PROGRAMMATION MOBILE ANDROID

TP 2

API REST

charles.meunier@u-bourgogne.fr

OBJECTIFS

- Effectuer des requêtes auprès d'une API Rest
- Manipuler des listes déroulantes
- Manipuler un token de sécurité.

EXERCICE 1.1 - PROJET

Vous allez créer une application de prise de commandes pour la pizzéria My Pizza !

L'utilisateur pourra créer son compte, se connecter, commander une pizza et voir la liste de ses précédentes commandes.

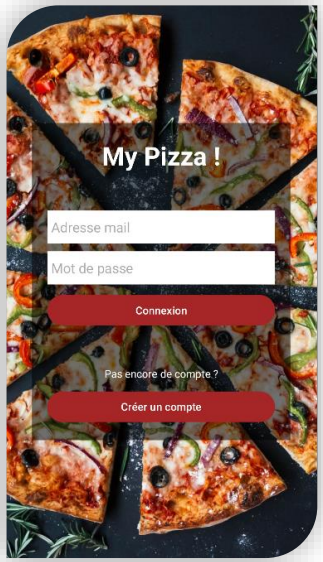
Le serveur de l'application est déjà prêt et propose une API Rest qui vous fournira les fonctionnalités serveur nécessaires.

Reprendre un projet

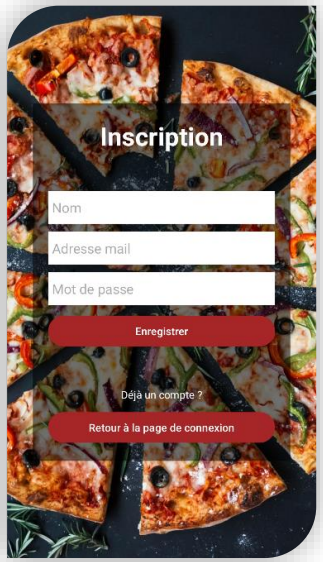
- Téléchargez le projet `Android 2024-TP2-Kotlin-Base` en cliquant ici.
- Ouvrez ce projet dans Android Studio.

Interface

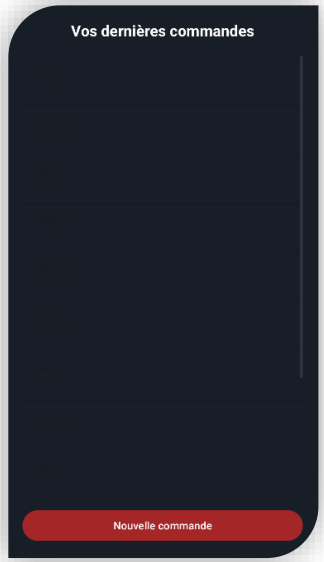
Les interfaces du projet sont déjà prêtes, vous pourrez ainsi vous concentrer sur le code :



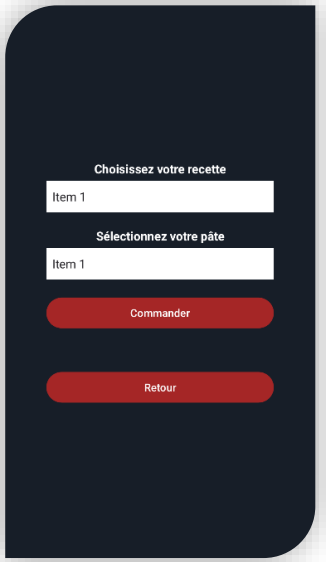
LoginActivity



RegisterActivity



OrdersActivity



OrderActivity

Test

Démarrez l'application pour vous assurer que tout fonctionne correctement.

EXERCICE 1.2 - PRÉPARATION

Accès à Internet

- Ajoutez au manifeste de l'application la permission `INTERNET`.

Dépendance

L'utilisation de l'API Rest nécessitera une dépendance vers la bibliothèque Gson qui se chargera de parser ou de transformer des données en JSON.

- Ouvrez le fichier `Gradle Scripts/build.gradle.kts(Module :app)`
- Ajoutez la ligne suivante dans la section `dependencies` :

```
implementation("com.google.code.gson:gson:2.10.1")
```

API

Une classe encapsulant le processus de requêtes HTTP est disponible ici.

- Téléchargez la classe Api et placez là dans le dossier de votre application, au même niveau que les activités.

