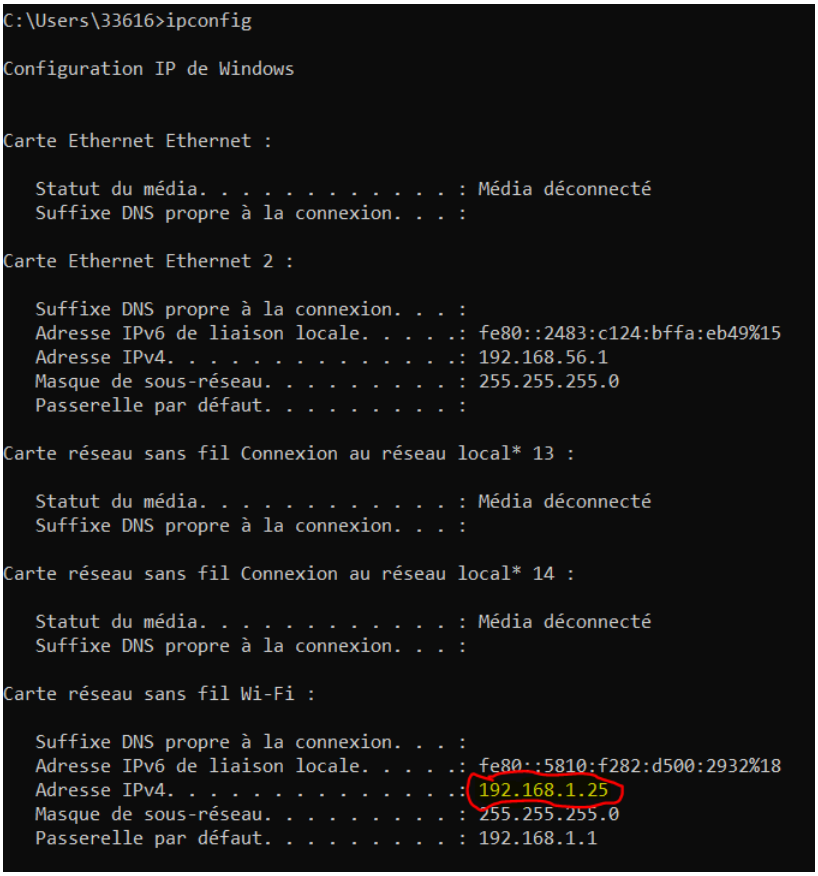
Chapter 1

Rapport TP3 Mobile

1.1 Introduction

Le projet a été fait sous Intelij en utilisant mon téléphone pour exécuter l'application. Mon téléphone a une résolution d'écran de $2310 \times 1080 pixels$.

Pour faire fonctionner l'application avec le serveur REST il faut d'abord éxécuter la commande ipconfig sur un terminal.



Copier l'adresse IP et remplacer l'adresse IP dans la fonction void télécharger(View rooter) dans la classe Fragment1_saisi.

```
public Formulaire(Context context) throws IOException {
public void telecharger(View rooter){
   RequestQueue queue = Volley.newRequestQueue(rooter.getContext());
    String url ="http://COPIERICI:8080/formulaire/api/form";
    StringRequest stringRequest = new StringRequest(Request.Method.GET, url
    ,
            new Response.Listener<String>() {
                @Override
                public void onResponse(String response) {
                    Gson gson = new Gson();
                    f = gson.fromJson(response, Formulaire.class);
                    updateaffichage();
                }
            }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            System.out.println("marche pas");
        }
    });
    queue.add(stringRequest);
}
```

et dans la fonction void Postrequest() de la classe Fragment2_affichage.

```
public void Postrequest() throws JSONException {
    Gson gson = new Gson();
    String json = gson.toJson(f);
    JSONObject parameters = new JSONObject(json);
    RequestQueue queue = Volley.newRequestQueue(getActivity());
    String url = "http://COPIERICI:8080/formulaire/api/formulaire";
    JsonObjectRequest jsonRequest = new JsonObjectRequest(Request.Method.
    POST, url, parameters, new Response.Listener<JSONObject>() {
        @Override
        public void onResponse(JSONObject response) {
        }
    }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
```

```
14                 error.printStackTrace();
15             }
16         });
17         queue.add(jsonRequest);
18     }
```

1.2 Partie application mobile

1.2.1 Classe Formulaire

La classe Formulaire implémente l'interface Parcelable qui permettra de faire passer l'objet entre différents fragments de l'application.

