
6 public class Database extends SQLiteOpenHelper {
7
8     14 usages
9     public Database(@Nullable Context context, @Nullable String name, @Nullable SQLiteDatabase.CursorFactory factory, int version) {
10         super(context, name, factory, version);
11     }
12
13
14     @Override
15     public void onCreate(SQLiteDatabase sqLiteDatabase) {
16         String qry1 = "create table stocks(quantite integer, produit_id integer, magasin_id integer, nom_produit text, nom_magasin text";
17         String qry2 = "create table stocks_update_add(quantite integer, produit_id integer, magasin_id integer, nom_produit text, nom_magasin text";
18         String qry3 = "create table stocks_delete(produit_id integer, magasin_id integer)";
19         sqLiteDatabase.execSQL(qry1);
20         sqLiteDatabase.execSQL(qry2);
21         sqLiteDatabase.execSQL(qry3);
22     }
23
24
25     @Override
26     public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45     @Override
46     public List<Stock> getAllEntries() {
47
48         1 usage
49         List<Stock> stocks = new ArrayList<>();
50         String[] columns={ "quantite", "produit_id", "magasin_id", "nom_produit", "nom_magasin"};
51         SQLiteDatabase db = getReadableDatabase();
52         Cursor c = db.query("stocks", columns, selection: null, selectionArgs: null, groupBy: null, having: null, orderBy: null);
53         if (c!=null) {
54             while (c.moveToNext()) {
55                 Integer qte = c.getInt(c.getColumnIndexOrThrow("quantite"));
56                 Integer p_id = c.getInt(c.getColumnIndexOrThrow("produit_id"));
57                 Integer m_id = c.getInt(c.getColumnIndexOrThrow("magasin_id"));
58                 String n_p = c.getString(c.getColumnIndexOrThrow("nom_produit"));
59                 String n_m = c.getString(c.getColumnIndexOrThrow("nom_magasin"));
60                 Stock stock =new Stock(new StockPK(p_id, m_id));
61                 stock.setQuantite(qte);
62                 Magasin mag = new Magasin(m_id);
63                 mag.setNom(n_m);
64                 Produit prod = new Produit(p_id);
65                 prod.setNom(n_p);
66             }
67         }
68     }
69 }
```

```
72     }
73     db.close();
74     return stocks;
75 }
76
77 /**
78  * usage
79  */
80 public List<Stock> getDeletedStocks() {
81     List<Stock> stocks = new ArrayList<>();
82     String[] columns={<code>"product_id", "magasin_id"</code>};
83     SQLiteDatabase db = getReadableDatabase();
84     Cursor c = db.query(<code>"stocks_delete"</code>, columns, <code>selection</code>: null, <code>selectionArgs</code>: null, <code>groupBy</code>: null, <code>having</code>: null, <code>orderBy</code>: null);
85     if (<code>c!=null</code>) {
86         while (<code>c.moveToNext()</code>) {
87             Integer p_id = c.getInt(c.getColumnIndexOrThrow(<code>"product_id"</code>));
88             Integer m_id = c.getInt(c.getColumnIndexOrThrow(<code>"magasin_id"</code>));
89             Stock stock =new Stock(new StockPK(p_id, m_id));
90             stocks.add(stock);
91         }
92     }
93     c.close();
94     db.close();
95     return stocks;
96 }
97
98 /**
99  * usage
100 */
101
```



The screenshot shows the Android Studio code editor with the file `Database.java` open. The code implements a database helper class for managing stocks. It includes methods for updating or adding stocks, inserting products, and getting a list of updated or added stocks. The code uses SQLiteCursor and ContentValues objects to interact with the database.

```
1 usage
95     public List<Stock> getUpdatedOrAddedStocks() {
96         List<Stock> stocks = new ArrayList<>();
97         String[] columns={<code>"quantite", "produit_id", "magasin_id", "nom_produit", "nom_magasin"</code>};
98         SQLiteDatabase db = getReadableDatabase();
99         Cursor c = db.query(<code>"stocks_update_add", columns, </code>selection: null, <code>selectionArgs: null, </code>groupBy: null, <code>having: null, </code>orderBy: null);
100        if (c!=null) {
101            while (c.moveToFirst()) {
102                Integer qte = c.getInt(c.getColumnIndexOrThrow(<code>"quantite"</code>));
103                Integer p_id = c.getInt(c.getColumnIndexOrThrow(<code>"produit_id"</code>));
104                Integer m_id = c.getInt(c.getColumnIndexOrThrow(<code>"magasin_id"</code>));
105                String n_p = c.getString(c.getColumnIndexOrThrow(<code>"nom_produit"</code>));
106                String n_m = c.getString(c.getColumnIndexOrThrow(<code>"nom_magasin"</code>));
107                Stock stock =new Stock(new StockPK(p_id, m_id));
108                stock.setQuantite(qte);
109                Magasin mag = new Magasin(m_id);
110                mag.setNom(n_m);
111                Produit prod = new Produit(p_id);
112                prod.setNom(n_p);
113                stock.setMagasinId(mag);
114                stock.setProduitId(prod);
115                stocks.add(stock);
116            }
117        }
118        c.close();
    }
```

The screenshot shows a Java code editor with the file `Database.java` open. The code contains three main methods: `removeAfterSyncDelete`, `removeAfterSyncAddOrUpdate`, and `save`. Each method uses a SQLite database to perform operations on a table named `stocks`.

```
public void removeAfterSyncDelete(Integer produitId, Integer magasinId) {
    String str[] = new String[2];
    str[0] = produitId.toString();
    str[1] = magasinId.toString();
    SQLiteDatabase db = getWritableDatabase();
    db.delete("stocks_delete", "produit_id=? and magasin_id=?", str);
    db.close();
}

public void removeAfterSyncAddOrUpdate(Integer produitId, Integer magasinId) {
    String str[] = new String[2];
    str[0] = produitId.toString();
    str[1] = magasinId.toString();
    SQLiteDatabase db = getWritableDatabase();
    db.delete("stocks_update_add", "produit_id=? and magasin_id=?", str);
    db.close();
}

public void save(Stock s) {
    ContentValues cv = new ContentValues();
    cv.put("quantite", s.getQuantite());
    cv.put("produit_id", s.getStockPK().getProduitId());
    cv.put("magasin_id", s.getStockPK().getMagasinId());
}
```

```
1 usage
141     @  public void save(Stock s) {
142         ContentValues cv = new ContentValues();
143         cv.put("quantite", s.getQuantite());
144         cv.put("produit_id", s.getStockPK().getProduitId());
145         cv.put("magasin_id", s.getStockPK().getMagasinId());
146         cv.put("nom_produit", s.getProduitId().getNom());
147         cv.put("nom_magasin", s.getMagasinId().getNom());
148         SQLiteDatabase db = getWritableDatabase();
149         db.insert("stocks", nullColumnHack: null, cv);
150         db.close();
151     }
152
153     @  public void addStock(Stock s) {
154         ContentValues cv = new ContentValues();
155         cv.put("quantite", s.getQuantite());
156         cv.put("produit_id", s.getStockPK().getProduitId());
157         cv.put("magasin_id", s.getStockPK().getMagasinId());
158         cv.put("nom_produit", s.getProduitId().getNom());
159         cv.put("nom_magasin", s.getMagasinId().getNom());
160         SQLiteDatabase db = getWritableDatabase();
161         db.insert("stocks", nullColumnHack: null, cv);
162         cv.put("quantite", s.getQuantite());
```

The screenshot shows the `Database.java` file in the Android Studio code editor. The code implements methods for inserting, deleting, and updating stock data in an SQLite database. The `insert` method inserts a new row into the `stocks_update_add` table. The `delete` method deletes rows from the `stocks` table where `produit_id` and `magasin_id` match the provided parameters. The `update` method updates the `stocks_delete` table with the values from a `Stock` object. The code uses `ContentValues` objects to handle the data and `SQLiteOpenHelper` to manage the database connection.

```
165     cv.put("nom_produit", s.getProduitId().getNom());
166     cv.put("nom_magasin", s.getMagasinId().getNom());
167     db.insert( table: "stocks_update_add", nullColumnHack: null, cv);
168     db.close();
169 }
170
171 /**
172  * @param produitId
173  * @param magasinId
174  */
175 public void delete(Integer produitId, Integer magasinId) {
176     String str[] = new String[2];
177     str[0] = produitId.toString();
178     str[1] = magasinId.toString();
179     SQLiteDatabase db = getWritableDatabase();
180     int rowsDeleted = db.delete( table: "stocks", whereClause: "produit_id=? and magasin_id=?", str);
181     boolean isDeleted = rowsDeleted > 0;
182     if (isDeleted) {
183         ContentValues cv = new ContentValues();
184         cv.put("produit_id", produitId);
185         cv.put("magasin_id", magasinId);
186         db.insert( table: "stocks_delete", nullColumnHack: null, cv);
187     }
188     db.close();
189 }
190
191 /**
192  * @param s
193  */
194 public void update(Stock s) {
195     ContentValues cv = new ContentValues();
196     cv.put("nom_produit", s.getProduitId().getNom());
197     cv.put("nom_magasin", s.getMagasinId().getNom());
198 }
```

```
Database.java
188 ContentValues cv = new ContentValues();
189 cv.put("quantite", s.getQuantite());
190 SQLiteDatabase db = getWritableDatabase();
191 String whereClause = "product_id = ? AND magasin_id = ?";
192 String[] whereArgs = {String.valueOf(s.getStockPK().getProduitId()), String.valueOf(s.getStockPK().getMagasinId())};
193 int rowsUpdated = db.update( table: "stocks", cv, whereClause, whereArgs);
194 if (rowsUpdated > 0) {
195     cv.put("quantite", s.getQuantite());
196     cv.put("product_id", s.getStockPK().getProduitId());
197     cv.put("magasin_id", s.getStockPK().getMagasinId());
198     cv.put("nom_produit", s.getProduitId().getNom());
199     cv.put("nom_magasin", s.getMagasinId().getNom());
200     db.insert( table: "stocks_update_add", nullColumnHack: null, cv);
201 }
202 db.close();
203 }
204
205 1 usage
206 public String checkStockQuantities() {
207     String message="";
208     String[] columns={"quantite","nom_produit","nom_magasin"};
209     SQLiteDatabase db = getReadableDatabase();
210     Cursor c = db.query( table: "stocks", columns, selection: null, selectionArgs: null, groupBy: null, having: null, orderBy: null);
211     if (c!=null) {
212         while (c.moveToNext()) {
```

The screenshot shows the Android Studio code editor with the file `Database.java` open. The code implements a method `checkStockQuantities()` that queries a SQLite database to check stock quantities. It uses a cursor to iterate through the results and build a message string. The code is annotated with usage counts (e.g., 1 usage for `checkStockQuantities()`, 2 usages for `getReadableDatabase()`, etc.). The right side of the screen shows the Device Manager, Grade, Notifications, and Device File Explorer.

