L'objectif du jour est de développer une application affichant la météo. Elle devra aller chercher les données météorologiques en format JSON sur le web (Section 1), puis les parser et les afficher sous forme de liste, en utilisant un `RecyclerView` (Section 2).

1 Récupérer les données météorologiques

Dans cette section, on cherche simplement à effectuer la requête HTTP afin de récupérer les données météorologiques. Celles-ci seront ensuite traitées et affichées en Section 2.

Question 1 Commencez par jeter un œil aux données, et à la manière dont le fichier JSON est structuré. Pour cela, utilisez votre navigateur pour visiter l'adresse suivante : `https://lacl.fr/julien.grange/Enseignements/Programmation_mobile/24_25/TP6/fakeweather.php`.

Malgré son nom trompeur, ce fichier est bien un fichier JSON. À chaque requête, de nouvelles données météo aléatoires sont générées. Rafraîchissez votre navigateur pour vous en convaincre.

Question 2 Créez un nouveau projet. Dans le `onCreate()` de la `MainActivity`, utilisez la bibliothèque **Volley** pour créer une nouvelle `RequestQueue`.

N'oubliez pas de demander dans le Manifest la permission de se connecter à internet.

Question 3 On constate que le contenu du fichier JSON est un tableau.

Créez donc une `JSONArrayRequest`, avec les callbacks suivants :

- En cas de bonne réception du fichier `fakeweather.php` sous forme de `JSONArray`, on affichera dans une `TextView` le contenu de la première case du tableau.
- En cas d'erreur, on affichera un `Toast` indiquant que la connexion n'a pas pu s'effectuer, et on enregistrera dans le `Log` la raison de l'erreur.

Ajoutez cette `JSONArrayRequest` à la `RequestQueue`. Vous devriez à ce stade obtenir un résultat similaire à ce qui est représenté en Figure 1, aux valeurs près.

2 Traitement et affichage des données

Maintenant qu'on sait aller chercher les données météorologiques sur le serveur distant, on veut les formater et les afficher sous forme de liste, au sein d'une `RecyclerView`.

Question 4 Créez une classe `Weather` pouvant contenir toutes les informations météorologiques correspondant à une ville, avec ses accesseurs en lecture et un constructeur complet.

Question 5 Dotez cette classe d'une méthode statique

```
public static Weather jsonToWeather(JSONObject o) throws JSONException;
```

qui parse un `JSONObject` représentant la météo dans une ville, et crée un nouveau `Weather` à partir de ces données.

Maintenant que l'on sait encapsuler les données météorologiques, on va s'attaquer à la partie "layout".

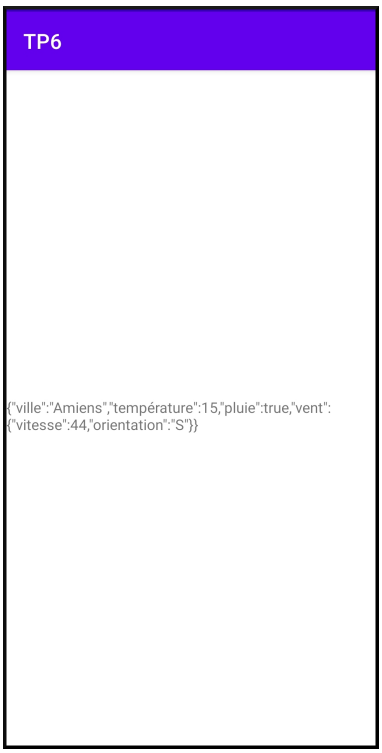

FIGURE 1 – Affichage du premier bulletin météo.

Question 6 Créez une classe `WeatherView` qui hérite de `ConstraintLayout`. Un tel objet aura trois attributs de type `TextView` nommés `villeView`, `tempView` et `ventView`, qui sont destinés à contenir respectivement le nom de la ville, la température, et les informations sur le vent.

FIGURE 2 – Contraintes sur les trois `TextView` : `villeView` est dans le coin en haut à gauche ; `tempView` est la droite de `villeView`, et centrée verticalement sur elle ; `ventView` est à la droite de `tempView`, et centrée verticalement sur elle.

