Pragmatec
INGENIERIE INFORMATIQUE

# Exercice
# Application Android
# Avec  Android Studio

# L’application PIZZA

Pragmatec
INGENIERIE INFORMATIQUE

# Table des matières

Pragmatec
INGENIERIE INFORMATIQUE

# Section 1 Présentation de l'application

Après la prise en main de Android Studio et des technologies Android nous allons créer une solution nommée « l'application Pizza »

Il s'agit de présenter dans une page de l'application une liste de Pizzas sous forme d'un titre, d'une description, une image, un prix.

Les techniques que l'on va mettre en place sont :

- Chercher les données sur un serveur Web
- Lecture d'un fichier JSON
- Affichage dans une liste

Cette application permet de conforter les acquis.

# Section 2 Création du projet

## 2.1 But

- Créer le projet sous Android Studio,
- créer une page

## 2.2 Créer le projet

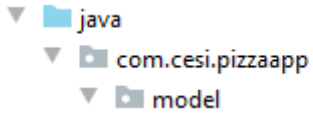
Sous Android Studio, créer un projet vide « PizzaApp » dans le package com.cesi.pizzaapp

Vérifier que la page par défaut s'affiche dans le viewer local puis dans le simulateur ou téléphone.

## 2.3 Créer la couche données

Nous allons créer un package réservé aux données de cette application

Créer par Android Studio un sous package à com.cesi.pizzaapp nommé model




Y créer une classe de nom Pizza

Créer des Propriétés (attributs avec des accesseurs):

- nom
- prix (float)
- une tableau d'ingrédients, ingredients
- une image, identifiée par son lien web (String)

Créer un constructeur permettant « d'hydrater » tous ses attributs.

Dans le code MainActivity.java, déclarer une liste de pizzas nommée pizzas.

Créer une méthode `private void createPizzas()`

qui crée quelques pizzas et les ajoute à la liste.

Dans onCreate() appeler createPizzas().

Tester ce code. Graphiquement rien de nouveau, mais l 'application ne doit pas planter !

# Section 3 Gestion des icônes

## 3.1 But

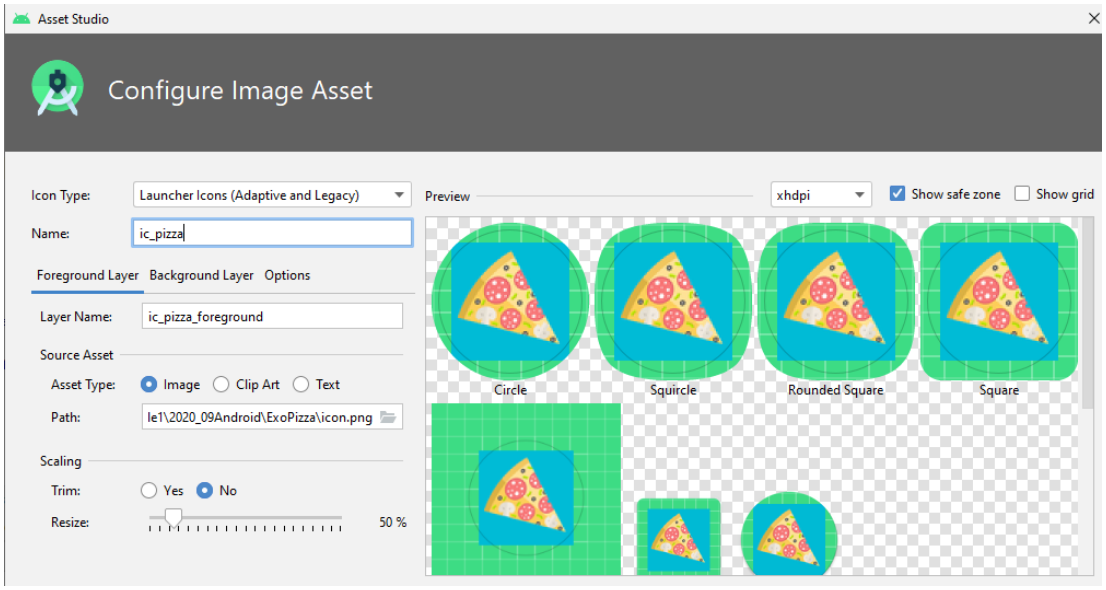
- Définir les icônes de l'application

## 3.2 Exercice

Le fichier icon.png est fourni.

