Documentation technique – MZNotesAPI

0. Introduction

Cette application consiste en notre première expérience de développement d'application Android native. Nous ne pouvons pas expliquer les détails spécifiques de fonctionnement du Framework Android. Ni les fonctionnalités proposées par les API du système.

Notre première version de l'application permettait d'enregistrer des recettes de cuisines venant de l'API Food2Fork. Cependant, cette API ne fournit plus de services ce qui nous a contraint de revoir notre application aux derniers moments.

Actuellement, nous disposons d'une application de gestions de tâches qui permet d'appeler OpenWeatherMap API pour visualiser le temps actuel d'une ville renseignée.

1. Logiciels

Afin de développer l'application mobile, l'IDE **Android Studio** a été utilisé.

Android Studio est un fork de Google d'IntelliJ Community par **Jetbrains**. Il est l'**IDE officiel** pour développer des applications natives Android.
Il était ainsi un choix évident.

2. Langage utilisé

Le choix du langage était libre, le but étant de fournir une application mobile, nous avions choisi de développer notre application en Java.

3. Version d'Android

Nous avons choisi la version SDK Android 32, correspondant à Android 11.

4. Bibliothèques utilisées

Plusieurs bibliothèques ont été utilisées dans cette application :
ExAssert, **Material**, **LocalizationActivity**, et beaucoup d'autres bibliothèques **AndroidX**.

ExAssert est une bibliothèque Android non officielle, développée par , implémentant des assertions avec un système d'exceptions catchées en interne en cas d'assertion fausse.
Sur Android, les assertions built-in sont toujours désactivées.
Un support dédié à Android n'est pas implémenté, du moins pour le moment : il faudrait prendre en compte la constante *BuildConfig.DEBUG* pour désactiver les assertions.
Code source : https://github.com/AntoineJT/ExAssert

Material est une bibliothèque Android officielle, permettant d'utiliser les *SnackBar* dans l'application.

LocalizationActivity est une bibliothèque Android non officielle, développée par akexorcist en Kotlin, permettant de changer la langue de l'application facilement au runtime.
Code source : https://github.com/akexorcist/Localization

AndroidX est un ensemble de bibliothèques officielles Android consistant en des versions réécrites des API Android qui ne sont pas fournies avec le système.

Parmi les bibliothèques **AndroidX** utilisées se trouvent : **ConstraintLayout**, **AppCompat**, **NavigationFragment**, **NavigationUI**, **RecyclerView**, **Volley**.

ConstraintLayout est un type de layout utilisé dans l'*activité ChangeLangActivity* où les éléments sont placés relativement entre eux avec des contraintes.

AppCompat est une bibliothèque permettant aux anciennes versions d'Android d'utiliser des versions de l'API plus récentes.

NavigationFragment et **NavigationUI** sont des bibliothèques liées à la navigation, elles ont été ajoutées par Android Studio directement dans le projet généré. **NavigationFragment** est utilisée pour inclure le *fragment TaskList* dans l'activité principale *MainActivity*.

RecyclerView est une bibliothèque permettant de dresser une liste plus efficace, pouvant être rafraichie efficacement.

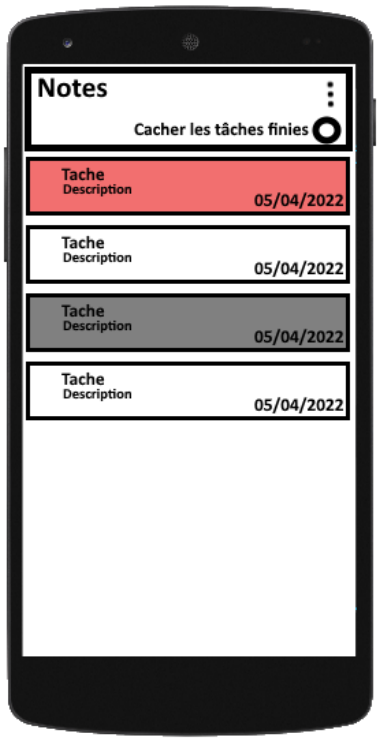
Volley est une bibliothèque http qui rend la mise en réseau simple et rapide. Elle est utilisée pour permettre l'appel de l'API OpenWeatherMap API.