EXERCICE 1.3 - CRÉATION DE COMPTE

RegisterData

```
data class RegisterData(
    val name: String,
    val mail: String,
    val password: String
)
```

- Ajoutez à votre projet une nouvelle `data class` nommée `RegisterData` qui représentera les données transmise au serveur lors de la création d'un compte :

Register

- Dans la classe `RegisterActivity`, ajoutez une méthode `register` qui sera connectée au bouton `Enregistrer` de l'interface.
- Dans cette méthode, récupérez les informations saisies par l'utilisateur et placez les dans une structure `RegisterData`.

RegisterSuccess

- Dans la classe `RegisterActivity`, ajoutez une méthode `registerSuccess` qui prendra en paramètre un entier nommé `responseCode`.
- Si `responseCode` vaut 200, terminé l'activité pour revenir à la page de connexion.

EXERCICE 1.4 - REQUÊTE VERS L'API

Api

La classe `Api` fournit un moyen simple d'effectuer des requêtes auprès d'un serveur. Elle met à votre disposition trois méthodes :

```
Api().get<DataToReceiveType>("URL DE l'API", ::onSuccessMethod, securityToken)

Api().post<DataToSendType>("URL DE l'API", dataToSend, ::onSuccessMethod, securityToken)

Api().post<DataToSendType, DateToReceiveType>("URL DE l'API", dataToSend, ::onSuccessMethod, securityToken)
```

*Note : **`securityToken`** est un paramètre facultatif.*

Exemple d'utilisation :

```
Api().get<List<String>>("https://monurl.com/books", ::loadingSuccess)
```

- Dans la méthode `register`, appelez `Api().post` pour transmettre les données de l'utilisateur à l'API `register` du serveur.

Note : la documentation de l'API se trouve à la fin du sujet.

- Testez et vérifiez que votre compte à bien été créé en ouvrant le lien suivant dans votre navigateur :
  https://mypizza.lesmoulinsdudev.com/users

EXERCICE 1.5 - CONNEXION

LoginData

➢ Créez une data class `LoginData` contenant un champ `mail` et un champ `password`.

```
data class LoginData(
    val mail: String,
    val password: String
)
```

LoginSuccess

➢ Ajoutez à la classe `LoginActivity` une méthode `loginSuccess` qui prendra en paramètre
- un entier nommé `responseCode`
- une chaine de caractères (`String?`) nommée `token`

Notez le `?` derrière `String` pour le type de `token`. Il signifie que le paramètre `token` vaudra peut être `null` et non une `String` comme on pourrait s'y attendre. Ceci notamment dans le cas où la requête ne serait pas valide.

➢ Démarrez l'activité `OrdersActivity` si `responseCode` est égal à 200. Pensez à transmettre dans l'intent le token reçu.

Login

➢ Ajoutez à la classe `LoginActivity` une méthode login sera connectée au bouton Connexion de l'interface.

➢ Transmettez les informations de connexion au serveur via l'API `auth`. Cette dernière vous renverra un `token` de connexion sous la forme d'une chaine de caractères.

Test

➢ Testez le bon fonctionnement.

EXERCICE 2.1 - RECETTES DE PIZZAS

RecipeData

- Créez une data class `RecipeData` contenant un attribut `id` (Int) et un attribut `name` (String).
- Redéfinissez la fonction `toString` de `RecipeData` de manière à ce qu'elle retourne la valeur de l'attribut `name`.

Recipes

- Ajoutez à la classe `OrderActivity` un attribut `recipes` de type `ArrayList<RecipeData>` et initialisé avec `ArrayList()`.

LoadRecipesSuccess

- Ajoutez à la classe `OrderActivity` une méthode `loadRecipesSuccess` qui prendra en paramètre un entier nommé `responseCode` et une `List<RecipeData>?` nommée `loadedRecipes`.
- Si `responseCode` est égal à 200 et `loadedRecipes` est différent de `null`, actualisez l'attribut `recipes` avec les valeurs contenues dans `loadedRecipes`.

LoadRecipes

- Ajoutez à la classe `OrderActivity` une méthode `loadRecipes` qui effectuera une requête auprès de l'API `recipe` du serveur.
- Appelez la méthode `loadRecipes` dans la méthode `onCreate` de `OrderActivity`

EXERCICE 2.2 - LISTE DÉROULANTES

Adaptateur

- Ajoutez à la classe `OrderActivity` un attribut `recipesAdapter` de type `ArrayAdapter<RecipeData>` qui sera initialisé plus tard.

- Dans la méthode `onCreate`, initialisez `recipesAdapter` comme indiqué ci-dessous :

```
recipesAdapter = ArrayAdapter<RecipeData>(this, androidx.appcompat.R.layout.support_simple_spinner_dropdown_item, recipes);
```

InitializeSpinners

- Ajoutez à la classe `OrderActivity` une méthode `initializeSpinners` qui affectera `recipesAdapter` comme adapter de la liste déroulante `spinRecipe`.