Sur le nœud app, faire New/Image Asset

Renseigner le fenêtre comme suit :




Modifier Androidmanifest.xml pour utiliser ces icones.

Tester la modification

# Section 4 Afficher la liste des Pizzas

## 4.1 But

- Utilisation du design pattern Adapter
- lecture de la liste des pizzas et affichage individuel

## 4.2 Exercice

Lien utilisé : https://www.javatpoint.com/android-recyclerview-list-example

Dans activity_main.xml

- Tout supprimer
- Mettre un LinearLayout vertical
- Y inclure un RecyclerView avec pour identifiant « recycler »

Le reclyclerView est un conteneur de plusieurs sous view de même type.

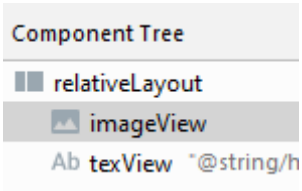
Créer le fichier de ressource dimens.xml. Pour cela :

- sur le nœud app : New/XML/ Values XML file
- le nommer dimens.xml
- Le contenu est le suivant :

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <dimen name="activity_horizontal_margin">16dp</dimen>
    <dimen name="activity_vertical_margin">16dp</dimen>
    <dimen name="ic_clear_margin">56dp</dimen>
</resources>
```

Créer un layout, sans code java associé, qui va décrire le contenu d'une ligne de la liste. Pour cela :

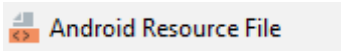
- sur le nœud app, New / XML / Layout XML File
- Utiliser la partie Design pour obtenir ceci




- On doit obtenir un code comme celui-ci (mais essayez avec le mode Design) :

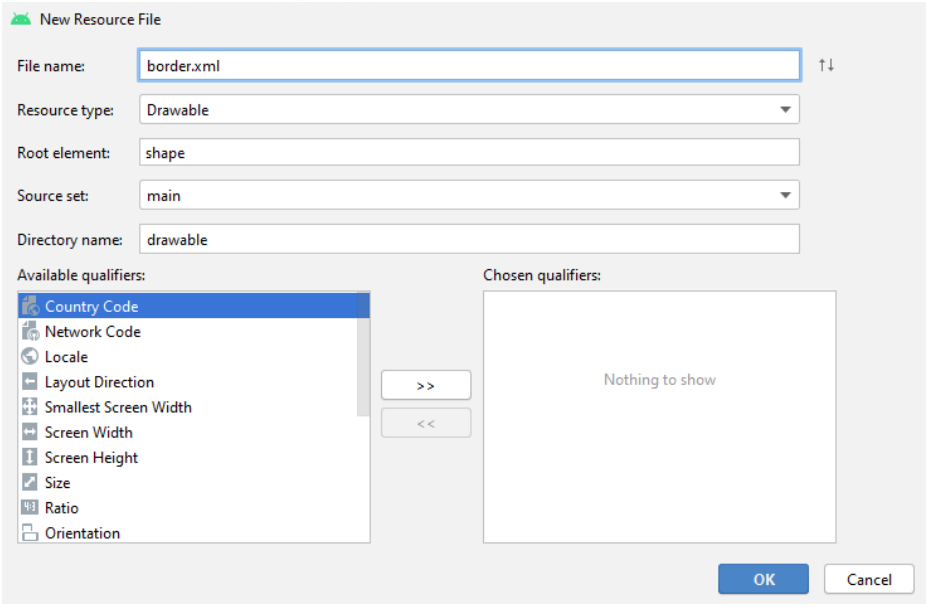
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/relativeLayout"
    android:layout_width="match_parent"
    android:layout_height="?android:attr/listPreferredItemHeightLarge"
    android:background="@drawable/border">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true"
        android:layout_alignParentLeft="true"
        android:layout_centerVertical="true"
        android:layout_marginStart="@dimen/activity_horizontal_margin"
        android:layout_marginEnd="@dimen/activity_horizontal_margin"
        android:contentDescription="Icon"
         />

    <TextView
        android:id="@+id/texView"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_toEndOf="@+id/imageView"
        android:layout_toRightOf="@id/imageView"
        android:gravity="center_vertical"
        android:text="@string/hello_blank_fragment"
        android:textSize="14sp" />
</RelativeLayout>
```

Créer le fichier border.xml pour les bords des items :




sur app menu New /

saisir les champs comme suit :




Le contenu du fichier est :

