

weathR Android - Consignes expert (4 ans et +)

Vous vous apprêtez à effectuer un test technique créé par Smart&Soft.

Avant de commencer

Assurez-vous que votre environnement est à jour avec les versions suivantes :

- SDK Android avec les versions 23 des derniers outils (SDK API 23, build tools 23.0.2, bibliothèques de support 23.1.1, etc.) ;
- Android Studio v1.5.1 ;
- Gradle 2.8 ;

Pour aborder le test sereinement, il convient également d'avoir :

- un compte Github ;
- un outil vous permettant une interaction avec un dépôt git et notamment les actions suivantes : clonage d'un dépôt, création d'une branche, commit, push et création d'un tag ;
- un téléphone et une tablette (7 pouces ou plus) sous Android 4.0.3 ou plus ou à défaut un émulateur.

Consignes

En cliquant sur le bouton ci-dessous, vous découvrirez le lien d'un dépôt git hébergé sur Github et vous disposerez d'une heure pour effectuer le test.

Ce dépôt contient deux choses :

- le code source java d'une application Android développée sous Android Studio et qui vous servira de base pour effectuer ce test technique ;
- un fichier PDF "consignes.pdf" qui contient les consignes de chacun des exercices qui vous seront demandés dans ce test technique.

Cet entretien technique se composant de plusieurs questions / exercices, il convient que de réaliser un commit par exercice avant de mettre en avant l'avancement du projet et mettre en lumière le code mis en place pour chacun des exercices proposée.

Les exercices s'enchaînent de manière logique. Il convient donc, dans la mesure du possible, de les réaliser dans l'ordre dans lequel ils apparaissent dans le document "consignes.pdf".

Présentation du projet

Présentation fonctionnelle

weathR est une application simple, permettant de connaître la météo fictive de trois grandes villes, à savoir Paris, Londres et Milan, sur les cinq à dix prochains jours.

Cette application se compose de 3 écrans :

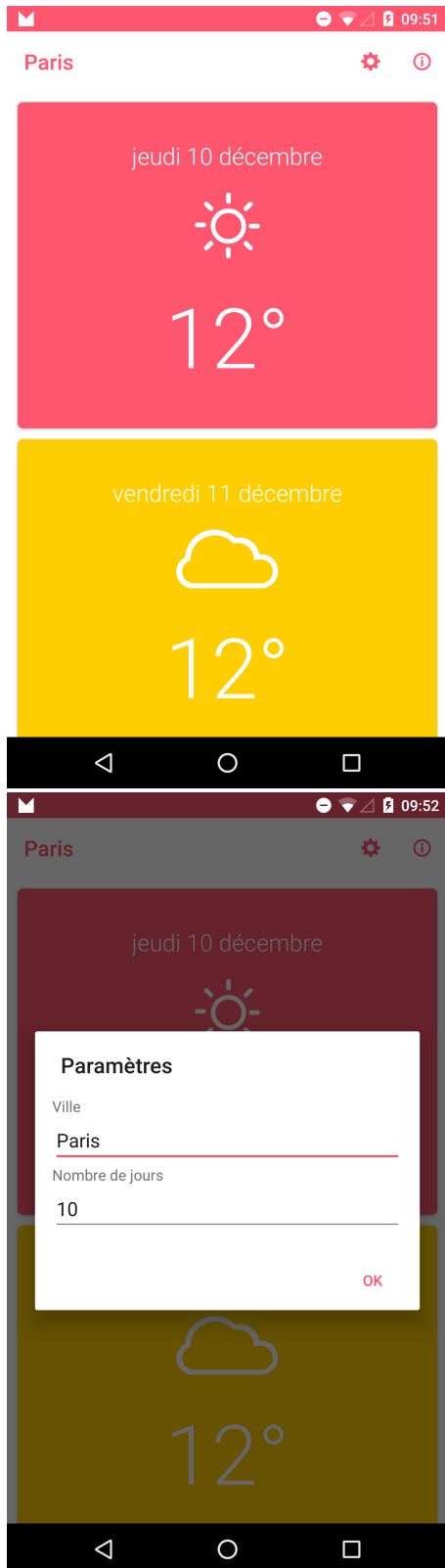
- L'écran "home" qui affiche sous la forme d'une liste les prévisions météo de la ville active.
- L'écran "détail" qui affiche de manière plus détaillée les prévisions météo d'une journée spécifique. On y accède en cliquant sur un item de la liste exposée par l'écran "home".
- L'écran "about", accessible depuis la barre d'action de la "home", qui permet d'afficher quelques informations relatives à l'éditeur de l'application, à savoir Smart&Soft.