Cette classe contient toutes les informations transmises par l'utilisateur via le téléphone (nom, prénom, mail ,etc..) Elle contient aussi une List de Hobbit. La classe Hobbit contient deux attributs : une chaîne de caractères nom et un boolean qui indiquent si ce hobbit est pratiqué par l'utilisateur ou pas.

La classe formulaire contient trois constructeurs

— Le premier constructeur est un constructeur basique où il faut passer en paramètre les données de la classe.
— Le deuxième constructeur prend en paramètre un Context qui va initialiser ces paramètres en lisant un fichier Json.
— Le dernier constructeur est utilisé par le parceur pour passer l'object Formulaire entre le Fragment1_saisi et fragment2_affichage

Code du constructeur initialisé par un fichier Json :

```
1  public Formulaire(Context context) throws IOException {
2  /*Ouvre le fichier Json en mode lecture*/
3      InputStream input = context.openFileInput("donner.json");
4      byte[] buffer=new byte[input.available()];//initialise un tableau de
       Byte qui va contenir le contenu du fichier
5      input.read(buffer);//lie le fichier Json et met les valeurs dans le
       Buffer
6      input.close();//referme le fichier
7      String text=new String(buffer);//convertit le buffer en chaine de
       caractères
8  /*Je convertis ensuite la chaine de caractères (qui est sous format Json)
       en Formulaire*/
9      Gson gson = new Gson();
10     Formulaire f = gson.fromJson(text, Formulaire.class);
11 /* initialise ensuite chaque attribut du formulaire*/
12     this.nom = f.nom;
13     this.prenom = f.prenom;
14     this.date = f.date;
15     this.num = f.num;
16     this.mail =f.mail;
17     this.hobbit = f.hobbit;
18 }
```

Cette classe contient aussi une fonction ecrire qui permet d'écrire le formulaire sur le téléphone sous le format Json.

```
1  public void ecrire(Context context){
2      try {
3      /*Ouvre le fichier(le créer si le fichier n'existe pas) Json en mode é
       criture*/
4          FileOutputStream fOut = context.openFileOutput("donner.json",0);
5      /*Je convertis le Formulaire en un String qui contiendra du Json*/
6          Gson gson = new Gson();
7          String json = gson.toJson(this);
8          fOut.write(json.getBytes());//écrit la chaine de caractères dans le
        fichier
9          fOut.close();//referme le fichier
10     }
11     catch (Exception e) {
```

```
12            e.printStackTrace();
13        }
14    }
```

1.2.2 Classe Fragment1_saisi

La classe fragment1_saisi est la classe qui permet d'afficher le fragment pour que l'utilisateur rentre les données du formulaire

La classe Fragment est composée de la fonction updateaffichage() qui permet de réactualiser le texte des EditText en fonction du téléchargement ou de la lecture d'un formulaire.

```
1  public void updateaffichage(){
2      TNom.setText(f.getNom());
3      TPrenom.setText(f.getPrenom());
4      TDate.setText(f.getDate());
5      TMail.setText(f.getMail());
6      TNum.setText(String.valueOf(f.getNum()));
7      checksport.setChecked(f.getHobbit().get(0).isVal());
8      checkmusique.setChecked(f.getHobbit().get(1).isVal());
9      checklecture.setChecked(f.getHobbit().get(2).isVal());
10 }
```

La fonction Lecturefichier() qui vérifie s'il existe un fichier Json avec un formulaire dedans et si oui appelle le constructeur de Formulaire.

```
1  public void Lecturefichier(){
2  /*Je récupère le Path du fichier Json*/
3      String file_name=getActivity().getFilesDir() + "/"+"donner.json";
4  /*Pour ensuite vérifier si le fichier Json existe et j'effectue la lecture
       que si le fichier n'existe pas*/
5      File t = new File(file_name);
6      if(t.exists()){
7          try {
8          /*appelle du constructeur de Formulaire qui initialise ces données
       par un fichier Json*/
```

```
9                 f = new Formulaire(getActivity());
10            } catch (IOException e) {
11                e.printStackTrace();
12            }
13            updateaffichage();//actualise l'affichage des EditText
14        }
15    }
```