- Appelez `initializeSpinners` dans `onCreate`.

UpdateRecipesList

- Ajoutez à la classe `OrderActivity` une méthode `updateRecipesList` qui appellera la méthode `notifyDataSetChanged` de `recipesAdapter`.

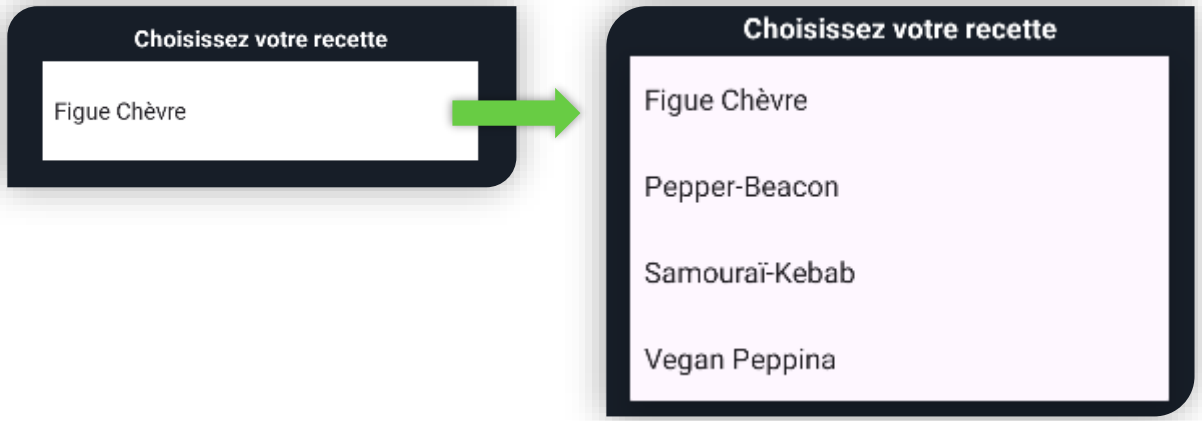
Attention : les appels à l'API se produisent dans un autre thread (IO) que celui qui gère l'interface utilisateur. Or seul ce dernier peut effectuer une actualisation de l'interface. Le bloc suivant permet de s'assurer que le bon thread va s'occuper de l'actualisation de la liste :

```
runOnUiThread {
    //Todo
}
```

EXERCICE 2.3 - TESTS ET ÉPAISSEURS DE PÂTE

Test

- Démarrez l'application et rendez-vous sur l'écran permettant de réaliser une nouvelle commande.
- Vous devriez voir quelque chose comme ceci :



Dough

- Reprenez les points 2.1 et 2.2 en remplaçant `Recipe` par `Dough` pour charger les épaisseurs de pâte.
- Testez le bon fonctionnement.

EXERCICE 2.4 - COMMANDE

OrderData

- Créez une data class `OrderData` contenant un attribut `recipeId` (Int) et un attribut `doughId` (Int).

OrderSuccess

- Ajoutez à la classe `OrderActivity` une méthode `orderSuccess` qui prendra en paramètre un entier nommé `responseCode`.
- Terminez l'activité si `responseCode` vaut 200.

sendOrder

- Ajoutez à la classe `OrderActivity` une méthode `sendOrder` qui récupérera les id de la recette et de l'épaisseurs de pâte sélectionnées et les placera dans une structure `OderData`.

```
(spinRecipe.selectedItem as RecipeData).id,
```

- Transmettez ces données au serveur via l'API `order`. N'oubliez pas de transmettre le `token` de sécurité.
- Connectez `sendOrder` au bouton `Commander` de l'interface.
- Testez le bon fonctionnement.

Note : des outils comme PostMan ou ReqBin vous permettent d'effectuer des requêtes vers des API et de voir le résultat.