```
204
205     1 usage
206     public String checkStockQuantities() {
207         String message="";
208         String[] columns={"quantite","nom_produit","nom_magasin"};
209         SQLiteDatabase db = getReadableDatabase();
210         Cursor c = db.query( table: "stocks", columns, selection: null, selectionArgs: null, groupBy: null, having: null, orderBy: null);
211         if (c!=null) {
212             while (c.moveToNext()) {
213                 Integer qte = c.getInt(c.getColumnIndexOrThrow( s: "quantite"));
214                 String n_p = c.getString(c.getColumnIndexOrThrow( s: "nom_produit"));
215                 String n_m = c.getString(c.getColumnIndexOrThrow( s: "nom_magasin"));
216                 if (qte < 2) {
217                     message += "Le stock pour " + n_p + " à la boutique " + n_m + "\n";
218                 }
219             }
220             c.close();
221         }
222         db.close();
223         return message;
224     }

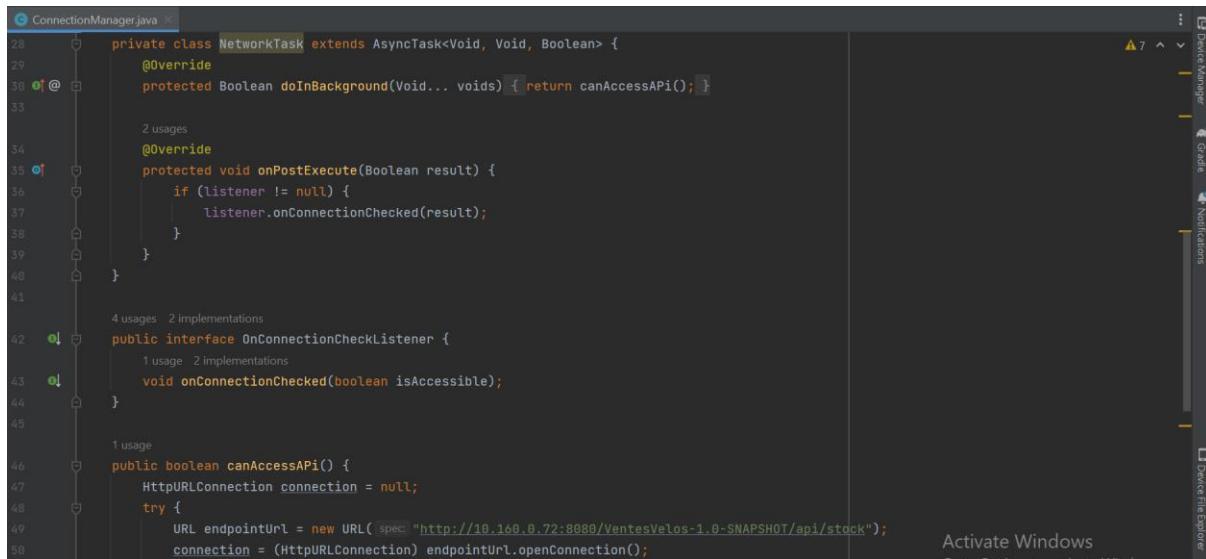
```

Nous créons aussi une classe **ConnectionManager** qui sera utilisé pour vérifier si l'accès au serveur est possible ou non. Pour cela on utilise AsyncTask qui permet d'exécuter cette tâche en arrière-plan évitant le blocage de l'interface utilisateur. On vérifie l'accès à l'aide de la méthode canAccessAPi() qui tente d'ouvrir une connection vers l'endpoint stock dans notre service web. Si la réponse est 200 alors la connexion est possible sinon on renvoie false.

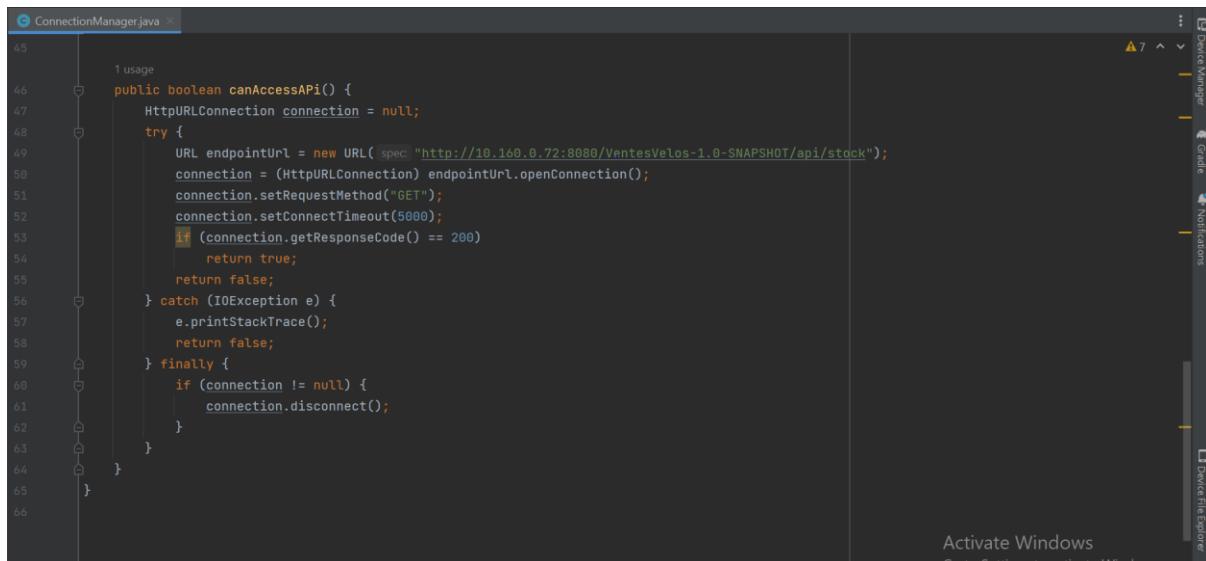
The screenshot shows the Android Studio code editor with the file `ConnectionManager.java` open. The code defines a class `ConnectionManager` that uses an `AsyncTask` to check API access. It has private fields for context and a listener, and methods to set the listener and perform the check. The code is annotated with usage counts (e.g., 10 usages for `ConnectionManager`, 2 usages for `setOnConnectionCheckListener`, etc.). The right side of the screen shows the Device Manager, Grade, Notifications, and Device File Explorer.

```
1 package com.example.bikeshopmobile.utils;
2
3 import ...
4
5 10 usages
6 public class ConnectionManager {
7
8     1 usage
9     private Context context;
10    3 usages
11    private OnConnectionCheckListener listener;
12
13    2 usages
14    public ConnectionManager(Context context) { this.context = context; }
15
16    2 usages
17    public void setOnConnectionCheckListener(OnConnectionCheckListener listener) {
18        this.listener = listener;
19    }
20
21    2 usages
22    public void checkApiAccess() { new NetworkTask().execute(); }
23
24    1 usage
25    private class NetworkTask extends AsyncTask<Void, Void, Boolean> {
26
27
28

```



```
ConnectionManager.java
28 private class NetworkTask extends AsyncTask<Void, Void, Boolean> {
29     @Override
30     protected Boolean doInBackground(Void... voids) { return canAccessAPi(); }
31
32     2 usages
33     @Override
34     protected void onPostExecute(Boolean result) {
35         if (listener != null) {
36             listener.onConnectionChecked(result);
37         }
38     }
39
40     4 usages 2 implementations
41     public interface OnConnectionCheckListener {
42         1 usage 2 implementations
43         void onConnectionChecked(boolean isAccessible);
44     }
45
46     1 usage
47     public boolean canAccessAPi() {
48         HttpURLConnection connection = null;
49         try {
50             URL endpointUrl = new URL("http://10.160.0.72:8080/VentesVélos-1.0-SNAPSHOT/api/stock");
51             connection = (HttpURLConnection) endpointUrl.openConnection();
52             connection.setRequestMethod("GET");
53             connection.setConnectTimeout(5000);
54             if (connection.getResponseCode() == 200)
55                 return true;
56             return false;
57         } catch (IOException e) {
58             e.printStackTrace();
59             return false;
60         } finally {
61             if (connection != null)
62                 connection.disconnect();
63         }
64     }
65 }
```



```
ConnectionManager.java
45
46     1 usage
47     public boolean canAccessAPi() {
48         HttpURLConnection connection = null;
49         try {
50             URL endpointUrl = new URL("http://10.160.0.72:8080/VentesVélos-1.0-SNAPSHOT/api/stock");
51             connection = (HttpURLConnection) endpointUrl.openConnection();
52             connection.setRequestMethod("GET");
53             connection.setConnectTimeout(5000);
54             if (connection.getResponseCode() == 200)
55                 return true;
56             return false;
57         } catch (IOException e) {
58             e.printStackTrace();
59             return false;
60         } finally {
61             if (connection != null)
62                 connection.disconnect();
63         }
64     }
65 }
```

Nous créons ensuite une classe **StockService** pour la gestion des opérations avec stock. Ce service va implémenter l'interface **OnConnectionCheckListener** de **ConnectionManager** afin de vérifier si l'accès au service est disponible. Si tel est le cas il récupère les entités et les enregistre localement dans la base de données SQLite. Ce service étant également responsable des notifications il s'exécute périodiquement raison pour laquelle dans la méthode **initDB**, si la base de données existe on la supprime avant de récréer pour ne pas se retrouver avec des duplicitats.

The screenshot shows the StockService.java file in the Android Studio code editor. The code defines a service that interacts with a database and an API. Several annotations are present: a warning icon at line 44, a red circle at line 44, a yellow exclamation mark at line 44, and a green checkmark at line 44. A tooltip for 'connectionManager' is visible at line 40. The code editor has a light theme with dark syntax highlighting. The status bar at the bottom right shows 'Activate Windows'.

```
1 package com.example.bikeshopmobile.services;
2
3 import ...
33
34 public class StockService extends Service implements ConnectionManager.OnConnectionCheckListener{
35
36     private static final int NOTIFICATION_ID = 1;
37
38     private static final String NOTIFICATION_CHANNEL_ID = "stock_channel";
39
40     private static final String NOTIFICATION_CHANNEL_NAME = "Stock Notification Channel";
41
42     private ApilInterface apiInterface;
43
44     private ConnectionManager connectionManager;
45
46     String msg="";
47
48     @Override
49     public void onCreate(){
50         super.onCreate();
51         apiInterface = RetrofitClient.getRetrofitInstance().create(ApiInterface.class);
52         connectionManager = new ConnectionManager( context: this);
53     }
54 }
```

The screenshot shows the StockService.java file in the Android Studio code editor. The code implements the ConnectionManager.OnConnectionCheckedChangeListener interface. It overrides the onConnectionChecked() method to handle database initialization and stock quantity checking. It also overrides the onStartCommand() method to create a notification channel and notification.

```
44     public void onCreate(){
45         super.onCreate();
46         apiInterface = RetrofitClient.getRetrofitInstance().create(ApiInterface.class);
47         connectionManager = new ConnectionManager( context: this );
48         connectionManager.setOnConnectionCheckListener(this);
49         connectionManager.checkApiAccess();
50     }
51
52     1 usage
53     @Override
54     public void onConnectionChecked(boolean isAccessible) {
55         if (isAccessible) {
56             initDB();
57         } else {
58             Database db = new Database(getApplicationContext(), name: "BikeShop", factory: null, version: 1);
59             String msg = db.checkStockQuantities();
60             showStockNotification(msg);
61         }
62     }
63
64     4 usages
65     @Override
66     public int onStartCommand(Intent intent, int flags, int startId) {
67         createNotificationChannel();
68         Notification notification = createForegroundNotification();
69     }
70 
```

The screenshot shows the `StockService.java` file in the Android Studio code editor. The code implements a service with methods for handling commands and checking stock quantities. A code inspection has been run, with several issues highlighted:

- Line 64: `createNotificationChannel();` is marked with a red circle and a warning message: "Method 'createNotificationChannel()' is never used".
- Line 78: `onResponse` is marked with a red circle and a warning message: "Method 'onResponse(Call<List<Stock>>, Response<List<Stock>>)' is never used".
- Line 85: `msg == ""` is marked with a red circle and a warning message: "Comparison with itself".

The code editor also displays the following information:

- Top right corner: **△ 13 ✓ 18**
- Right sidebar: Device Manager, Grade, Notifications, Device File Explorer.
- Bottom right corner: **Activate Windows**

```
StockService.java x
63
64 Override
65     public int onStartCommand(Intent intent, int flags, int startId) {
66         createNotificationChannel();
67         Notification notification = createForegroundNotification();
68         startForeground(NOTIFICATION_ID, notification);
69
70         return START_STICKY;
71     }
72
73     2 usages
74     private void checkQuantities() {
75         Database db = new Database(getApplicationContext(), name: "BikeShop", factory: null, version: 1);
76         Call<List<Stock>> call = apiInterface.getStocks();
77         call.enqueue(
78             new Callback<List<Stock>>() {
79                 @Override
80                 public void onResponse(Call<List<Stock>> call, Response<List<Stock>> response) {
81                     if (response.isSuccessful()) {
82                         for (Stock s: response.body()) {
83                             db.save(s);
84                             if (s.getQuantite() < 2) {
85                                 msg += "Le stock pour " + s.getProduitId().getNom() + " à la boutique " + s.getMagasinId().getNom() + "\n";
86                             }
87                         }
88                     }
89                     if (msg == "") {
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
778
779
779
780
781
782
783
784
785
786
787
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
968
969
969
970
971
972
973
974
975
976
977
978
978
979
979
980
981
982
983
984
985
986
987
987
988
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1047
1048
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1087
1088
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1097
1097
1098
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1147
1148
1148
1149
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1167
1168
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1177
1178
1178
1179
1179
1180
1181
1182
1183
1184
1185
1186
1187
1187
1188
1188
1189
1189
1190
1191
1192
1193
1194
1195
1196
1197
1197
1198
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1217
1218
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1227
1228
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1237
1238
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1247
1248
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1257
1257
1258
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1267
1268
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1277
1278
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1287
1287
1288
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1297
1297
1298
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1317
1318
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1327
1328
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1337
1338
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1347
1348
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1357
1358
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1367
1368
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1377
1378
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1387
1387
1388
1388
1389
1389
1390
1391
1392
1393
1394
1395
1396
1397
1397
1398
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1407
1408
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1417
1418
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1427
1428
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1437
1438
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1447
1447
1448
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1457
1458
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1467
1468
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1477
1478
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1487
1487
1488
1488
1489
1489
1490
1491
1492
1493
1494
1495
1496
1497
1497
1498
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1507
1508
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1517
1518
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1527
1528
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1537
1538
1538
1539
1539
1540
1541
1542
1543
1544
1545
1546
1547
1547
1548
1548
1549
1549
1550
1551
1552
1553
1554
1555
1556
1557
1557
1558
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1567
1568
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1577
1578
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1587
1587
1588
1588
1589
1589
1590
1591
1592
1593
1594
1595
1596
1597
1597
1598
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1607
1608
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1617
1618
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1627
1628
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1637
1638
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1647
1647
1648
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1657
1657
1658
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1667
1668
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1677
1678
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1687
1688
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1697
1697
1698
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1707
1708
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1717
1718
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1727
1728
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1737
1738
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1747
1748
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1757
1758
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1767
1768
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1777
1778
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1787
1788
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1797
1798
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1807
1808
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1817
1818
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1827
1828
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1837
1838
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1847
1848
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1857
1858
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1867
1868
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1877
1878
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1887
1888
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1897
1897
1898
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1907
1908
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1917
1918
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1927
1928
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1937
1938
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1947
1948
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1957
1958
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1967
1968
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1977
1978
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1987
1988
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1997
1998
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2007
2008
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2017
2018
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2027
2028
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2037
2038
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2047
2048
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2057
2058
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2067
2068
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2077
2078
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2087
2088
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2097
2097
2098
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2107
2108
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2117
2118
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2127
2128
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2137
2138
2138
2139
2139
2140
2141
2142
2143
2144
2145
2146
2147
2147
2148
2148
2149
2149
2150
2151
2152
2153
2154
2155
2156
2157
2157
2158
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2167
2168
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2177
2178
2178
2179
2179
2180
2181
2182
2183
2184
2185
2186
2187
2187
2188
2188
2189
2189
2190
2191
2192
2193
2194
2195
2196
2197
2197
2198
2198
2199
2199
2200
2201
2202
2203
2204
2205
2206
2207
2207
2208
2208
2209
2209
2210
```