La fonction telecharger(View rooter) qui permet de télécharger le formulaire présent dans le serveur Web REST et actualise ensuite les EditText avec le nouveau formulaire.

```
1  public void telecharger(View rooter){
2  /*création d'une Liste de request REST à éxécuter*/
3      RequestQueue queue = Volley.newRequestQueue(rooter.getContext());
4      /* j'initialise l'adresse de la fonction GET du serveur REST*/
5      String url ="http://192.168.1.25:8080/formulaire/api/form";
6      /*éxécute la requet GET sur le serveur*/
7      StringRequest stringRequest = new StringRequest(Request.Method.GET, url
       ,
8          new Response.Listener<String>() {
9              @Override
10             public void onResponse(String response) {
11             /*récupère le fichier Json envoyé par le serveur et le
       convertit en Formulaire*/
12                 Gson gson = new Gson();
13                 f = gson.fromJson(response, Formulaire.class);
14                 updateaffichage();//j'actualise ensuite l'affichage pour
       mettre celui qui vient d'être télécharger.
15                 }
16             }, new Response.ErrorListener() {
17         @Override
18         public void onErrorResponse(VolleyError error) {
19             System.out.println("marche pas");
20         }
21     });
22     queue.add(stringRequest);//met la requet précédente la pile de request
       à éxécuter.
23 }
```

Puis la fonction changementFragment() qui permet de passer entre le Fragment1_saisi vers le fragment fragment2_affichage tout en transmettant le formulaire saisi par l'utili-

sateur et en mentionnant s'il doit être synchronisé avec le serveur Web REST.

```
1  public void changementFragment(){
2      Bundle bundle = new Bundle();//intilialisation du Bundle
3  /*Je récupère les données saisies par l'utilisateur pour créer le
       Formulaire*/
4      ArrayList<Hobbit> t = new ArrayList<>();
5      t.add( new Hobbit("Sport", checksport.isChecked() ,0));
6      t.add( new Hobbit("Musique", checkmusique.isChecked() ,1));
7      t.add( new Hobbit("Lecture", checklecture.isChecked() ,2));
8      f = new Formulaire(TNom.getText().toString() ,TPrenom.getText().toString
       () ,TDate.getText().toString() ,TNum.getText().toString() ,TMail.getText().
       toString() ,t);
9
10     bundle.putParcelable("FORMULAIRE" , f);//envoie le formulaire au
       fragment fragment2_affichage
11     bundle.putBoolean("SYNCHRONISER" ,checksynchroniser.isChecked());//
       envoie si l'utilisateur à demandé une synchronisation avec le serveur.
12     /*initialisation du Fragment de transaction*/
13     FragmentManager fragmentManager = getFragmentManager();
14     FragmentTransaction fragmentTransaction = fragmentManager.
       beginTransaction();
15     /*initialisation du 2 éme fragment*/
16     fragment2_affichage fragment  = new fragment2_affichage();
17     fragment.setArguments(bundle);//passage des paramètres à transmettre
18     /*remplacement du fragment Fragment1_saisi par fragment2_affichage*/
19     fragmentTransaction.replace(R.id.fragmentContainerView , fragment);
20     fragmentTransaction.commit();
21 }
```

Et la fonction principale de Fragment1_saisi est View onCreateView(LayoutInflater inflater, ViewGroup container,Bundle savedInstanceState) qui va permettre de gérer les ChekBox,EditText , etc... du fragment.

Pour l'EditText qui permet de rentrer le numéro de téléphone j'utilise la fonction :

```
1  TNum.addTextChangedListener(new PhoneNumberFormattingTextWatcher());
```

Pour que le numéro soit affiché sous un format de numéro de téléphone (un espace tout

les 2 chiffres)