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
    android:shape="rectangle">
    <solid android:color="#FFFFFF" />
    <stroke
        android:width="1dp"
        android:color="#CCCCCC" />
</shape>
```

La partie list_item.xml contient un imageView qui va afficher une image dont on fournit le lien URL.

Il n'y a pas de propriété directe pour fournir ce lien. Il faut fournir une bitmap.

La bibliothèque Picasso va faire cette transformation.

https://square.github.io/picasso/

Pour la mettre en place :

Aller dans Gradle Scripts/build.gradle(Module : app) et ajouter en fin :

```
implementation 'com.squareup.picasso:picasso:2.71828'
```

Faire l'action Buid/Rebuild Project

Partie la plus compliquée : le mécanisme d'allocation et de remplissage des lignes.

MyListAdapter est une classe héritée de RecyclerView.Adapter

La méthode onCreateViewHolder() alloue et remplit les widgets de list_item.xml.

OnBindViewHolder() remplit individuellement chaque widget d'une seule ligne.

Créer dans le package com.pizza.pizzaapp la classe MyListAdapter

Copier le code suivant :

```
public class MyListAdapter extends RecyclerView.Adapter<MyListAdapter.ViewHolder>{
    private Pizza[] listdata;

    // RecyclerView recyclerView;
    public MyListAdapter(Pizza[] listdata) {
        this.listdata = listdata;
    }
    @Override
    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        LayoutInflater layoutInflater = LayoutInflater.from(parent.getContext());
        View listItem= layoutInflater.inflate(R.layout.list_item, parent, false);
        ViewHolder viewHolder = new ViewHolder(listItem);
        return viewHolder;
    }

    @Override
    public void onBindViewHolder(ViewHolder holder, int position) {
        final Pizza myListData = listdata[position];
        holder.textView .setText(listdata[position].getNom());
        Picasso.get().load(listdata[position].getImgURI()).resize
(150,150).centerCrop().into(holder.imageView);

        holder.relativeLayout.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Toast.makeText(view.getContext(),"click on item:
"+myListData.getNom(),Toast.LENGTH_LONG).show();
            }
        });
    }


    @Override
    public int getItemCount() {
        return listdata.length;
    }

    public static class ViewHolder extends RecyclerView.ViewHolder {
        public ImageView imageView;
        public TextView textView;
        public RelativeLayout relativeLayout;
        public ViewHolder(View itemView) {
            super(itemView);
            this.imageView = (ImageView) itemView.findViewById(R.id.imageView);
            this.textView = (TextView) itemView.findViewById(R.id.texView );
            this.relativeLayout =
(RelativeLayout)itemView.findViewById(R.id.relativeLayout);
        }
    }
}
```

Pragmatec
INGENIERIE INFORMATIQUE

Tester

Modifier le code pour afficher la liste des ingrédients dans la fenêtre Popup qui s'ouvre sur clic d'une ligne.

# Section 5 Gestion du splash screen

## 5.1 But

- Définir la vue de lancement

## 5.2 Exercice

Le terme splash screen désigne la vue visible au lancement de l'application, pour faire patienter l'utilisateur.

Généralement une photo fixe et texte fixe de présentation.

Le site suivant fournit un exemple

https://abhiandroid.com/programming/splashscreen#:~:text=%20%20%201%20Step%201%20%3A%20Create,-%3E%20MainActivity.%20java%20and%20add%20the...%20More%20

Nous ne l'implémenterons pas dans cet exercice.

Pragmatec
INGENIERIE INFORMATIQUE

# Section 6 Le format JSON

## 6.1 But

- Ecrire des données JSON
- Dé-sérialisation de données JSON

## 6.2 Exercice

Le format JSON (Javascript Object Notation) est un support pour formater les données texte.

La dé-sérialisation est faite dans notre appli en utilisant le package Json.net

Etape 1 : simuler des données JSON

Aller sur le site https://jsoneditoronline.org et essayer de créer des données JSON qui vont décrire deux pizzas

Copier ce code et le placer dans une String pizzaJSON dans createPizzas de MainActivity.java

Etape 2 : convertir le JSON en objets de classe Pizza

Il n'existe pas de deserializer JSON to object en Android.

Il existe des librairies, ex https://benjii.me/2010/04/deserializing-json-in-android-using-gson/

Nous allons créer nous même un convertisseur JSON en Pizza

- Dans le package model, créer la classe JsonToObject
- créer une méthode statique qui accepte en paramètre une chaîne JSON et qui fournit en sortie une liste de Pizzas