Cette application se compose également d'une popup permettant de paramétrer la ville pour laquelle on souhaite connaître la météo, ainsi que le nombre de jour à afficher (entre cinq et dix). Cette popup est accessible depuis la "home", en cliquant sur la roue dentée depuis la barre d'action.

A noter que l'écran "home" supporte une action de "swipe-to-refresh" afin de forcer le rafraichissement de son contenu.

Les données de l'application sont récupérées depuis un web-service.


Pour mieux vous rendre du contenu de l'application, voici quelques captures d'écran :





← weathR

vendredi 11 décembre



12°

Température min.	11°
Température max.	12°
Index UV	12
Vent	E / 11



←  WeathR

Conçu, développé et réalisé par :



L'agence software qui stimule
votre expérience mobile

www.smartnsoft.com



Présentation technique

Le web-service

Comme précisé précédemment, les données permettant d'alimenter l'application sont issues d'un web-service : <http://smartnsoft.com/shared/weather/index.php?city={city}&forecasts={forecasts}>

Voici un exemple d'appel et du JSON retourné :

```
{
  "code": 200,
  "message": "OK",
  "city": "paris",
  "forecasts": [
    {
      "date": "2015-12-10",
      "temperatureMin": 10,
      "temperatureMax": 12,
      "type": "SUNNY",
      "uvIndex": 12,
      "windDirection": "W",
      "windSpeed": 10
    },
    {
      "date": "2015-12-11",
      "temperatureMin": 11,
      "temperatureMax": 12,
      "type": "CLOUDY",
      "uvIndex": 12,
      "windDirection": "E",
      "windSpeed": 11
    },
    {
      "date": "2015-12-12",
      "temperatureMin": 12,
      "temperatureMax": 12,
      "type": "RAINY",
      "uvIndex": 12,
      "windDirection": "NW",
      "windSpeed": 12
    },
    {
      "date": "2015-12-13",
      "temperatureMin": 12,
      "temperatureMax": 13,
      "type": "SNOWY",
      "uvIndex": 12,
      "windDirection": "SW",
      "windSpeed": 13
    },
    {
      "date": "2015-12-14",
      "temperatureMin": 12,
```

```
"temperatureMax": 14,  
"type": "STORMY",  
"uvIndex": 12,  
"windDirection": "NE",  
"windSpeed": 14
```

```
}
]
}
```

Architecture de l'application Android

Le projet weathR est un projet Android Studio ne contenant qu'un seul module, nommé "**app**" et correspondant à l'application Android.

Le *package name* de l'application est **com.smartnsoft.weathr** et présente dès sa racine quatre classes :

- **Constants.java** regroupe quelques constantes utilisées dans le projet ;
- **HomeActivity.java** représente l'Activité de l'écran "home" ;
- **DetailActivity.java** représente l'Activité de l'écran "détail" ;
- **AboutActivity.java** représente l'Activité de l'écran "about".

Détaillons alors le contenu des "sous-packages".

Le sous-package "**bo**" contient le seul et unique objet métier (*business object*) de l'application :

- **Weather.java** est un **POJO** Java très simple permettant de mapper le retour du web-service en objet Java. Les annotations sur l'objet sont liées à la bibliothèque Jackson. Cette bibliothèque, à l'image d'un Gson, permet de faciliter le *parsing* et l'exploitation de données au format JSON.

Le sous-package "**fragment**" contient les Fragments utilisés au sein de l'application. Dans le cadre de cette application, les Activités ne sont que des *containers*. Toute la logique métier (chargement et affichage des données) est gérée par les Fragments :

- **HomeFragment.java** représente le Fragment de l'écran "home" ;
- **DetailFragment.java** représente le Fragment de l'écran "détail" ;
- **AboutFragment.java** représente le Fragment de l'écran "about" ;
- **ParameterDialogFragment.java** représente le Fragment de la boîte de dialogue permettant de paramétrer la ville pour laquelle on souhaite afficher la météo ainsi que le nombre de jours à afficher.

Le sous-package "**ws**" contient les classes relatives aux appels à l'unique web-service :

- **WeathRServices.java** contient une seule méthode publique :
 - **getWeather** permet d'appeler le web-service décrit plus haut dans ce document. Cette méthode accepte deux paramètres : une ville et un nombre de jours.