Pour que l'EditText qui permet de rentrer la date soit sous un format de date j'utilise le code :

```
1  public void onTextChanged(CharSequence s, int start, int before, int count)
       {
2      if (!s.toString().equals(current)) {
3          String clean = s.toString().replaceAll("[^\\d.]|\\.", "");
4          String cleanC = current.replaceAll("[^\\d.]|\\.", "");
5          int cl = clean.length();
6          int sel = cl;
7          for (int i = 2; i <= cl && i < 6; i += 2) sel++;
8          if (clean.equals(cleanC)) sel--;
9          if (clean.length() < 8) clean = clean + ddmmyyyy.substring(clean.
       length());
10         else{
11             int day  = Integer.parseInt(clean.substring(0,2));
12             int mon  = Integer.parseInt(clean.substring(2,4));
13             int year = Integer.parseInt(clean.substring(4,8));
14             mon = mon < 1 ? 1 : mon > 12 ? 12 : mon;
15             cal.set(Calendar.MONTH, mon-1);
16             year = (year<1900)?1900:(year>2100)?2100:year;
17             cal.set(Calendar.YEAR, year);
18             day = (day > cal.getActualMaximum(Calendar.DATE))? cal.
       getActualMaximum(Calendar.DATE):day;
19             clean = String.format("%02d%02d%02d",day, mon, year);
20             }
21         clean = String.format("%s/%s/%s", clean.substring(0, 2),
22                 clean.substring(2, 4),
23                 clean.substring(4, 8));
24         sel = sel < 0 ? 0 : sel;
25         current = clean;
26         TDate.setText(current);
27         TDate.setSelection(sel < current.length() ? sel : current.length())
       ;
28     }
29 }
```

1.2.3 Classe Fragment2_affichage

La classe Fragment2_affichage est un fragment qui permet d'afficher le contenu qu'a saisi l'utilisateur sur Fragment1_saisi. Elle permet aussi d'écrire le contenu saisi dans un fichier sous le format Json et si la checkBox synchroniser du Fragment1_saisi a été coché alors il synchronise le formulaire sur le serveur REST.

La classe Fragment2_affichage contient plusieurs fonctions.

Une fonction ChangementFragment() qui comme pour Fragment1_saisi permet de changer le fragment. Dans notre cas dès qu'on appuie sur le bouton retour le fragment passe sur Fragment1_saisi.

```
1  public void ChangementFragment(){
2      /*initialisation du Fragment de transaction*/
3      FragmentManager fragmentManager = getFragmentManager();
4      FragmentTransaction fragmentTransaction = fragmentManager.
       beginTransaction();
5          /*initialisation du 2 éme fragment*/
6      Fragment1_saisi fragment  = new Fragment1_saisi();
7          /*remplacement du fragment Fragment2_affichage par fragment1_saisi
       */
8      fragmentTransaction.replace(R.id.fragmentContainerView , fragment);
9      fragmentTransaction.commit();
10 }
```

La fonction Postrequest() permet d'envoyer le nouveau formulaire au serveur REST qui va pouvoir ensuite mettre à jour son formulaire.

```
1  public void Postrequest() throws JSONException {
2  /*convertion du formulaire en une chaine de caractères Json*/
3      Gson gson = new Gson();
4      String json = gson.toJson(f);
5      /*créer un JSONObject à passer en paramètre de la fonction POST*/
6      JSONObject parameters = new JSONObject(json);
7      /*Création d'une pile de request à exécuter*/
8      RequestQueue queue = Volley.newRequestQueue(getActivity());
9      /*intialisation de l'adresse de la request POST du serveur REST*/
10     String url = "http://192.168.1.25:8080/formulaire/api/formulaire";
```

```
11      /*initialise la request POST avec en paramètre le JSONObject*/
12      JsonObjectRequest jsonRequest = new JsonObjectRequest(Request.Method.
        POST, url, parameters, new Response.Listener<JSONObject>() {
13          @Override
14          public void onResponse(JSONObject response) {
15          }
16      }, new Response.ErrorListener() {
17          @Override
18          public void onErrorResponse(VolleyError error) {
19              error.printStackTrace();
20          }
21      });
22      queue.add(jsonRequest);//met la request dans la pile d'éxécution.
23  }
```

Puis une dernière fonction View onCreateView(LayoutInflater inflater, ViewGroup container,Bundle savedInstanceState) qui elle permet de gérer l'affichage des TextView et gère les évènements des Boutons.

1.3 Parti Serveur Web REST

Le serveur REST a été fait avec SpringBoot. Le programme contient quatre packages.

— Un package models qui va contenir les classes principales du serveur telles que Formulaire et Hobbit.

— Un package Data qui va permettre d'initialiser le formulaire au démarrage du serveur par un formulaire pré-généré.

— Un package repository qui permet de créer les repository pour chaque classe afin d'interagir avec la BDD du serveur.