```
86     if (msg!="") {
87         showStockNotification(msg);
88         msg="";
89     } else {
90         Log.d( tag: "StockService" , msg: "La récupération des données a échoué");
91     }
92 }
93
94
95     @Override
96     public void onFailure(Call<List<Stock>> call, Throwable t) {
97         Log.d( tag: "StockService" , msg: "Error "+ t);
98     }
99 }
100
101
102
103
104
105     ↑ usage
106     private void initDB() {
107         String databaseName = "BikeShop";
108         String databasePath = getApplicationContext().getDatabasePath(databaseName).getPath();
109         File databaseFile = new File(databasePath);
110         if (databaseFile.exists()) {
```

Activate Windows

```
101
102
103
104
105     ↑ usage
106     private void initDB() {
107         String databaseName = "BikeShop";
108         String databasePath = getApplicationContext().getDatabasePath(databaseName).getPath();
109         File databaseFile = new File(databasePath);
110         if (databaseFile.exists()) {
111             boolean isDeleted = databaseFile.delete();
112             if (isDeleted) {
113                 checkQuantities();
114                 Log.d( tag: "StockService" , msg: "Base de données initialisée");
115             } else {
116                 Log.d( tag: "StockService" , msg: "Pas de base de donnée");
117             }
118         } else {
119             checkQuantities();
120             Log.d( tag: "StockService" , msg: "Base de données initialisée");
121         }
122
123
124 }
```

Activate Windows

Au niveau des captures, nous voyons des éléments pour la gestion des notifications mais nous y reviendrons plus tard.

On utilise donc la base de données ainsi créé au niveau de nos activités pour effectuer les modifications en locale et les reporter lorsque la connexion est disponible.

ListStocks

Au niveau de l'activité ListStocks qui implémente l'interface OnConnectionCheckedChangeListener, nous utilisons créons une instance de Database pour accéder à la base de données.

On créer une méthode getStocks() dans laquelle on récupérer l'ensemble des enregistrements contenus dans notre base de données en utilisant la méthode getAllEntries() que l'on affiche dans le ListView.

On override la méthode `onConnectionChecked` de sorte à ce que lorsque la connection est disponible on récupère les stocks supprimés `getDeleteStocks()` en utilisant la méthode et les stocks modifiés ou ajoutés en utilisant la méthode `getAddedOrUpdatedStock()`. On communique ensuite avec le service pour utiliser les endpoints appropriés et synchroniser les données au niveau du serveur en prenant soin lorsque la synchronisation réussie de supprimer les données qui se trouvaient précédemment au niveau de la base locale et de les remplacer également par les nouvelles.

Au niveau de la méthode deleteStock, on utilise la méthode delete() qui se trouve dans notre base de données locale puis on redémarre l'activité afin de répercuter la suppression dans la base et synchroniser si la connection est disponible.

```
1 package com.example.bikeshopmobile.activities.stock;
2
3 import ...
33
34 public class ListStocks extends AppCompatActivity implements ConnectionManager.OnConnectionCheckListener{
35
36     2 usages
37     List<Stock> stockList;
38     2 usages
39     ListView listView;
40     4 usages
41     ApiInterface apiInterface;
42     2 usages
43     Button btn1;
44     2 usages
45     ImageButton btn2;
46     6 usages
47     Database db = new Database( context: this, name: "BikeShop", factory: null, version: 1);
48     3 usages
49     private ConnectionManager connectionManager;
50
51     @Override
52     protected void onCreate(Bundle savedInstanceState) {
53         super.onCreate(savedInstanceState);
54     }
55 }
```

```
64
65     btn2.setOnClickListener(new View.OnClickListener() {
66         @Override
67         public void onClick(View view) {
68             startActivity(new Intent(getApplicationContext(), HomeActivity.class));
69         }
70     });
71 }
72
73 /**
74  * Get stocks from database
75 */
76 private void getStocks() {
77     stockList = db.getAllEntries();
78     ArrayAdapter<Stock> adapter = new ArrayAdapter<>(context, R.layout.list_tile_layout, R.id.title, stockList) {
79         @Override
80         public View getView(int position, View convertView, ViewGroup parent) {
81             View view = super.getView(position, convertView, parent);
82             if (view == null) {
83                 LayoutInflator inflater = LayoutInflator.from(getContext());
84                 view = inflater.inflate(R.layout.list_tile_layout, parent, false);
85             }
86             Stock item = getItem(position);
87             if (item != null) {
88                 TextView elementNameTextView = view.findViewById(R.id.title);
```

Activate Windows
Go to Settings to activate Windows.

```
81             view = inflater.inflate(R.layout.list_tile_layout, parent, false);
82
83             Stock item = getItem(position);
84
85             if (item != null) {
86                 TextView elementNameTextView = view.findViewById(R.id.title);
87                 TextView detailsTextView = view.findViewById(R.id.details);
88                 ImageView trashcanIcon = view.findViewById(R.id.trashcanIcon);
89
90                 elementNameTextView.setText(item.getProduitId().getNom().substring(0, 10));
91                 detailsTextView.setText("Magasin: " + item.getMagasinId().getNom().substring(0, 10) + " Quantite: " + item.getQuantite());
92
93                 view.setOnClickListener(new View.OnClickListener() {
94                     @Override
95                     public void onClick(View v) {
96                         Intent intent = new Intent(getApplicationContext(), DetailsStocks.class);
97                         intent.putExtra("selectedStock", item);
98                         startActivity(intent);
99                     }
100                });
101
102                trashcanIcon.setOnClickListener(new View.OnClickListener() {
103                    @Override
104                    public void onClick(View v) {
105                        deleteStock(item.getStockPK().getMagasinId(), item.getStockPK().getProduitId());
```

Activate Windows
Go to Settings to activate Windows.

```
106                     }
107                 });
108             }
109         }
110         return view;
111     };
112     listView.setAdapter(adapter);
113 }
114
115 /**
116  * Delete stock
117  */
118 private void deleteStock(Integer magasin_id, Integer produit_id) {
119     db.delete(magasin_id, produit_id);
120     startActivity(new Intent(getApplicationContext(), ListStocks.this, ListStocks.class));
121 }
122
123 /**
124  * Check connection
125  */
126 @Override
127 public void onConnectionChecked(boolean isAccessible) {
128     if (isAccessible) {
129         List<Stock> deletedStocks = db.getDeletedStocks();
130         if (deletedStocks != null) {
131             for (Stock s: deletedStocks) {
```

Activate Windows
Go to Settings to activate Windows.

```
1 usage
122     @Override
123     public void onConnectionChecked(boolean isAccessible) {
124         if (isAccessible) {
125             List<Stock> deletedStocks = db.getDeletedStocks();
126             if (deletedStocks!=null) {
127                 for (Stock s: deletedStocks) {
128                     Call<Reponse> call = apiInterface.deleteStock(s.getStockPK().getMagasinId(), s.getStockPK().getProduitId());
129                     call.enqueue(new Callback<Reponse>() {
130                         @Override
131                         public void onResponse(Call<Reponse> call, Response<Reponse> response) {
132                             db.removeAfterSyncDelete(s.getStockPK().getProduitId(), s.getStockPK().getMagasinId());
133                         }
134
135                         @Override
136                         public void onFailure(Call<Reponse> call, Throwable t) {
137                             Toast.makeText(context, "La synchronisation a échoué", Toast.LENGTH_SHORT);
138                         }
139                     });
140                 }
141             }
142             List<Stock> stocks = db.getUpdatedOrAddedStocks();
143             if (stocks!=null) {
144                 for (Stock s: stocks) {
145                     Call<Stock> call = apiInterface.addOrUpdateStock(s);
```

Activate Windows
Go to Settings to activate Windows.

```
142     List<Stock> stocks = db.getUpdatedOrAddedStocks();
143     if (stocks!=null) {
144         for (Stock s: stocks) {
145             Call<Stock> call = apiInterface.addOrUpdateStock(s);
146             call.enqueue(new Callback<Stock>() {
147                 @Override
148                 public void onResponse(Call<Stock> call, Response<Stock> response) {
149                     db.removeAfterSyncAddOrUpdate(s.getStockPK().getProduitId(), s.getStockPK().getMagasinId());
150                 }
151
152                 @Override
153                 public void onFailure(Call<Stock> call, Throwable t) {
154                     Toast.makeText(context, "La synchronisation a échoué", Toast.LENGTH_SHORT);
155                 }
156             });
157         }
158     } else {
159         Log.d("ListStock", "Connection Not Available");
160     }
161     getStocks();
162 }
163 }
```

Activate Windows
Go to Settings to activate Windows.

AddStock/DetailsStocks

Au niveau des activités AddStock et DetailsStocks, on a le même principe de fonctionnement où nos méthodes addStock() et updateStock() accède au méthode save() et update() de la base de données pour reporter les modifications au niveau de la base de données locale. Après cela, elle redirige vers l’activité ListStock où OnConnectionCheckListener sera exécutée et la synchronisation avec le serveur effectuée si possible.

```
58     }
59 }
60 }
61 });
62 }
63 }
64 btn2.setOnClickListener(new View.OnClickListener() {
65     @Override
66     public void onClick(View view) {
67         startActivity(new Intent(getApplicationContext(), DetailsStocks.this, ListStocks.class));
68     }
69 }
70 }
71 }
72 private void updateStock(Stock stock) {
73     db.update(stock);
74     startActivity(new Intent(getApplicationContext(), DetailsStocks.this, ListStocks.class));
75 }
76 }
```

Activate Windows
Go to Settings to activate Windows