Finalement, une *inner-class* est permet de mutualiser l'affichage d'une prévision entre la liste et l'écran de détail :

- **WeatherSimpleViewHolder** représente un **ViewHolder**

Exercices

Exercice 1

Après avoir récupéré le projet et l'avoir ouvert dans Android Studio, compilez l'application et exécutez la sur un terminal ou un émulateur.

Vous devriez constater que l'application ne fait rien de spécial, si ce n'est afficher un *loader*.

Il convient de mettre en place l'appel réseau à travers l'appel à la méthode :

```
WeathRServices.getInstance().getWeather(preferences.getString(HomeFragment.
CITY_PREFERENCES_KEY, Constants.DEFAULT_CITY),
preferences.getInt(HomeFragment.FORECASTS_DAYS_PREFERENCES_KEY,
Constants.DEFAULT_FORECASTS_DAYS));
```

A noter que c'est également à vous d'écrire le corps de la méthode qui dans l'état actuelle renvoie *null*.

Vous devez respecter la signature actuelle de la méthode pour qu'elle renvoie directement un objet **Weather** dont la structure et les informations sont mappées sur le json décrit plus haut dans ce document.

L'URL à appeler pour obtenir la météo est la suivante : <http://smartnsoft.com/shared/weather/index.php?city={city}&forecasts={forecasts}> où :

- **city** a pour valeur l'une des trois villes suivantes : *paris*, *milan* ou *london* ;
- **forecast** a pour valeur un entier compris entre 1 et 5 ;

Pour témoigner du bon fonctionnement de votre appel réseau, vous pouvez, par exemple, afficher la ville contenu dans l'objet **Weather** au sein d'un **Toast** ou d'une **Snackbar**.

Exercice 2

Maintenant que le web-service est consommé, il convient de restituer graphiquement ses informations à l'aide d'une liste.

Pour designer les éléments de votre liste, vous pouvez vous aider du document "**Guidelines_Android_WeathR.jpg**" présent dans le dépôt git.

A noter que la classe **WeatherSimpleViewHolder** représente un **ViewHolder** et peut-être utilisée ou non.

Exercice 3

Maintenant que la liste est en place, il convient de pouvoir ouvrir l'écran de détail.

Il convient de lui faire passer deux *extras* à l'ouverture via les clefs/valeurs sont disponibles ci-dessous :

- **HomeFragment.BUSINESS_OBJECT_EXTRA** : l'objet **Forecast** à ouvrir ;
- **HomeFragment.BACKGROUND_COLOR_EXTRA** : la couleur d'arrière plan.

Exercice 4

Maintenant que l'écran de détail s'ouvre, vous devriez constater que les informations suivantes ne s'affichent pas à l'écran, contrairement à ce qui est proposé dans les captures d'écran fournies plus tôt :

- la température minimale
- la température maximale
- l'index UV
- les informations relatives au vent

Exercices bonus

Les exercices suivants sont du bonus permettant de démontrer ce dont vous êtes capables. Contrairement aux exercices précédents, ils ne sont pas bloquants et peuvent être traités dans n'importe quel ordre !

Bonus 1

Actuellement, que ça soit sur téléphone, phablette ou tablette, la liste des prévisions météos affichées sur l'écran home se fait sur une seule colonne.

Il convient de modifier le code existant avant que :

- sur téléphone : la liste s'affiche sur une seule colonne ;
- sur tablette : la liste s'affiche sur deux colonnes.

Bonus 2

Actuellement, l'application est franco-française. Il convient de mettre en place une solution permettant que le mot "city" s'affiche à la place du mot

"ville" dans le cas d'un terminal en langue anglaise.

Bonus 3

Actuellement, au niveau de la popup de paramètres, lorsque l'utilisateur clique sur le champ lui permettant de saisir le nombre de jours qu'il souhaite voir afficher à l'écran, c'est le clavier "classique" qui s'ouvre. Puisque ce champ est destiné à saisir un nombre faites en sorte que le clavier qui soit proposés à l'utilisateur soit un clavier de type "numérique".

Bonus 4

Actuellement, les différents écrans de l'application supportent la rotation, à savoir un affichage en mode portrait et paysage. Faites en sorte que l'écran de détail ne puisse plus subir une rotation et que son orientation soit bloquée en mode portrait.

Bonus 5

Actuellement, l'application ne géolocalise pas réellement l'utilisateur. Faites en sorte que l'écran principal de l'application géolocalise l'utilisateur. Cette géolocalisation n'aura aucun impact sur le fonctionnel de l'application et les coordonnées de l'utilisateur pourront simplement être affichées dans le logcat à travers une log de type "info".