```
public static ArrayList<Pizza> convert(String json)
```

Vous aurez besoin des méthodes :

- JSONArray()
- getJSONObject() ou optJSONObject()
- obj.getString(), obj.getJSONArray()
- Integer.parseInt()

Utiliser cette méthode dans createPizzas () de MainActivity.java

Mettre un point d'arrêt après cette ligne et regarder le contenu de la liste pizzas avec le debugger.

Simuler le résultat.

Tester l'application.

Pragmatec
INGENIERIE INFORMATIQUE

# Section 7 JSON fourni par un serveur

## 7.1 But

- Utiliser les données de notre application en utilisant un échange JSON depuis un serveur.
- Utiliser un serveur

## 7.2 Exercice

Notre application va interroger un serveur pour obtenir ses données.

Notre appli va formuler sa demande dans un message http de type GET.

Le serveur va formuler sa réponse, non pas en html, css, javascript, mais dans un format binaire ou String. Le serveur ne peut pas envoyer des Objets Pizza, au sens POO. => il va les fournir en texte au format JSON


Pour simuler cet échange nous allons utiliser un lien vers Google Drive.

Ce lien est

https://drive.google.com/uc?export=download&id=1YVa8DNIuESXJI8sM4sxcd-w-rk6xpFIZ

A son appel, Google Drive fournit les données au format JSON.

Sur un vrai téléphone Android, le transfert de données depuis internet ne va pas se faire. Pour l'autoriser, il faut ouvrir le fichier AndroidManifest.xml et ajouter la ligne suivante au dessus de <application>

```
<uses-permission android:name="android.permission.INTERNET" />
```

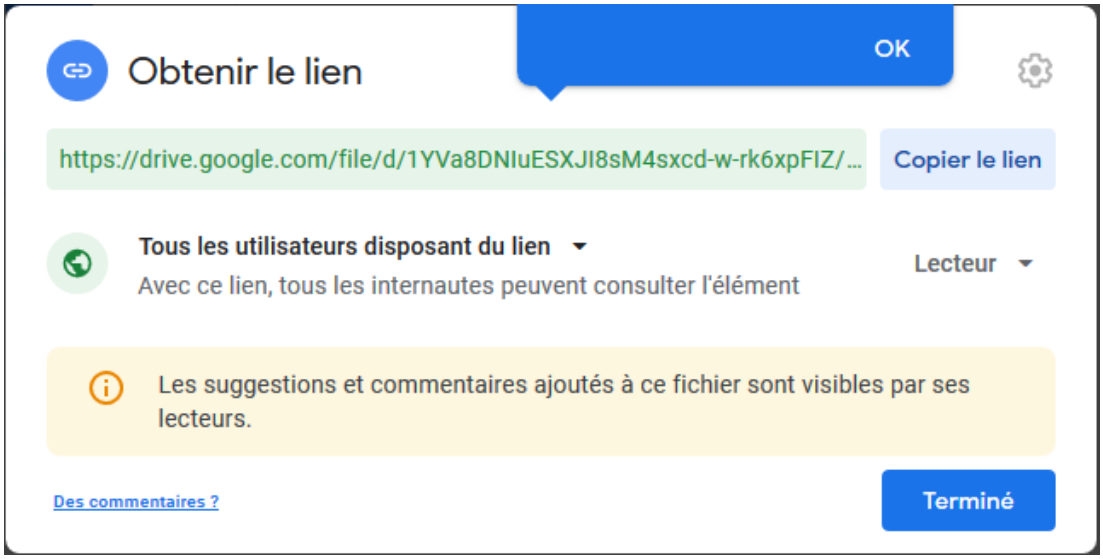
Pour mettre votre propre fichier JSON sur un drive Google :

Se connecter sur le drive Google, rubrique Mon Drive

Créer un sous répertoire

Y placer le fichier JSON

Désigner le fichier JSON et choisir Partager

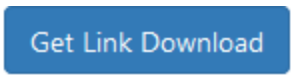



Appuyer sur Copier le lien

Vous pouvez essayer le lien dans un navigateur Web. Il ne fournit pas directement le contenu mais une page Web qui montre le contenu.

Pour obtenir un lien direct sur les données, aller sur le site https://www.gdirect.link/ et fournir le lien

Le bouton




fournit cette fois le lien à utiliser.

Retour sur notre application :

En Java, pour se connecter et recevoir des données depuis un serveur, le code principal est :

- former une Url avec la classe URL
- établir une connexion avec le serveur , url.openConnection()
- récupérer le flux de données , urlConnection.getInputStream()
- exploiter les données du stream avec un Bufferreader

Cependant tout ceci va prendre un certain temps, tributaire des échanges sur le Web – plusieurs secondes.

Si ces traitement sont faits sur le thread principal de l'application, celui qui gère la file d'attente des événements et l'affichage, l'application peut se geler quelques instants, pire Android peut décider de stopper l'application.

Le traitement de la récupération de données doit se faire dans un thread séparé ...

# Section 8 Traitements asynchrones

## 8.1 But

- Traitement asynchrone de la récupération des données

## 8.2 Exercice

La problématique est que l'on bloque l'application pendant le temps de chargement des données JSON.

Il faut rendre cette méthode asynchrone.

Le code va se trouver dans le fichier GetJSONData.java placé dans le package model

En voici le contenu.