```
177     private void getProducts() {
178         Intent serviceIntent = new Intent(getApplicationContext(), ApiService.class);
179         serviceIntent.setAction("getProducts");
180         startService(serviceIntent);
181     }
182
183     private void getMagasins() {
184         Intent serviceIntent = new Intent(getApplicationContext(), ApiService.class);
185         serviceIntent.setAction("getMagasins");
186         startService(serviceIntent);
187     }
188
189     private void addStock(Stock stock) {
190         db.addStock(stock);
191         startActivity(new Intent(getApplicationContext(), AddStock.this, ListStocks.class));
192     }
193 }
194 }
```

Activate Windows

Nous montrerons le fonctionnement en détail dans les démos car il est difficile à capturer juste par le biais d'image.

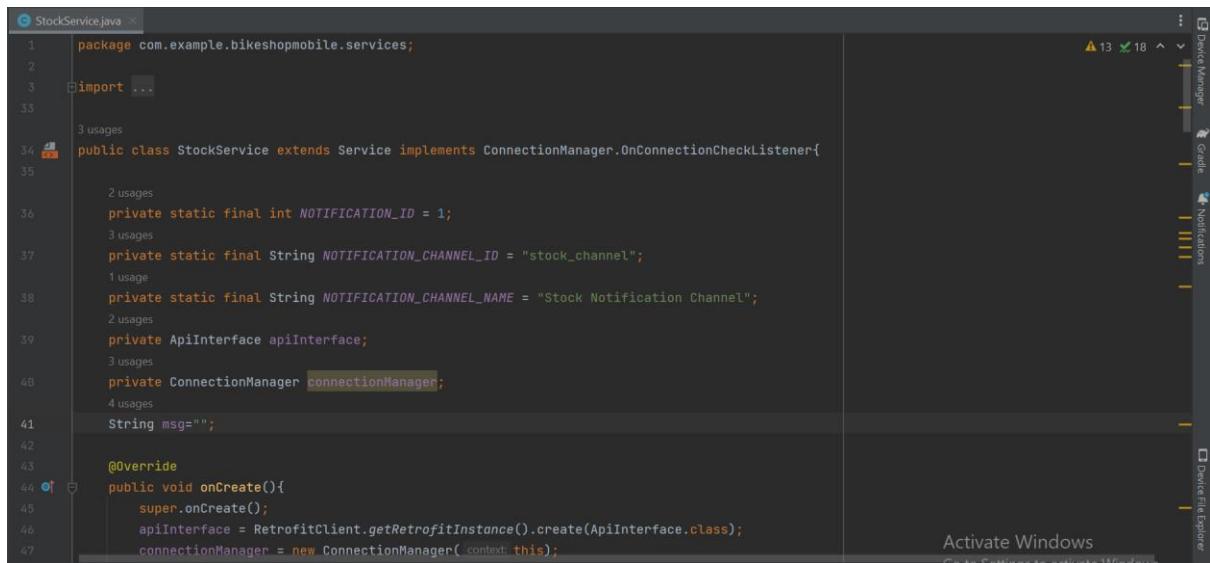
III. Envoi de notifications

Nous utilisons la classe **StockService** précédemment créée pour gérer les notifications

Nous définissons les constantes pour l'ID and le nom du canal de notification pour identifier la notification lorsqu'on met à jour ou que l'on supprime.

Nous créons la méthode `showStockNotification()` qui est responsable pour afficher les notifications. Lorsqu'appelé, elle crée une notification avec le message. Elle crée une instance du `NotificationManager` pour les notifications et le `NotificationCompat.Builder` pour construire la notification. On utilise également un `BigTextStyle` est utilisé pour agrandir les notifications. Un intent est crée pour ouvrir l'activité notification lorsque cliqué.

On crée également une méthode `createNotificationChannel()` qui crée le canal de notification et une méthode `createForegroundNotification()` qui crée la notification lorsque le service démarre. La notification est affichée lorsque le service s'exécute en arrière-plan. Elle inclue une icône, le titre et le contenu.



```
1 package com.example.bikeshopmobile.services;
2
3 import ...
33
34 public class StockService extends Service implements ConnectionManager.OnConnectionCheckListener{
35
36     private static final int NOTIFICATION_ID = 1;
37
38     private static final String NOTIFICATION_CHANNEL_ID = "stock_channel";
39
40     private static final String NOTIFICATION_CHANNEL_NAME = "Stock Notification Channel";
41
42     private ApiInterface apiInterface;
43
44     private ConnectionManager connectionManager;
45
46     String msg="";
47
48     @Override
49     public void onCreate(){
50         super.onCreate();
51         apiInterface = RetrofitClient.getRetrofitInstance().create(ApiInterface.class);
52         connectionManager = new ConnectionManager(context: this);
53     }
54 }
```

```
125 private void showStockNotification(String message) {
126     NotificationManager notificationManager = (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
127     NotificationCompat.Builder builder = new NotificationCompat.Builder(context, NOTIFICATION_CHANNEL_ID)
128         .setSmallIcon(R.drawable.ic_warning)
129         .setContentTitle("Faible stock")
130         .setContentText("Les stocks suivant sont faibles\n " + message)
131         .setPriority(NotificationCompat.PRIORITY_DEFAULT);
132
133     NotificationCompat.BigTextStyle bigTextStyle = new NotificationCompat.BigTextStyle();
134     bigTextStyle.setBigContentTitle("Faible stock");
135     bigTextStyle.bigText("Les stocks suivant sont faibles\n " + message);
136     builder.setStyle(bigTextStyle);
137
138     Intent notificationIntent = new Intent(packageContext, NotificationActivity.class);
139     notificationIntent.putExtra(name, "Message", message);
140     PendingIntent pendingIntent = PendingIntent.getActivity(context, requestCode, notificationIntent, flags);
141     builder.setContentIntent(pendingIntent);
142
143     notificationManager.notify(NOTIFICATION_ID, builder.build());
144 }
145
146 1 usage
147 private void createNotificationChannel() {
148     if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
149         NotificationChannel channel = new NotificationChannel(
150             NOTIFICATION_CHANNEL_ID,
151             NOTIFICATION_CHANNEL_NAME,
152             NotificationManager.IMPORTANCE_DEFAULT
153         );
154         NotificationManager notificationManager = getSystemService(NotificationManager.class);
155         notificationManager.createNotificationChannel(channel);
156     }
157
158 1 usage
159 @ private Notification createForegroundNotification() {
160     NotificationCompat.Builder builder = new NotificationCompat.Builder(context, NOTIFICATION_CHANNEL_ID)
161         .setSmallIcon(R.drawable.ic_warning)
162         .setContentTitle("Stock Service")
163         .setContentText("Le service s'exécute en arrière plan")
164         .setPriority(NotificationCompat.PRIORITY_DEFAULT);
165
166     Intent notificationIntent = new Intent(packageContext, HomeActivity.class);
167     PendingIntent pendingIntent = PendingIntent.getActivity(context, requestCode, notificationIntent, flags);
168     builder.setContentIntent(pendingIntent);
169
170     return builder.build();
171 }
172
173 @Nullable
174 @Override
175 public IBinder onBind(Intent intent) {
176     return null;
177 }
```

Activate Windows

```
146 private void createNotificationChannel() {
147     if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
148         NotificationChannel channel = new NotificationChannel(
149             NOTIFICATION_CHANNEL_ID,
150             NOTIFICATION_CHANNEL_NAME,
151             NotificationManager.IMPORTANCE_DEFAULT
152         );
153         NotificationManager notificationManager = getSystemService(NotificationManager.class);
154         notificationManager.createNotificationChannel(channel);
155     }
156
157 1 usage
158 @ private Notification createForegroundNotification() {
159     NotificationCompat.Builder builder = new NotificationCompat.Builder(context, NOTIFICATION_CHANNEL_ID)
160         .setSmallIcon(R.drawable.ic_warning)
161         .setContentTitle("Stock Service")
162         .setContentText("Le service s'exécute en arrière plan")
163         .setPriority(NotificationCompat.PRIORITY_DEFAULT);
164
165     Intent notificationIntent = new Intent(packageContext, HomeActivity.class);
166     PendingIntent pendingIntent = PendingIntent.getActivity(context, requestCode, notificationIntent, flags);
167     builder.setContentIntent(pendingIntent);
168
169     return builder.build();
170 }
```

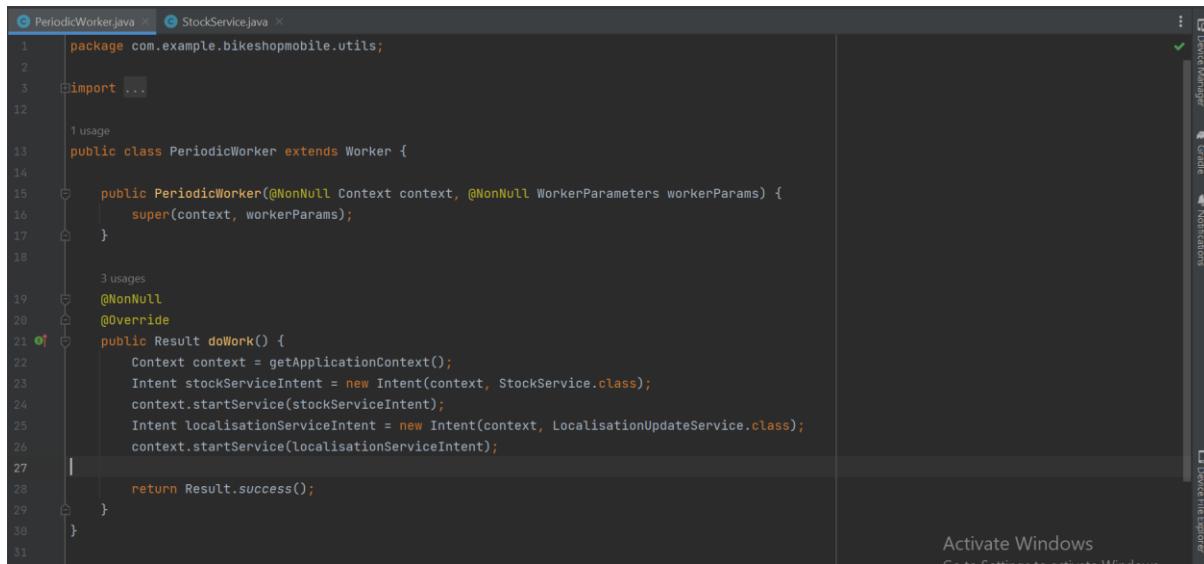
Activate Windows

```
157 1 usage
158 @ private Notification createForegroundNotification() {
159     NotificationCompat.Builder builder = new NotificationCompat.Builder(context, NOTIFICATION_CHANNEL_ID)
160         .setSmallIcon(R.drawable.ic_warning)
161         .setContentTitle("Stock Service")
162         .setContentText("Le service s'exécute en arrière plan")
163         .setPriority(NotificationCompat.PRIORITY_DEFAULT);
164
165     Intent notificationIntent = new Intent(packageContext, HomeActivity.class);
166     PendingIntent pendingIntent = PendingIntent.getActivity(context, requestCode, notificationIntent, flags);
167     builder.setContentIntent(pendingIntent);
168
169     return builder.build();
170 }
171
172 @Nullable
173 @Override
174 public IBinder onBind(Intent intent) {
175     return null;
176 }
```

Activate Windows

Nous créons une classe **PeriodicWorker** qui va exécuter les tâches en arrière-plan, elle étend la class Worker. Dans la méthode doWork() qui constitue le point

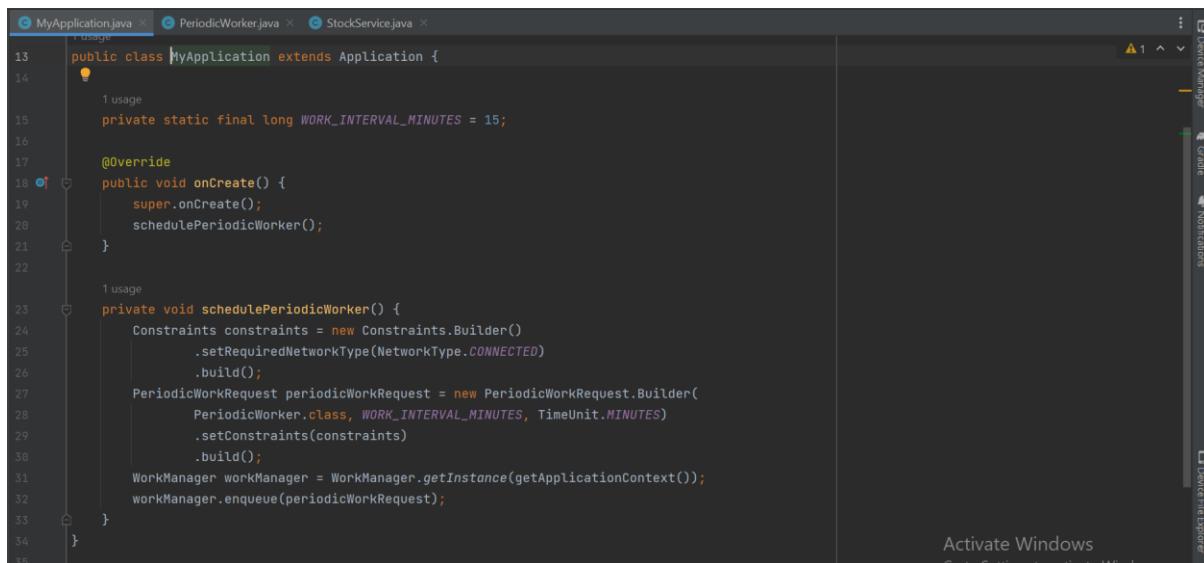
d'entrée des travaux en arrière-plan and démarre le StockService ainsi que le LocalisationUpdateSerivice. Lorsque le travail est effectué avec succès la méthode Result.success() est applé.