— Un package controlers qui va contenir toutes les fonctions PUT, DELETE, POST et GET afin le lien entre l'application du téléphone et le serveur REST.

1.3.1 models

Le package models contient les classes Formulaire et Hobbit qui contiennent les mêmes attributs que les classes Formulaire et Hobbit de l'application mobile.

1.3.2 Controlers

Le controllers du serveur REST contient deux fonctions.
Une fonction GET qui va retourner le formulaire présent dans le serveur.

```
1  private static final String uri = "formulaire/api";
2  @GetMapping(uri + "/form")//adresse de la de la request sous l'adresse ip/
       formulaire/api/form
3  public Formulaire Getformulaire() {
4  /*je retourne le formulaire avec l'id de 0(le seul formulaire du serveur)*/
5      Optional<Formulaire> c  = formulairerepository.findById((long) 0);
6      return c.get();
7  }
```

La deuxième fonction est une fonction POST qui va récupérer le formulaire envoyé par l'application et mettre à jour son formulaire à lui.

```
1  @ResponseStatus(HttpStatus.CREATED)
2  @PostMapping(uri+"/formulaire")//adresse de la de la request sous l'adresse
       ip/formulaire/api/formulaire
3  public Formulaire Postformulaire(@RequestBody Formulaire f) {
4  /*je récupère le formulaire envoyé par l'application mobile et mettre à
       jour les Hobbits*/
5  /*Met à jour le Hobbit sport*/
6      this.hobbitrepo.findById((long)0).map(Hobbit -> {
7                  Hobbit.setNom(f.getHobbit().get(0).getNom());
8                  Hobbit.setVal(f.getHobbit().get(0).isVal());
9                  return hobbitrepo.save(Hobbit);
10             })
11             .orElseGet(() -> hobbitrepo.save(f.getHobbit().get(0)));
12             /*Met a jour le Hobbit musique*/
13     this.hobbitrepo.findById((long)1).map(Hobbit -> {
14                 Hobbit.setNom(f.getHobbit().get(1).getNom());
15                 Hobbit.setVal(f.getHobbit().get(1).isVal());
16                 return hobbitrepo.save(Hobbit);
17             })
18             .orElseGet(() -> hobbitrepo.save(f.getHobbit().get(1)));
19              /*Met a jour le Hobbit lecture*/
20     this.hobbitrepo.findById((long)2).map(Hobbit -> {
21                 Hobbit.setNom(f.getHobbit().get(2).getNom());
22                 Hobbit.setVal(f.getHobbit().get(2).isVal());
```

```
23                  return hobbitrepo.save(Hobbit);
24              })
25              .orElseGet(() -> hobbitrepo.save(f.getHobbit().get(2)));
26
27  /* je mets ensuite à jour le formulaire avec l id 0 (le seul formulaire pré
        sent dans le serveur)*/
28      return formulairerepository.findById((long) 0)
29              .map(formulaire -> {
30                  formulaire.setNom(f.getNom());
31                  formulaire.setPrenom(f.getPrenom());
32                  formulaire.setDate(f.getDate());
33                  formulaire.setNum(f.getNum());
34                  formulaire.setMail(f.getMail());
35                  return formulairerepository.save(formulaire);
36              })
37              .orElseGet(() -> formulairerepository.save(f));
38  }
```

1.3.3 Data

Le dernier package contient une classe Formulairedata qui va permettre d'initaliser le formulaire par un formulaire créé arbitrairement.

```
1   @Bean
2   /*cette fonction va s'éxécuter au démarrage du serveur*/
3   public CommandLineRunner initDatabase(FormulaireRepository rep ,
        HobbitRepository he ) {
4       return args -> {
5           Formulaire f = new Formulaire(0,"Monot","Lea","03/03/2000","06 66
        66 66 66","lea.monot@gmail.com");//création d'un premier formulaire
6
7           rep.save(f);//enregistre ce formulaire dans la BDD
8           /*initialise les Hobbits*/
9           Hobbit h = new Hobbit(0,"Sport",true,f);
10          Hobbit h2 = new Hobbit(1,"Musique",false,f);
11          Hobbit h3 = new Hobbit(2,"Lecture",true,f);
12          /*enregistre les Hobbits dans la BDD*/
13          he.save(h);
14          he.save(h2);
15          he.save(h3);
16      };
```

```
17  }
```