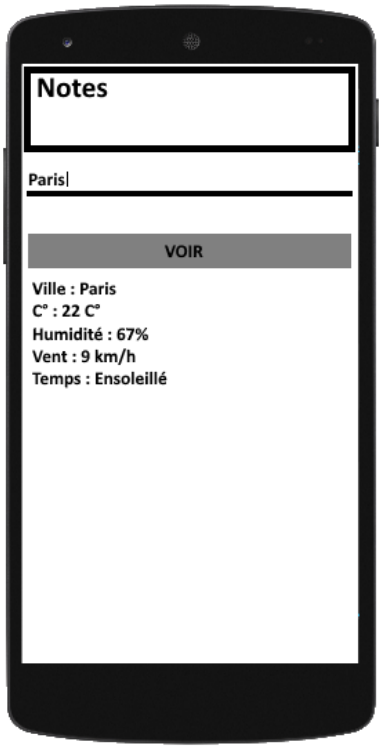
5. Langues

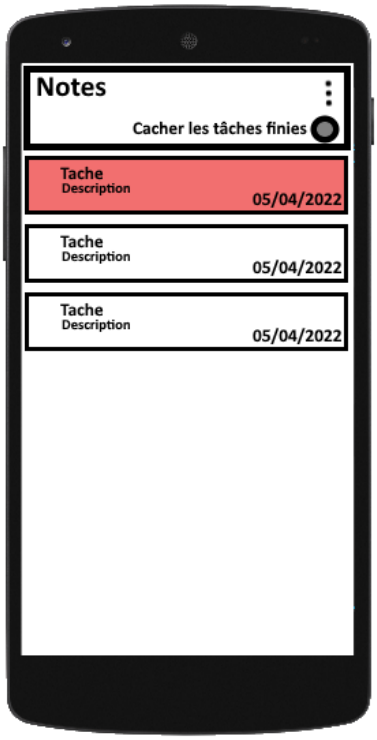
Nous avons utilisé le système d'internationalisation (i18n) inclut dans Android afin de proposer 2 langues distinctes : l'anglais (par défaut) et le français.

Travail préparatoire

1. Maquettes

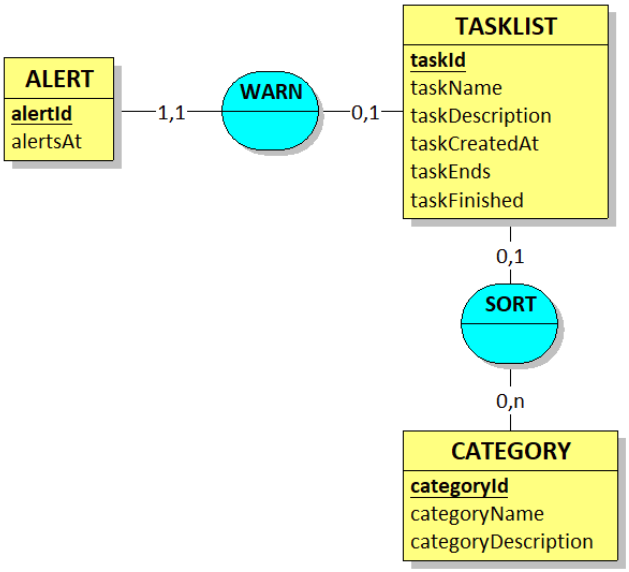






2. Modèle Conceptuel de Données

De ce modèle, nous n'utilisons dans l'application que la table **TaskList**, les autres servant à des fonctionnalités que nous n'avons pas eu le temps d'implémenter, tel que l'alerte par notification peu avant expiration de la tâche, ainsi que le système de classement des tâches par catégories.