```
PeriodicWorker.java
1 package com.example.bikeshopmobile.utils;
2
3 import ...
4
5 1 usage
6 public class PeriodicWorker extends Worker {
7
8     public PeriodicWorker(@NonNull Context context, @NonNull WorkerParameters workerParams) {
9         super(context, workerParams);
10    }
11
12    3 usages
13    @NonNull
14    @Override
15    public Result doWork() {
16        Context context = getApplicationContext();
17        Intent stockServiceIntent = new Intent(context, StockService.class);
18        context.startService(stockServiceIntent);
19        Intent localisationServiceIntent = new Intent(context, LocalisationUpdateService.class);
20        context.startService(localisationServiceIntent);
21
22        return Result.success();
23    }
24
25    }
26
27
28
29
30
31
```

The screenshot shows the PeriodicWorker.java file in the Android Studio code editor. The code defines a PeriodicWorker class that extends the Worker class. It overrides the doWork() method to start two services: StockService and LocalisationUpdateService. The code editor highlights the StockService and LocalisationUpdateService imports with orange. The status bar at the bottom right indicates "Activate Windows".

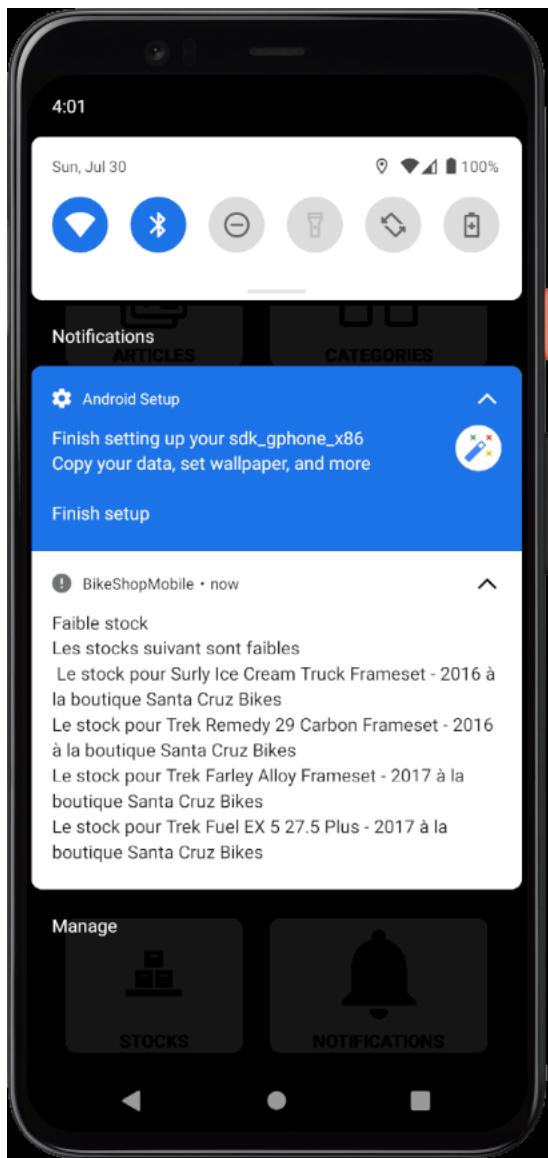
Nous créons également une classe **MyApplication** qui nous permet d'exécuter notre worker chaque 15 minutes.



```
MyApplication.java
13 public class MyApplication extends Application {
14
15     1 usage
16     private static final long WORK_INTERVAL_MINUTES = 15;
17
18     @Override
19     public void onCreate() {
20         super.onCreate();
21         schedulePeriodicWorker();
22     }
23
24     1 usage
25     private void schedulePeriodicWorker() {
26         Constraints constraints = new Constraints.Builder()
27             .setRequiredNetworkType(NetworkType.CONNECTED)
28             .build();
29         PeriodicWorkRequest periodicWorkRequest = new PeriodicWorkRequest.Builder(
30             PeriodicWorker.class, WORK_INTERVAL_MINUTES, TimeUnit.MINUTES)
31             .setConstraints(constraints)
32             .build();
33         WorkManager workManager = WorkManager.getInstance(getApplicationContext());
34         workManager.enqueue(periodicWorkRequest);
35     }
36
37 }
```

The screenshot shows the MyApplication.java file in the Android Studio code editor. The code defines a MyApplication class that extends the Application class. It overrides the onCreate() method to call schedulePeriodicWorker(). The schedulePeriodicWorker() method creates a PeriodicWorkRequest for the PeriodicWorker class with an interval of 15 minutes. The code editor highlights the PeriodicWorkRequest import with orange. The status bar at the bottom right indicates "Activate Windows".

Voici le résultat final où nous recevons une notification:

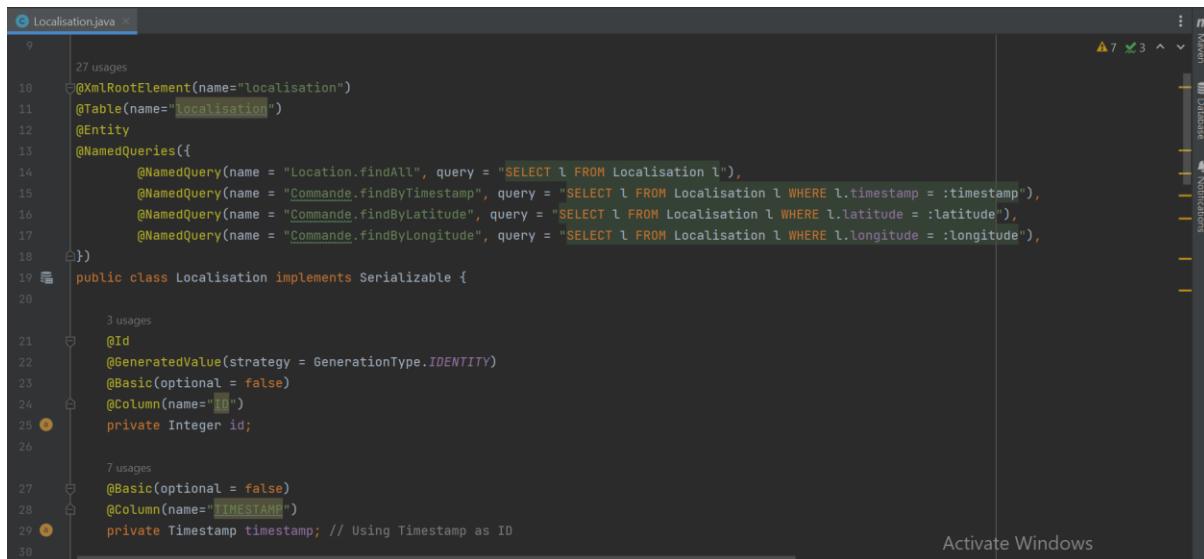


IV. Traçage du téléphone

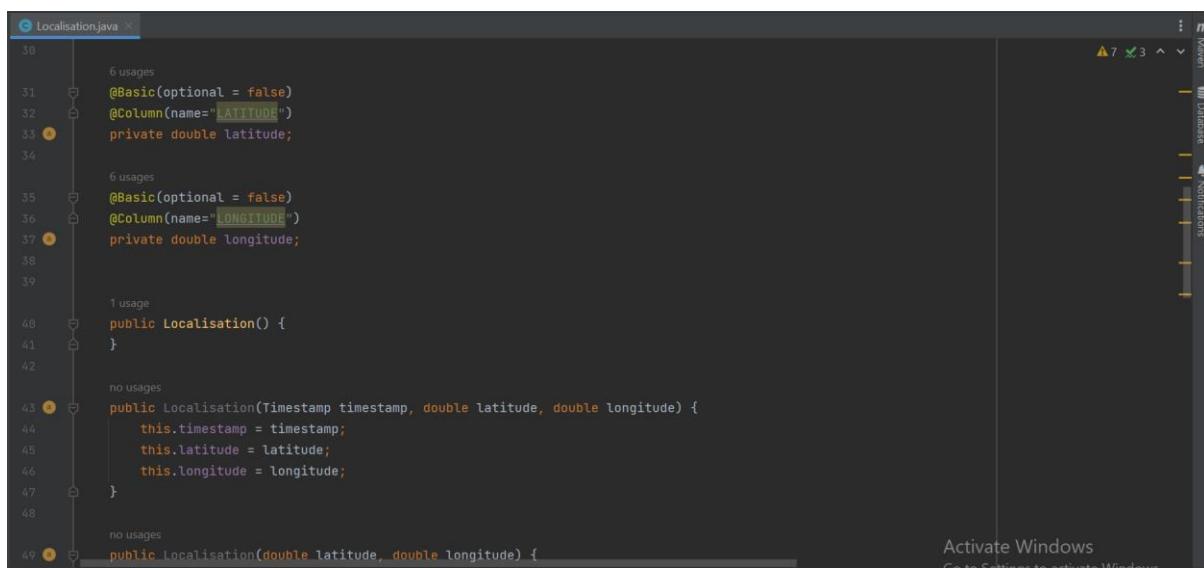
Nous commençons par créer au niveau de notre application JakartaEE une entité Localisation qui va nous permettre de disposer d'une table au niveau de la base de données.

On utilise les annotation @Entity, @Table et @XmlRootElement. On ajoute les attributs id, timestamp, latitude et longitude avec les constructeurs ainsi que les getters et setters.

Pour finir on ajoute également les méthodes hashCode() et toString().



```
Localisation.java
9
10 27 usages
11  @XmlRootElement(name="localisation")
12  @Table(name="localisation")
13  @Entity
14  @NamedQueries({
15      @NamedQuery(name = "Location.findAll", query = "SELECT l FROM Localisation l"),
16      @NamedQuery(name = "Commande.findByTimestamp", query = "SELECT l FROM Localisation l WHERE l.timestamp = :timestamp"),
17      @NamedQuery(name = "Commande.findByLatitude", query = "SELECT l FROM Localisation l WHERE l.latitude = :latitude"),
18      @NamedQuery(name = "Commande.findByLongitude", query = "SELECT l FROM Localisation l WHERE l.longitude = :longitude"),
19  })
20  public class Localisation implements Serializable {
21
22      3 usages
23      @Id
24      @GeneratedValue(strategy = GenerationType.IDENTITY)
25      @Basic(optional = false)
26      @Column(name="ID")
27      private Integer id;
28
29      7 usages
30      @Basic(optional = false)
31      @Column(name="TIMESTAMP")
32      private Timestamp timestamp; // Using Timestamp as ID
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
```



```
Localisation.java
30
31      6 usages
32      @Basic(optional = false)
33      @Column(name="LATITUDE")
34      private double latitude;
35
36      6 usages
37      @Basic(optional = false)
38      @Column(name="LONGITUDE")
39      private double longitude;
40
41
42
43      1 usage
44      public Localisation() {
45
46
47
48
49      no usages
50      public Localisation(Timestamp timestamp, double latitude, double longitude) {
51          this.timestamp = timestamp;
52          this.latitude = latitude;
53          this.longitude = longitude;
54
55      }
56
57      no usages
58      public Localisation(double latitude, double longitude) {
59
60
61
62
63
64
65
66
67
68
69
```

```
Localisation.java
49     public Localisation(double latitude, double longitude) {
50         this.latitude = latitude;
51         this.longitude = longitude;
52     }
53
54     public Integer getId() { return id; }
55
56     public void setId(Integer id) { this.id = id; }
57
58     public Timestamp getTimestamp() { return timestamp; }
59
60     public void setTimestamp(Timestamp timestamp) { this.timestamp = timestamp; }
61
62     7 usages
63     public double getLatitude() { return latitude; }
64
65     4 usages
66     public void setLongitude(double longitude) { this.longitude = longitude; }
67
68     7 usages
69     public double getLongitude() { return longitude; }
70
71     4 usages
72     public void setLatitude(double latitude) { this.latitude = latitude; }
73
74     7 usages
75     public double getTimestamp() { return timestamp; }
76
77     4 usages
78     public void setTimestamp(Timestamp timestamp) { this.timestamp = timestamp; }
79
80     7 usages
81     public Integer getId() { return id; }
82
83     4 usages
84     public void setId(Integer id) { this.id = id; }
85
```

```
Localisation.java
87     public void onPrePersist() {
88         if (timestamp == null) {
89             timestamp = new Timestamp(System.currentTimeMillis());
90         }
91     }
92
93     @Override
94     public boolean equals(Object o) {
95         if (this == o) return true;
96         if (!(o instanceof Localisation)) return false;
97         Localisation that = (Localisation) o;
98         return Double.compare(that.getLatitude(), getLatitude()) == 0 && Double.compare(that.getLongitude(), getLongitude()) == 0 && getId().equal
99     }
100
101    @Override
102    public int hashCode() { return Objects.hash(getId(), getTimestamp(), getLatitude(), getLongitude()); }
103
104    @Override
105    public String toString() {
106        return "Localisation{" +
107            "id=" + id +
108            "timestamp=" + timestamp +
109            ", latitude=" + latitude +
110            ", longitude=" + longitude +
111            '}';
112    }
113
114}
```