Dans le constructeur, on créera trois nouvelles `TextView`, on n'oubliera pas d'attribuer des ID à toutes les `View`, et on imposera des contraintes en suivant le schéma présenté en Figure 2.

Pour marquer la différence de police, on pourra par exemple utiliser

```
TextViewCompat.setTextAppearance(villeView,
    android.R.style.TextAppearance_Large);
```

Question 7 Écrivez une méthode `WeatherView::setWeather()` qui prend en argument un `Weather`, et qui remplit proprement les différents champs de texte.

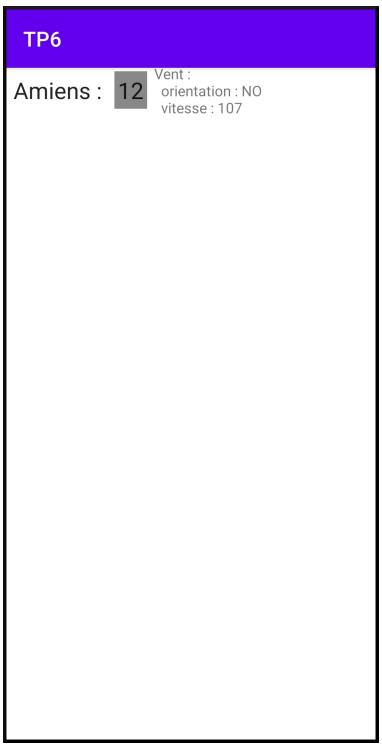

FIGURE 3 – Aperçu d'une `WeatherView`.

La pluie, ou son absence, sera représentée par la couleur de `tempView` : s'il pleut, celle-ci sera grise, sinon elle sera bleue. On utilisera pour cela la méthode `View::setBackgroundColor()`.

Testez votre code en créant, dans le `JSONArrayRequest`, une `WeatherView` affichant les données de la première case. Le résultat obtenu devrait ressembler à la Figure 3.

Terminons maintenant notre application en affichant ces `WeatherView` dans un `RecyclerView`.

Question 8 Créez une classe `WeatherViewHolder` héritant de `RecyclerView.ViewHolder`, ainsi qu'une classe `WeatherAdapter` héritant de `RecyclerView.Adapter<WeatherViewHolder>`, qui aura comme attribut une liste de `Weather`.

Implémentez les méthodes suivantes dans ces deux classes (celles-ci sont très similaires à celles exposées en cours) :

- Dans `WeatherViewHolder` :
  - un constructeur `WeatherViewHolder(View weatherView)`
  - `WeatherViewHolder::setWeather(Weather weather)`
- Dans `WeatherAdapter` :
  - `void addWeather(Weather weather)`
  - `WeatherViewHolder onCreateViewHolder(ViewGroup parent, int viewType)`
  - `void onBindViewHolder(WeatherViewHolder holder, int position)`
  - `int getItemCount()`

Question 9 Placez un `RecyclerView` dans le XML, et initialisez-le dans le `onCreate()` avec les bons attributs (en plus du `WeatherAdapter`, on choisira un `LinearLayoutManager`).

Question 10 Enfin, modifiez votre `JSONArrayRequest` pour que la réception des données météorologiques remplisse le `WeatherAdapter`. Vous devriez maintenant obtenir un résultat similaire à la Figure 4.

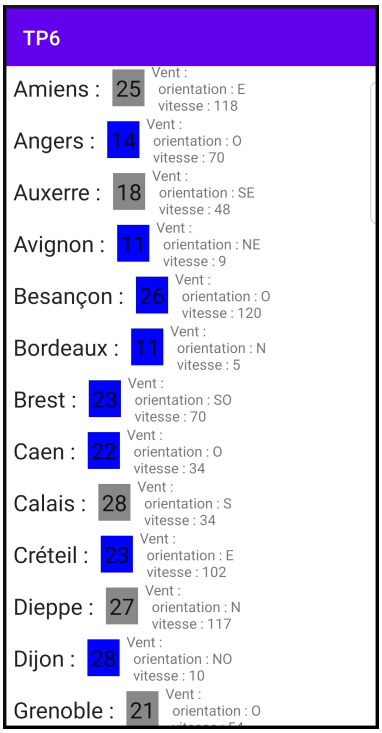

FIGURE 4 – Aperçu du résultat final.