BONUS- LISTE DES COMMANDES

Order

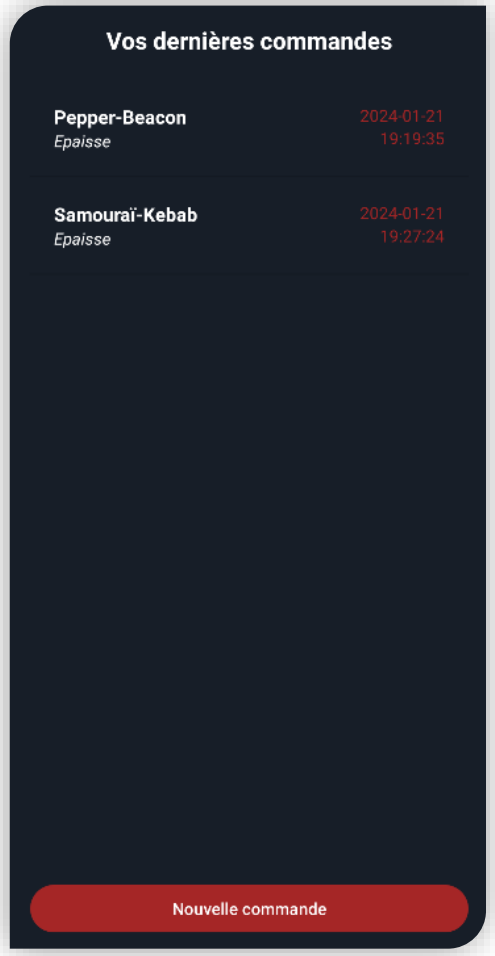
- Créez une data class `Order` contenant un attribut `recipe` (`String`), un attribut `dough` (`String`) et un attribut `orderData` (`String`).

Listes les commandes

- A partir des éléments vus précédemment et lors du TD 2, affichez la liste des commandes passées par l'utilisateur en utilisant l'API `orders`.

- Ajoutez des Toast informatifs pour l'utilisateur ("Compte créé", "Commande envoyée", ...). Attention a bien utiliser `runUIThread`.

- Vous êtes prêts pour le projet ;-)



DOCUMENTATION DE L'API

register

URL de l'API :	**https://mypizza.lesmoulinsdudev.com/register**
Méthode de connexion :	**POST**
Paramètres attendus :	▪ name : string ▪ mail : string (Attention : format mail vérifié !) ▪ password : string
Code de réponse :	▪ Code 200 si enregistrement OK ▪ Code 500 si erreur lors de l'enregistrement ▪ Code 400 si la requête est mauvaise
Donnée en retour	Aucune

DOCUMENTATION DE L'API

auth

URL de l'API :	**https://mypizza.lesmoulinsdudev.com/auth**
Méthode de connexion :	**POST**
Paramètres attendus :	▪ mail : string (Attention : format mail vérifié !) ▪ password : string
Code de réponse :	▪ Code 200 si enregistrement OK ▪ Code 401 si les identifiants sont incorrects
Donnée en retour	token de sécurité : string

DOCUMENTATION DE L'API

recipes

URL de l'API :	**https://mypizza.lesmoulinsdudev.com/recipes**
Méthode de connexion :	**GET**
Paramètres attendus :	Aucun
Code de réponse :	▪ Code 200 si requête correcte ▪ Code 400 si requête incorrecte
Donnée en retour	Tableau au format JSON : [{"id":1, "name":"Recette 1", "ingredients":"..."}, {"id":2, "name":"Recette 2", "ingredients":"..."}, ...]

DOCUMENTATION DE L'API

doughs

URL de l'API :	**https://mypizza.lesmoulinsdudev.com/doughs**
Méthode de connexion :	**GET**
Paramètres attendus :	Aucun
Code de réponse :	▪ Code 200 si requête correcte ▪ Code 400 si requête incorrecte
Donnée en retour	Tableau au format JSON : [{"id":1, "name":"Epaisseur 1"}, {"id":2, "name":"Epaisseur 2"}, ...]

DOCUMENTATION DE L'API

order

URL de l'API :	**https://mypizza.lesmoulinsdudev.com/order**
Méthode de connexion :	**POST**
Header	Authorization: Bearer ***token***
Paramètres attendus :	▪ recipeId : integer ▪ doughId: integer
Code de réponse :	▪ Code 200 si commande enregistrée ▪ Code 400 si requête incorrecte ▪ Code 401 si accès non autorisé (token invalide)
Donnée en retour	Aucune

DOCUMENTATION DE L'API

orders

URL de l'API :	**https://mypizza.lesmoulinsdudev.com/orders**
Méthode de connexion :	**GET**
Header	Authorization: Bearer ***token***
Paramètres attendus :	Aucun
Code de réponse :	▪ Code 200 si commande enregistrée ▪ Code 401 si accès non autorisé (token invalide)
Donnée en retour	[{"recipe":"Recette 1", "dough":"Epaisseur 2","orderDate":"2024-01-01 13:28:54"}, {"recipe":"Recette 1", "dough":"Epaisseur 1","orderDate":"2024-01-04 11:51:23"}, ...]