```
Localisation.java
74     if (this == o) return true;
75     if (!(o instanceof Localisation)) return false;
76     Localisation that = (Localisation) o;
77     return Double.compare(that.getLatitude(), getLatitude()) == 0 && Double.compare(that.getLongitude(), getLongitude()) == 0 && getId().equal
78 }
79
80     @Override
81     public int hashCode() { return Objects.hash(getId(), getTimestamp(), getLatitude(), getLongitude()); }
82
83     @Override
84     public String toString() {
85         return "Localisation{" +
86             "id=" + id +
87             "timestamp=" + timestamp +
88             ", latitude=" + latitude +
89             ", longitude=" + longitude +
90             '}';
91     }
92
93 }
```

On voit qu'après exécution une table localisation est créée

The screenshot shows a Java code editor and a database management interface side-by-side.

Java Code Editor:

```
1 package com.ventes.velos;
2
3 public class Localisation {
4     ...
5     @Override
6     public boolean equals(Object o) {
7         if (this == o) return true;
8         if (!(o instanceof Localisation)) return false;
9         Localisation that = (Localisation) o;
10        return Double.compare(that.getLatitude(), getLatitude()) == 0 && Double.compare(that.getLongitude(), getLongitude()) == 0;
11    }
12
13    @Override
14    public int hashCode() { return Objects.hash(id, getTimestamp(), getLatitude(), getLongitude()); }
15
16    @Override
17    public String toString() {
18        return "Localisation{" +
19            "id=" + id +
20            ", timestamp=" + timestamp +
21            ", latitude=" + latitude +
22            ", longitude=" + longitude +
23            '}';
24    }
25}
```

Database Browser:

- Database: ventes_velos@localhost
- Tables (11):
 - article_commande
 - categorie
 - client
 - commande
 - employe
 - localisation
 - magasin
 - marque
 - personne
 - produit
 - stock
- sequences (0)
- Database Objects
- Server Objects

On ajoute également la façade et le Mbean

```

1 package sn.ept.ventesvelos.facades;
2
3 import jakarta.ejb.Stateless;
4 import jakarta.persistence.EntityManager;
5 import jakarta.persistence.PersistenceContext;
6 import sn.ept.ventesvelos.entites.Localisation;
7
8 4 usages
9 @Stateless
10 public class LocalisationFacade extends AbstractFacade{
11     1 usage
12     @PersistenceContext(name="velos_PU")
13     private EntityManager em;
14
15     no usages
16     public LocalisationFacade() { super(Localisation.class); }
17
18     @Override
19     protected EntityManager getEntityManager() { return this.em; }
20
21 }
22

```

Activate Windows

```

1 package sn.ept.ventesvelos.mbeans;
2
3 import ...
4
5 no usages
6
7 @Named("localisationMBean")
8 @ViewScoped
9 public class LocalisationMBean implements Serializable {
10
11     5 usages
12     @EJB
13     private LocalisationFacade localisationFacade;
14
15     4 usages
16     private List<Localisation> localisations;
17     10 usages
18     private Localisation selectedLocalisation;
19     9 usages
20     private List<Localisation> selectedLocalisations;
21
22     @PostConstruct
23     public void init() {
24         if (this.localisations == null) {
25             this.localisations = localisationFacade.findAll();
26         }
27     }
28
29 }
30
31
32
33
34
35

```

Activate Windows

Go to Settings to activate Windows

On crée les pages JSF pour voir les localisations

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <ui:composition xmlns="http://www.w3.org/1999/xhtml"
3     xmlns:h="http://xmlns.jcp.org/jsf/html"
4     xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
5     xmlns:p="http://primefaces.org/ui"
6     xmlns:f="http://xmlns.jcp.org/jsf/core"
7     template="welcome.xhtml">
8
9     <ui:define name="content">
10        <div class="card" style="border:none;">
11            <h:form id="form">
12
13                <p:growl id="messages" showDetail="true"/>
14
15                <p:toolbar style="...">
16                    <p:toolbarGroup>
17                        <p:commandButton value="Ajouter" icon="pi pi-plus" actionListener="#{localisationMBean.openNew}"
18                            update=":dialogs:manage-localisation-content" oncomplete="PF('manageLocalisationDialog').show()"
19                            styleClass="ui-button-success" style="...">
20                            <p:resetInput target=":dialogs:manage-localisation-content" />
21                        </p:commandButton>
22                        <p:commandButton id="delete-localisations-button" value="#{localisationMBean.deleteButtonMessage}"
23                            icon="pi pi-trash" actionListener="#{localisationMBean.deleteSelectedLocalisations}"
24                            styleClass="ui-button-danger" disabled="#{!localisationMBean.hasSelectedLocalisations()}" update="@this">

```

Activate Windows

Go to Settings to activate Windows

Bike Shop

localhost:8080/VentesVelos-1.0-SNAPSHOT/localisation-pf.xhtml

Autres favoris

Accueil Articles Categories Marques Clients Commandes Employes Produits Magasins Stocks Localisations

+ Ajouter Supprimer

MARQUES

No records found.

ID ↑ Timestamp ↑ Latitude ↑ Longitude ↑

<< < > >>

Activate Windows
Go to Settings to activate Windows.

Bike Shop

localhost:8080/VentesVelos-1.0-SNAPSHOT/localisation-pf.xhtml

Autres favoris

Accueil Articles Categories Marques Clients Commandes Employes Produits Magasins Stocks Localisations

+ Ajouter Supprimer

MARQUES

No records found.

ID ↑

Latitude *
0.00

Longitude *
0.00

Details Localisations

Enregistrer Annuler

Longitude ↑

Activate Windows
Go to Settings to activate Windows.

Bike Shop

localhost:8080/VentesVelos-1.0-SNAPSHOT/localisation-pf.xhtml

Autres favoris

Accueil Articles Categories Marques Clients Commandes Employes Produits Magasins Stocks Localisations

+ Ajouter Supprimer

MARQUES

ID ↑	Timestamp ↑	Latitude ↑↓	Longitude ↑↓
1	2023-07-30 16:39:46.557	-14.65	17.31

<< < 1 > >>

Activate Windows
Go to Settings to activate Windows.

Bike Shop

localhost:8080/VentesVelos-1.0-SNAPSHOT/localisation-pf.xhtml

Autres favoris

Accueil Articles Categories Marques Clients Commandes Employes Produits Magasins Stocks Localisations

+ Ajouter Supprimer

MARQUES

ID ↑	Timestamp ↑	Latitude ↑↓	Longitude ↑↓
1	2023-07-30 16:39:46.557	-14.65	17.31

Details Localisations

Latitude *
-14.96

Longitude *
17.31

✓ Enregistrer ✖ Annuler

Activate Windows
Go to Settings to activate Windows.

The screenshot shows a web application interface for a bike shop. The top navigation bar includes links for Accueil, Articles, Categories, Marques, Clients, Commandes, Employes, Produits, Magasins, Stocks, and Localisations. Below the navigation is a toolbar with 'Ajouter' (Add) and 'Supprimer' (Delete) buttons. The main content area is titled 'MARQUES' and displays a table with columns: ID ↑, Timestamp ↑, Latitude ↑, and Longitude ↑. A single record is listed: ID 1, Timestamp 2023-07-30 16:39:46.557, Latitude -14.96, and Longitude 17.31. To the right of the table are edit and delete icons. Navigation controls at the bottom include '<<', '<', '1', '>', and '>>'. A watermark for 'Activate Windows' is visible in the background.

This screenshot shows the same web application interface as the first one, but it displays a message stating 'No records found.' instead of a list of records. The rest of the interface, including the navigation bar, toolbar, and watermark, remains identical.

On crée une ressource pour les localisations au niveau de notre service web

```
Localisat&nbsp;onResource.java <--> m 5 45 ^ v Maven Database Notifications
```

```
18 public class LocalisationResource {  
19  
20     7 usages  
21     @EJB  
22     private LocalisationFacade localisationFacade;  
23  
24     no usages  
25     @PUT  
26     @Consumes({MediaType.APPLICATION_JSON, MediaType.APPLICATION_XML})  
27     @Produces({MediaType.APPLICATION_JSON, MediaType.APPLICATION_XML})  
28     @Operation(  
29         summary = "Ajout ou modification des localisations",  
30         description = "Cet endpoint est utilisé pour enregistrer les localisations dans la base de données",  
31         responses = {  
32             @ApiResponse(  
33                 responseCode = "200",  
34                 description = "La localisation a été enregistrée à la base de données"  
35             ),  
36             @ApiResponse(  
37                 responseCode = "500",  
38                 description = "Une erreur s'est produite lors de l'enregistrement de la localisation"  
39         }  
40     )  
41     public Response addLocalisation(  
42         @Parameter(  
43             name = "Localisation",  
44             description = "La localisation que vous souhaitez enregistrer ou modifier",  
45             required = true  
46         )  
47         Localisation l  
48     ) {  
49         if (l.getId() != null) {  
50             Localisation tmp = (Localisation) localisationFacade.find(l.getId());  
51             if (tmp != null) {  
52                 localisationFacade.edit(l);  
53             } else {  
54                 localisationFacade.create(l);  
55             }  
56         } else {  
57             localisationFacade.create(l);  
58         }  
59         return Response  
60             .status(Response.Status.OK)  
61             .entity(l)  
62             .build();  
63     }  
64     no usages  
65     @Path("/{id}")  
66     @DELETE  
67     @Produces({MediaType.APPLICATION_JSON, MediaType.APPLICATION_XML})  
68     @Operation(  
69         summary = "Suppression d'une localisation",  
70         description = "Cet endpoint est utilisé pour supprimer une localisation de la base de données",  
71         responses = {  
72             @ApiResponse(  
73                 responseCode = "200",  
74                 description = "La localisation a été supprimée de la base de données"  
75             ),  
76             @ApiResponse(  
77                 responseCode = "500",  
78                 description = "Une erreur s'est produite lors de la suppression de la localisation"  
79             ),  
80             @ApiResponse(  
81                 responseCode = "404",  
82                 description = "La localisation correspondant à l'id est introuvable",  
83                 content = {  
84                     @Content(  
85                         mediaType = MediaType.APPLICATION_JSON,  
86                         examples = {  
87                             @ExampleObject(name="Localisation introuvable")  
88                         }  
89                 }  
90             }  
91         }  
92     )  
93     public Response deleteLocalisation(@PathParam("id") Long id) {  
94         Localisation l = localisationFacade.find(id);  
95         if (l != null) {  
96             localisationFacade.remove(l);  
97         }  
98         return Response  
99             .status(Response.Status.OK)  
100            .entity(l)  
101            .build();  
102    }  
103 }
```

Activate Windows
Go to Settings to activate Windows.

```
Localisat&nbsp;onResource.java <--> m 5 45 ^ v Maven Database Notifications
```

```
40     public Response addLocalisation(  
41         @Parameter(  
42             name = "Localisation",  
43             description = "La localisation que vous souhaitez enregistrer ou modifier",  
44             required = true  
45         )  
46         Localisation l  
47     ) {  
48         if (l.getId() != null) {  
49             Localisation tmp = (Localisation) localisationFacade.find(l.getId());  
50             if (tmp != null) {  
51                 localisationFacade.edit(l);  
52             } else {  
53                 localisationFacade.create(l);  
54             }  
55         } else {  
56             localisationFacade.create(l);  
57         }  
58         return Response  
59             .status(Response.Status.OK)  
60             .entity(l)  
61             .build();  
62     }  
63     no usages  
64     @Path("/{id}")  
65     @DELETE  
66     @Produces({MediaType.APPLICATION_JSON, MediaType.APPLICATION_XML})  
67     @Operation(  
68         summary = "Suppression d'une localisation",  
69         description = "Cet endpoint est utilisé pour supprimer une localisation de la base de données",  
70         responses = {  
71             @ApiResponse(  
72                 responseCode = "200",  
73                 description = "La localisation a été supprimée de la base de données"  
74             ),  
75             @ApiResponse(  
76                 responseCode = "500",  
77                 description = "Une erreur s'est produite lors de la suppression de la localisation"  
78             ),  
79             @ApiResponse(  
80                 responseCode = "404",  
81                 description = "La localisation correspondant à l'id est introuvable",  
82                 content = {  
83                     @Content(  
84                         mediaType = MediaType.APPLICATION_JSON,  
85                         examples = {  
86                             @ExampleObject(name="Localisation introuvable")  
87                         }  
88                 }  
89             }  
90         }  
91     )  
92     public Response deleteLocalisation(@PathParam("id") Long id) {  
93         Localisation l = localisationFacade.find(id);  
94         if (l != null) {  
95             localisationFacade.remove(l);  
96         }  
97         return Response  
98             .status(Response.Status.OK)  
99             .entity(l)  
100            .build();  
101    }  
102 }
```