```
public class GetJSONData {
    private String url;
    private IProcessJSONData processAAppeler;

    public GetJSONData(String url, IProcessJSONData processAAppeler) {
        this.url = url;
        this.processAAppeler = processAAppeler;
    }


    public void startDownLoad()
    {
        DownLoadJSONData downLoadJSONData = new DownLoadJSONData();
        downLoadJSONData.execute(this.url); // Appelle
DownLoadJSONData.doInBackground()
    }

    class DownLoadJSONData extends AsyncTask<String, Void, String>
    {
        // exécuté en tâche de fond, dans un autre thread
        @Override
        protected String doInBackground(String... strings) {
            BufferedReader reader = null;
            InputStream inputStream = null;
            HttpURLConnection urlConnection = null;
            StringBuffer buffer = null;
            try {
                URL url = new URL(strings[0]);
                urlConnection = (HttpURLConnection) url.openConnection ();
```

```
                inputStream = urlConnection.getInputStream();
                if (inputStream == null)
                    return  null;
                reader = new BufferedReader(new
InputStreamReader(inputStream));
                String line;
                buffer = new StringBuffer();
                while ((line=reader.readLine()) != null)
                {
                    buffer.append(line + "\n");
                }
            } catch (MalformedURLException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            } finally {
                if (urlConnection != null)
                    urlConnection.disconnect();
                if (reader != null) {
                    try {
                        reader.close();
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }
            }
            return buffer.toString();
        }

        @Override
        protected void onPostExecute(String result) {
            super.onPostExecute(result);
            Log.i("JSON", result);
            processAAppeler.processJSONData(result);
        }
    }
    public interface IProcessJSONData
    {
        public void processJSONData(String json);
    }
}
```

Lire et tenter de comprendre le contenu.

Pour que l'application soit autorisée par Android à accéder à des sites distants, il faut modifier AndroidManifest.xml et ajouter la ligne suivante au dessus de <application>

```
<uses-permission android:name="android.permission.INTERNET" />
```

Il faut modifier MainActivity.java pour utiliser cette classe :

- Il faut que la classe MainActivity implémente l'interface IprocessJSONData
- Ctrl + Entrée propose d'implémenter la méthode processJSONData(). Y mettre le code suivant :

```
@Override
public void processJSONData(String json) {
    pizzas = JsonToObject.convert(json);
    RecyclerView recyclerView = (RecyclerView)
findViewById(R.id.recycler);
    MyListAdapter adapter = new MyListAdapter(pizzas);
    recyclerView.setHasFixedSize(true);
    recyclerView.setLayoutManager(new LinearLayoutManager(this));
    recyclerView.setAdapter(adapter);

}
```

- dans createPizzas() commenter le code existant et y ajouter

```
GetJSONData jsonData = new GetJSONData("https://drive.google.com/uc?
export=download&id=1YVa8DNIuESXJI8sM4sxcd-w-rk6xpFIZ", this);
jsonData.startDownLoad();
```

- Dans le onCreate() commenter le code qui est maintenant déplacé dans `processJSONData()` ci-dessus

Simuler ce code et constater qu'une page blanche s'affiche avant les données.

Cette page blanche traduit que le traitement de recherche des données est maintenant asynchrone.

# Section 9 Suite possible

## 9.1 Suite

Cette application est loin d'être finie !

La suite logique à donner :

- développer une page d'attente pendant que les données sont récupérées du serveur
- implémenter un serveur qui crée dynamiquement les données JSON
- implémenter le Pull to refresh : sur scroll haut ou bas, demander au serveur un nouveau lot de données
- Trier les données par nom, par prix, croissant ou décroissant
- Mémoriser les préférences de l'utilisateur
- gérer les favoris