Explication de certains snippets de code

WeatherActivity.java

```
@Override
public void onResponse(String response) {
    String output = "";
    try {
        JSONObject jsonResponse = new JSONObject(response);
        JSONArray jsonArray = jsonResponse.getJSONArray( name: "weather");
        JSONObject jsonObjectWeather = jsonArray.getJSONObject( index: 0);
        String description = jsonObjectWeather.getString( name: "description");
        JSONObject jsonObjectMain = jsonResponse.getJSONObject("main");
        double temp = jsonObjectMain.getDouble( name: "temp") - 273.15;
        double feelsLike = jsonObjectMain.getDouble( name: "feels_like") - 273.15;
        float pressure = jsonObjectMain.getInt( name: "pressure");
        int humidity = jsonObjectMain.getInt( name: "humidity");
        JSONObject jsonObjectWind = jsonResponse.getJSONObject("wind");
        String wind = jsonObjectWind.getString( name: "speed");
        JSONObject jsonObjectClouds = jsonResponse.getJSONObject("clouds");
        String clouds = jsonObjectClouds.getString( name: "all");
        JSONObject jsonObjectSys = jsonResponse.getJSONObject("sys");
        String countryName = jsonObjectSys.getString( name: "country");
        String cityName = jsonResponse.getString( name: "name");
        tvResult.setTextColor(Color.rgb( red: 68,  green: 134,  blue: 199));
        output += "Current weather of " + cityName + " (" + countryName + ")"
                + "\n Temp: " + df.format(temp) + " °C"
                + "\n Feels Like: " + df.format(feelsLike) + " °C"
                + "\n Humidity: " + humidity + "%"
                + "\n Description: " + description
                + "\n Wind Speed: " + wind + "m/s (meters per second)"
                + "\n Cloudiness: " + clouds + "%"
                + "\n Pressure: " + pressure + " hPa";
        tvResult.setText(output);
    } catch (JSONException e) {
        e.printStackTrace();
    }
}
```

Ici, cette méthode permet de récupérer les données JSON retournées par l'appel de l'API lorsqu'une ville valide a été saisie.

Les données sont récupérées puis transmis vers le layout activity_weather.xml.

On exécute un try & catch pour relever une possible exception dans le cas où l'API ne revoit aucune valeurs

L'affichage est simple et basique.

CreateTaskActivity.java createTask()

```java
private TaskCreationStatus createTask() {
    try (DatabaseHandler db = DatabaseHandler.get(this.getApplicationContext())) {
        final List<Integer> fieldsId = Arrays.asList(
                R.id.fieldTaskName,
                R.id.fieldTaskEnd,
                R.id.fieldTaskDescription);
        // ordre --> nom, deadline, description
        final List<String> values = fieldsId.stream()
                .map(this::getEditTextValue)
                .collect(Collectors.toList());

        if (values.stream().anyMatch(String::isEmpty)) {
            return TaskCreationStatus.EMPTY_FIELDS;
        }

        final String name = values.get(0);
        final String deadline = values.get(1);
        final String description = values.get(2);

        if (!isDeadlineCorrect(deadline)) {
            return TaskCreationStatus.DATE_FORMAT;
        }

        final boolean succeed = db.createTask(name, description, DateFormatter.parse(deadline)) != -1;

        return succeed ? TaskCreationStatus.OK : TaskCreationStatus.ERROR;
    } catch (Exception e) {
        return TaskCreationStatus.ERROR;
    }
}
```

Cette méthode permet de créer une tâche, tout en indiquant à l'utilisateur si l'opération réussi ou non, et en lui donnant des informations dans le cas d'un échec.

Tout d'abord, on récupère les 3 champs fieldTaskName, fieldTaskEnd, fieldTaskDescription.
Ensuite, on récupère la valeur contenue dans chaque champ.
On vérifie que tous les champs sont remplis. Dans le cas contraire, une erreur indiquant que les champs ne sont pas tous remplis est retournée à l'utilisateur.

On vérifie ensuite que l'échéance est formatée correctement.
Finalement, on tente de créer la tâche. En cas d'échec une erreur assez vague est retournée à l'utilisateur, et en cas de réussite, il est notifié du succès également.

L'indication de réussite ou d'échec est faite via l'affichage d'une SnackBar.