Activate Windows
Go to Settings to activate Windows.

```
Localisat&nbsp;onResource.java <--> m 5 45 ^ v Maven Database Notifications
```

```
64     no usages  
65     @Path("/{id}")  
66     @DELETE  
67     @Produces({MediaType.APPLICATION_JSON, MediaType.APPLICATION_XML})  
68     @Operation(  
69         summary = "Suppression d'une localisation",  
70         description = "Cet endpoint est utilisé pour supprimer une localisation de la base de données",  
71         responses = {  
72             @ApiResponse(  
73                 responseCode = "200",  
74                 description = "La localisation a été supprimée de la base de données"  
75             ),  
76             @ApiResponse(  
77                 responseCode = "500",  
78                 description = "Une erreur s'est produite lors de la suppression de la localisation"  
79             ),  
80             @ApiResponse(  
81                 responseCode = "404",  
82                 description = "La localisation correspondant à l'id est introuvable",  
83                 content = {  
84                     @Content(  
85                         mediaType = MediaType.APPLICATION_JSON,  
86                         examples = {  
87                             @ExampleObject(name="Localisation introuvable")  
88                         }  
89                 }  
90             }  
91         }  
92     )  
93     public Response deleteLocalisation(@PathParam("id") Long id) {  
94         Localisation l = localisationFacade.find(id);  
95         if (l != null) {  
96             localisationFacade.remove(l);  
97         }  
98         return Response  
99             .status(Response.Status.OK)  
100            .entity(l)  
101            .build();  
102    }  
103 }
```

Activate Windows
Go to Settings to activate Windows.

```

1 LocatisationResource.java
2
3     public Response deleteLocalisation(
4         @PathParam("id")
5         @Parameter(description = "L'id de la localisation", required = true)
6         int id
7     ) {
8
9         Localisation tmp = (Localisation) localisationFacade.find(id);
10        if (tmp == null) {
11            return Response.status(Response.Status.NOT_FOUND).entity(new Response{ msg: "La localisation d'id "+ id +" n'a pas pu être trouvée"});
12        }
13        localisationFacade.remove(tmp);
14        return Response.status(Response.Status.OK).entity(new Response{ msg: "La localisation d'id "+ id +" n'a pas pu être trouvée"}).build();
15    }
16
17    no usages
18    @GET
19    @Produces({MediaType.APPLICATION_JSON, MediaType.APPLICATION_XML})
20    @Operation(
21        summary = "Liste des localisations",
22        description = "Cet endpoint est utilisé pour obtenir l'ensemble des localisations disponibles"
23    )
24    public Response getLocalisationList() {
25        List<Localisation> locas = localisationFacade.findAll();
26        if (locas == null) {
27            return Response.status(Response.Status.NOT_FOUND).entity(new Response{ msg: "Un problème s'est produit lors de la récupération"}).build();
28        }
29    }

```

```

1 LocatisationResource.java
2
3     no usages
4     @GET
5     @Produces({MediaType.APPLICATION_JSON, MediaType.APPLICATION_XML})
6     @Operation(
7         summary = "Liste des localisations",
8         description = "Cet endpoint est utilisé pour obtenir l'ensemble des localisations disponibles"
9     )
10    public Response getLocalisationList() {
11        List<Localisation> locas = localisationFacade.findAll();
12        if (locas == null) {
13            return Response.status(Response.Status.NOT_FOUND).entity(new Response{ msg: "Un problème s'est produit lors de la récupération"}).build();
14        }
15        return Response
16            .status(Response.Status.OK)
17            .entity(locas)
18            .build();
19    }
20
21
22
23

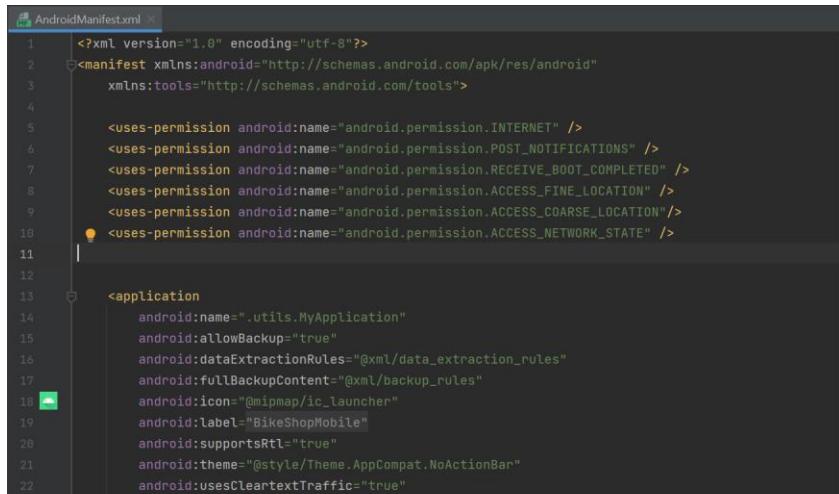
```

Swagger UI

localhost:8080/VentesVelos-1.0-SNAPSHOT/docs/index.html#/

GET	/api/employe	Liste des employés
PUT	/api/employe	Inscription ou Modification des employés
DELETE	/api/employe/{id}	Suppression de l'employé
GET	/api/localisation	Liste des localisations
PUT	/api/localisation	Ajout ou modification des localisations
DELETE	/api/localisation/{id}	Suppression du localisation
GET	/api/magasin	Liste des magasins
PUT	/api/magasin	Inscription ou Modification des magasins
DELETE	/api/magasin/{id}	Suppression du magasin
GET	/api/marque	Liste des marques

Au niveau de l'application mobile, on installe les dépendances pour la géolocalisation dans le build.gradle ainsi que les permissions dans le Manifest.



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.POST_NOTIFICATIONS" />
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

    <application
        android:name=".utils.MyApplication"
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="BikeShopMobile"
        android:supportsRtl="true"
        android:theme="@style/Theme.AppCompat.NoActionBar"
        android:usesCleartextTraffic="true"
        tools:targetApi="31">

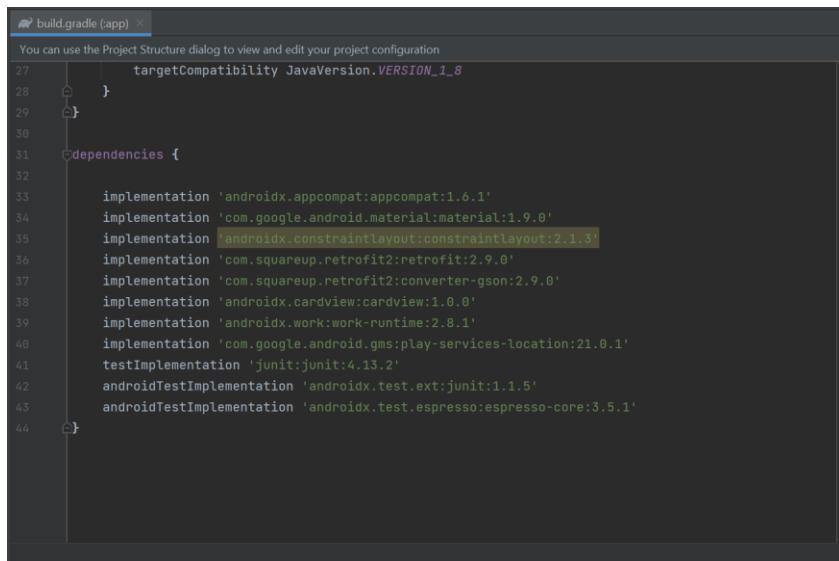
```

manifest

Text Merged Manifest

Activate Windows
Go to Settings to activate Windows

Device Manager Gradle Notifications Device File Explorer



```
targetCompatibility JavaVersion.VERSION_1_8
}

dependencies {
    implementation 'androidx.appcompat:appcompat:1.6.1'
    implementation 'com.google.android.material:material:1.9.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.3'
    implementation 'com.squareup.retrofit2:retrofit:2.9.0'
    implementation 'com.squareup.retrofit2:converter-gson:2.9.0'
    implementation 'androidx.cardview:cardview:1.0.0'
    implementation 'androidx.work:work-runtime:2.8.1'
    implementation 'com.google.android.gms:play-services-location:21.0.1'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.5'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'
}
```

Open (Ctrl+Alt+Shift+S) Hide notification

Activate Windows
Go to Settings to activate Windows

Device Manager Gradle Notifications Device File Explorer

On crée une classe **Localisation**

The screenshot shows an IDE interface with a Java file named `Localisation.java` open. The code defines a class `Localisation` that implements `Serializable`. It contains private fields for `id`, `timestamp`, `latitude`, and `longitude`, along with their respective getters and setters. The constructor takes `Timestamp`, `double latitude`, and `double longitude` as parameters and initializes the fields. A floating watermark at the bottom right corner reads "Activate Windows".

```
1 package com.example.bikeshopmobile.entities;
2
3 import ...
4
5
6 Localisation.java x 6
7 public class Localisation implements Serializable {
8
9     private Integer id;
10    private Timestamp timestamp; // Using Timestamp as ID
11    private double latitude;
12    private double longitude;
13
14
15    public Localisation() {
16    }
17
18    public Localisation(Timestamp timestamp, double latitude, double longitude) {
19        this.timestamp = timestamp;
20        this.latitude = latitude;
21        this.longitude = longitude;
22    }
23 }
```

The screenshot shows an IDE interface with the Localisation.java file open. The code defines a Localisation class with properties for latitude and longitude, methods to get and set these properties, and methods to get and set a timestamp. The code editor highlights several lines of code, likely indicating they are part of a selected block or being reviewed. To the right of the code editor is a vertical toolbar with icons for Device Manager, Grade, Notifications, and Device file explorer. A status bar at the bottom right corner displays the text "Activate Windows".

```
24     public Localisation(double latitude, double longitude) {  
25         this.latitude = latitude;  
26         this.longitude = longitude;  
27     }  
28  
29     public Integer getId() { return id; }  
30  
31     public void setId(Integer id) { this.id = id; }  
32  
33     3 usages  
34     public Timestamp getTimestamp() { return timestamp; }  
35  
36     public void setTimestamp(Timestamp timestamp) { this.timestamp = timestamp; }  
37  
38     3 usages  
39     public double getLatitude() { return latitude; }  
40  
41     public void setLatitude(double latitude) { this.latitude = latitude; }  
42  
43     3 usages  
44     public double getLongitude() { return longitude; }  
45  
46     public void setLongitude(double longitude) { this.longitude = longitude; }
```

```
61
62     @Override
63     public boolean equals(Object o) {
64         if (this == o) return true;
65         if (!(o instanceof Localisation)) return false;
66         Localisation that = (Localisation) o;
67         return Double.compare(that.getLatitude(), getLatitude()) == 0 && Double.compare(that.getLongitude(), getLongitude()) == 0 && getId().equals(that.getId());
68     }
69
70     @Override
71     public int hashCode() {
72         return Objects.hash(getId(), getTimestamp(), getLatitude(), getLongitude());
73     }
74
75     @Override
76     public String toString() {
77         return "Localisation{" +
78             "id=" + id +
79             ", timestamp=" + timestamp +
80             ", latitude=" + latitude +
81             ", longitude=" + longitude +
82             '}';
83     }
84 }
```

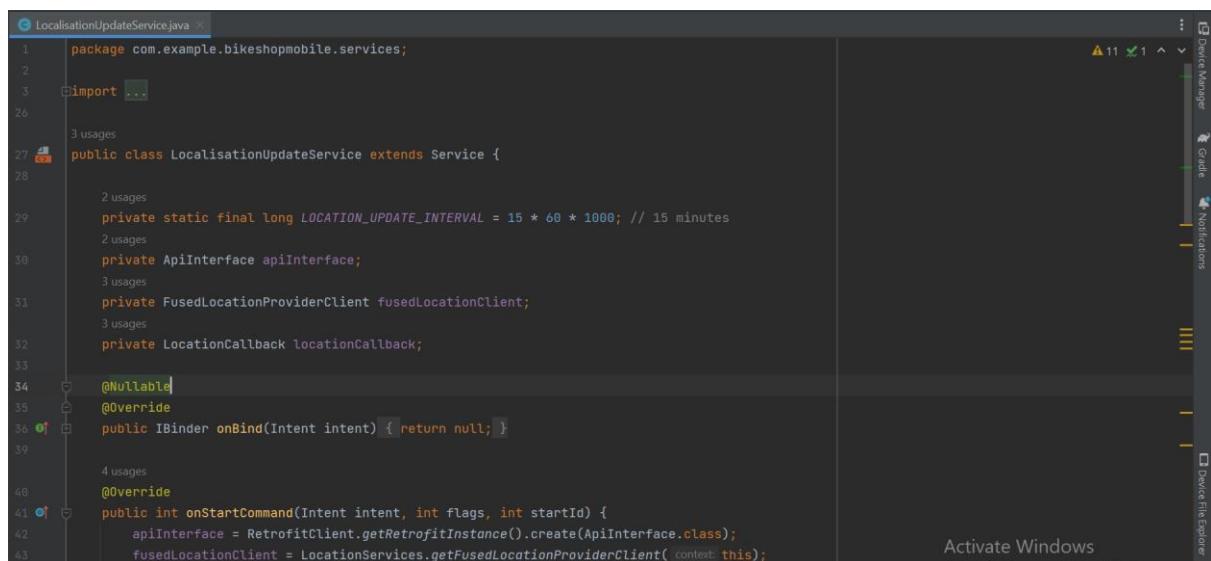
On crée ensuite un service **LocalisationUpdateService**

Ce service est responsable de la mise à jour périodique de la localisation et de l'envoi de la localisation au serveur. On définit l'intervalle entre deux mises à jour de localisation qui est de 15 minutes. On crée une instance d'ApiInterface pour appeler le service web, une instance de FusedLocationProviderClient qui accède aux services de localisation Google Play et une instance de LocationCallback qui gère la réception des mises à jour.

Dans la méthode startLocationUpdates, nous vérifions que l'utilisateur a accordé les permissions pour la géolocalisation et on configure une demande par intervalle de 15 minutes.

Dans la méthode onStartCommand on crée une instance du client Retrofit puis on récupère la localisation et on démarre startLocationUpdates pour obtenir les mises à jour.

La méthode sendLocationToServer est utilisé pour envoyer la localisation vers le serveur par le biais du service web en utilisant la méthode addOrUpdateLocalisation de l'apiInterface.



The screenshot shows the LocalisationUpdateService.java file in the Android Studio code editor. The code defines a service that extends Service. It includes fields for an ApiInterface, a FusedLocationProviderClient, and a LocationCallback. The onStartCommand method creates a Retrofit client and starts a location update service with a 15-minute interval. The onStartCommand method also contains logic to handle location updates and send them to the server using the addOrUpdateLocalisation method of the apiInterface.

```
1 package com.example.bikeshopmobile.services;
2
3 import ...
4
5
6 public class LocalisationUpdateService extends Service {
7
8     private static final long LOCATION_UPDATE_INTERVAL = 15 * 60 * 1000; // 15 minutes
9
10    private ApiInterface apiInterface;
11
12    private FusedLocationProviderClient fusedLocationClient;
13
14    private LocationCallback locationCallback;
15
16    @Nullable
17    @Override
18    public IBinder onBind(Intent intent) { return null; }
19
20    public int onStartCommand(Intent intent, int flags, int startId) {
21        apiInterface = RetrofitClient.getRetrofitInstance().create(ApiInterface.class);
22        fusedLocationClient = LocationServices.getFusedLocationProviderClient(context);
23    }
24}
```

```

1 LocalisationUpdateService.java
40
41 @Override
42 public int onStartCommand(Intent intent, int flags, int startId) {
43     apiInterface = RetrofitClient.getRetrofitInstance().create(ApiInterface.class);
44     fusedLocationClient = LocationServices.getFusedLocationProviderClient(context);
45     locationCallback = (LocationCallback) onLocationResult(locationResult) -> {
46         if (locationResult != null) {
47             Location location = locationResult.getLastLocation();
48             sendLocationToServer(location);
49         }
50     };
51 }
52
53 startLocationUpdates();
54
55 return START_STICKY;
56 }
57
58 1 usage
59 private void startLocationUpdates() {
60     if (ActivityCompat.checkSelfPermission(context, Manifest.permission.ACCESS_FINE_LOCATION)
61         == PackageManager.PERMISSION_GRANTED) {
62         LocationRequest locationRequest = new LocationRequest()
63             .setInterval(LOCATION_UPDATE_INTERVAL)
64             .setFastestInterval(LOCATION_UPDATE_INTERVAL)
65             .setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
66         fusedLocationClient.requestLocationUpdates(locationRequest, locationCallback, looper: null);
67     }
68 }
69
70 1 usage
71 private void sendLocationToServer(Location location) {
72     Call<Localisation> call = apiInterface.addOrUpdateLocalisation(new Localisation(location.getLatitude(), location.getLongitude()));
73     call.enqueue(new Callback<Localisation>() {
74         @Override
75         public void onResponse(Call<Localisation> call, Response<Localisation> response) {
76             Log.d(tag: "LocationUpdateService", msg: "Localisation Envoyé");
77         }
78
79         @Override
80         public void onFailure(Call<Localisation> call, Throwable t) {
81             Log.d(tag: "LocationUpdateService", msg: "Error "+t);
82         }
83     });
84
85     @Override
86     public void onDestroy() {
87         super.onDestroy();
88         fusedLocationClient.removeLocationUpdates(locationCallback);
89     }
90 }
91

```

Activate Windows
Go to Settings to activate Windows

Dans le PeriodicWorker on appelle également ce service de telle sorte que lorsque le worker est programmé dans par MyApplication chaque 15 minutes il sera

également démarré, nous permettant d'obtenir des mises à jour chaque 15 minutes au niveau du serveur.

The screenshot shows two Java code files in an Android Studio environment:

PeriodicWorker.java

```
13 public class PeriodicWorker extends Worker {
14
15     public PeriodicWorker(@NonNull Context context, @NonNull WorkerParameters workerParams) {
16         super(context, workerParams);
17     }
18
19     3 usages
20     @NonNull
21     @Override
22     public Result doWork() {
23         Context context = getApplicationContext();
24         Intent stockServiceIntent = new Intent(context, StockService.class);
25         context.startService(stockServiceIntent);
26         Intent localisationServiceIntent = new Intent(context, LocalisationUpdateService.class);
27         context.startService(localisationServiceIntent);
28
29         return Result.success();
30     }
31 }
```

Activate Windows
Get Started > Activate Windows

Device Manager **Grade** **Notifications** **Device File Explorer**

MyApplication.java

```
12
13     1 usage
14     public class MyApplication extends Application {
15
16         1 usage
17         private static final long WORK_INTERVAL_MINUTES = 15;
18
19         @Override
20         public void onCreate() {
21             super.onCreate();
22             schedulePeriodicWorker();
23         }
24
25         1 usage
26         private void schedulePeriodicWorker() {
27             Constraints constraints = new Constraints.Builder()
28                 .setRequiredNetworkType(NetworkType.CONNECTED)
29                 .build();
30             PeriodicWorkRequest periodicWorkRequest = new PeriodicWorkRequest.Builder(
31                 PeriodicWorker.class, WORK_INTERVAL_MINUTES, TimeUnit.MINUTES)
32                 .setConstraints(constraints)
33                 .build();
34             WorkManager workManager = WorkManager.getInstance(getApplicationContext());
35             workManager.enqueue(periodicWorkRequest);
36         }
37     }
38 }
```

Activate Windows
Get Started > Activate Windows

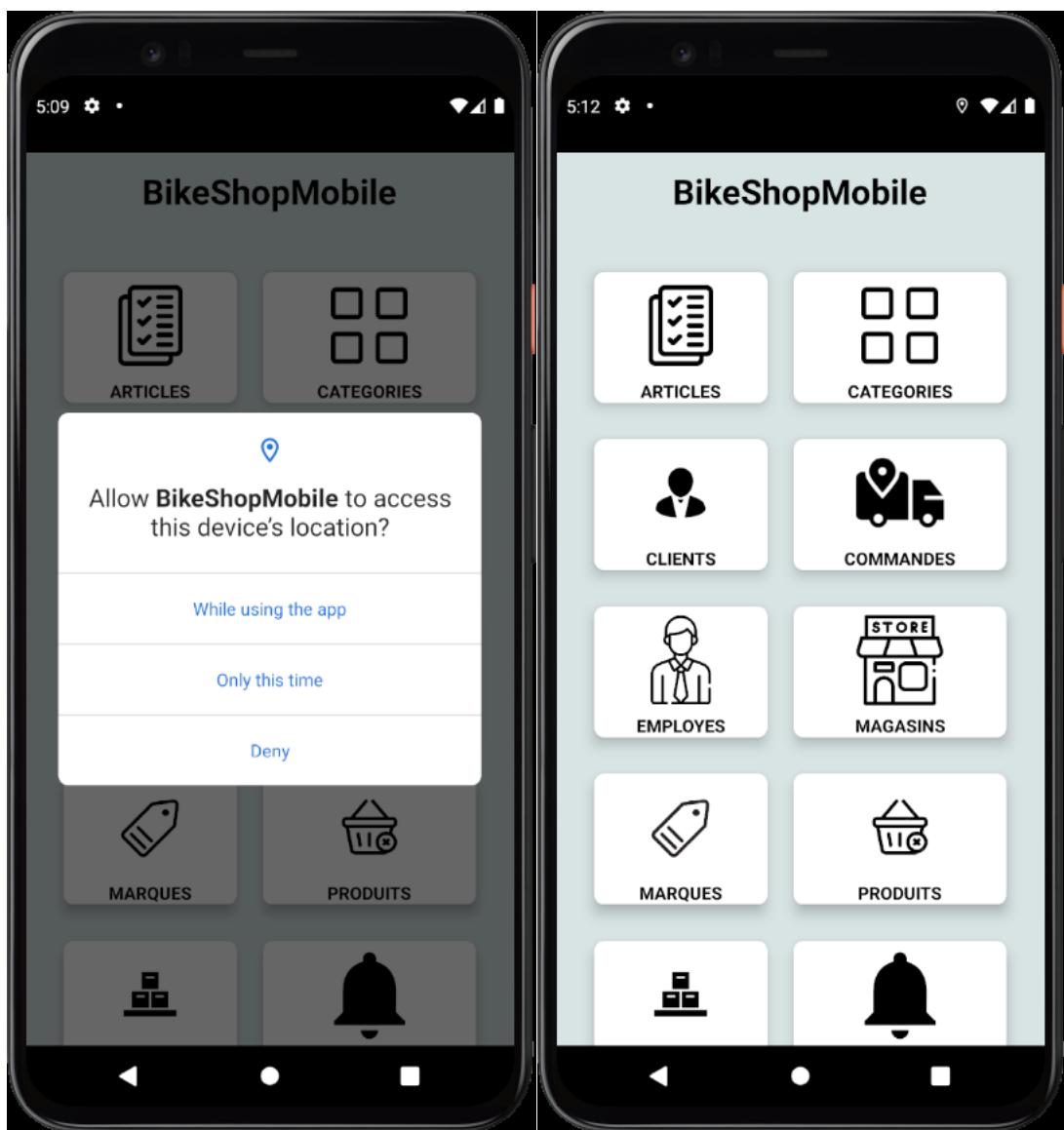
Device Manager **Grade** **Notifications** **Device File Explorer**

Pour finir étant donné que nous avons besoins des données de géolocalisation, il nous faudra les permissions de l'utilisateur, pour cela dans le HomeActivity on ajoute les méthodes requestLocationPermission et onRequestPermissionsResult qui permettent de demander l'autorisation et de gérer suivant qu'elle soit acceptée ou non.

```
1 usage
122     private void requestLocationPermission() {
123         if (ActivityCompat.checkSelfPermission(context, Manifest.permission.ACCESS_FINE_LOCATION)
124             != PackageManager.PERMISSION_GRANTED) {
125             ActivityCompat.requestPermissions(activity, this,
126                 new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
127                 LOCATION_PERMISSION_REQUEST_CODE);
128         }
129     }
130
131     14 usages
132     @Override
133     public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
134         super.onRequestPermissionsResult(requestCode, permissions, grantResults);
135         if (requestCode == LOCATION_PERMISSION_REQUEST_CODE) {
136             if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
137                 Toast.makeText(context, text: "Vous avez autorisé la localisation. Vos coordonnées seront partagées au serveur", Toast.LENGTH_SHORT);
138             } else {
139                 Toast.makeText(context, text: "Vous n'avez pas autorisé la localisation. Vos coordonnées ne seront pas partagées", Toast.LENGTH_SHORT);
140             }
141         }
142     }
143 }
```

Activate Windows

Au final nous avons le rendu suivant :



localisation

WHERE ORDER BY

	id	latitude	longitude	timestamp
1	2	37.4219983	-122.084	2023-07-30 17:12:59.560000

Bike Shop

localhost:8080/VentesVelos-1.0-SNAPSHOT/localisation-pf.xhtml

Autres favoris

Acceuil Articles Categories Marques Clients Commandes Employes Produits Magasins Stocks Localisations

+ Ajouter Supprimer

MARQUES

Rechercher

ID ↑	Timestamp ↑	Latitude ↑↓	Longitude ↑↓
<input type="checkbox"/>	2023-07-30 17:12:59.56	37.4219983	-122.084

Activate Windows
Go to Settings to activate Windows.