File - D:\4. Softwares\AndroidStudioProjects\CookBook\app\src\main\java\com\fdx\cookbook\Note.java

```java
package com.fdx.cookbook;

import java.util.Date;

public class Note {
    private Integer mNote;
    private User mUser;
    private Date mDate;


    public Note(Integer note, User u){
        mNote=note;
        mUser=u;
        mDate=new Date();
    }

    public Note(Integer note, User u, Date d){
        mNote=note;
        mUser=u;
        mDate=d;
    }

    public Integer getNote() {
        return mNote;
    }

    public User getUser() {
        return mUser;
    }

    public Date getDate() {
        return mDate;
    }

    public void setNote(Integer note) {
        mNote = note;
    }

    public void setUser(User user) {
        mUser = user;
    }

    public void setDate(Date date) {
        mDate = date;
    }

    @Override
    public boolean equals(Object o){
        if (o == null) return false;
        if (o == this) return true;
        if (getClass() != o.getClass()) return false;
        Note n=(Note) o;
        boolean b=(this.mNote.equals(n.getNote()));
        b=b && (mDate.toString().equals(n.getDate().toString()));
        b=b && (mUser.equals(n.getUser()));
        return b;
    }
```

```
58 }
59
```

File - D:\4. Softwares\AndroidStudioProjects\CookBook\app\src\main\java\com\fdx\cookbook\User.java

```java
package com.fdx.cookbook;

import java.util.Date;
import java.util.UUID;

public class User {
    private UUID mId;
    private String mFamily;
    private String mName;
    private Date mDate;
    private static final String TAG = "DebugUser";

    public User(String family, String name){
        mFamily=family;
        mName=name;
        mId =UUID.randomUUID();
        mDate=new Date();
    }

    public User(){
        mFamily="not found";
        mName="not found";
        mId =UUID.randomUUID();
        mDate=new Date();
    }
    public User(UUID uuid){
        mId =uuid;
        // waiting for user database
        //todo erase !
        switch(mId.toString()){
            case "c81d4e2e-bcf2-11e6-869b-7df92533d2db":
                mFamily="Devaux_Lion de ML";
                mName="Fabrice";
                return;
            case "c81d4e2e-bcf2-11e7-869b-7df92533d2db":
                mFamily="Devaux_Lion de ML";
                mName="Lucile";
                return;
            case "c81d4e2e-bcf3-11e6-869b-7df92533d2db":
                mFamily="Devaux_Lion de ML";
                mName="Véronique";
                return;
            default:
                mFamily="not found";
                mName="not found";
        }
    }

    public UUID getId() {
        return mId;
    }

    public void setId(UUID id) {
        this.mId = id;
    }

    public String getFamily() {
```

```
 58          return mFamily;
 59      }
 60
 61      public void setFamily(String family) {
 62          mFamily = family;
 63      }
 64
 65      public String getName() {
 66          return mName;
 67      }
 68
 69      public void setName(String name) {
 70          mName = name;
 71      }
 72
 73      public String getNameComplete() {
 74          return mName+"@"+ mFamily;
 75      }
 76
 77      public boolean IsEqual(User r){return (mId.toString().equals(r.
    getId().toString()));}
 78
 79      public Date getDate() {
 80          return mDate;
 81      }
 82
 83      public void setDate(Date date) {
 84          mDate = date;
 85      }
 86
 87      @Override
 88      public boolean equals(Object o){
 89          if (o == null) return false;
 90          if (o == this) return true;
 91          if (getClass() != o.getClass()) return false;
 92          User u=(User) o;
 93          return u.getId().toString().equals(mId.toString());
 94      }
 95      //todo P2 épurer affichage quand champs vides
 96      //todo P2 tuto swipe page
 97      //todo P1 messagerie
 98
 99  }
100
```

```java
package com.fdx.cookbook;

import android.graphics.Bitmap;

import com.google.gson.Gson;
import com.google.gson.reflect.TypeToken;

import java.lang.reflect.Type;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.UUID;


enum StatusRecipe {Submitted,Visible, Deleted }
public class Recipe {
    private UUID mId;
    private User mOwner;
    private Date mLastUpdateRecipe;
    private Date mLastUpdatePhoto;
    private String mTitle;
    private String mSource;
    private URL mSource_url;
    private int mNbPers;
    private String[] mSteps;
    private double mNoteAvg;
    private ArrayList<Note> mNotes;
    private ArrayList<Comment> mComments;
    private RecipeSeason mSeason;
    private RecipeType mType;
    private RecipeDifficulty mDifficulty;
    private String[] mIngredients;
    private StatusRecipe mStatus;
    private String mMessage;
    private User mIdFrom;
    private Boolean mTS_recipe;
    private Boolean mTS_photo;
    private Boolean mTS_comment;
    private Boolean mTS_note;
    private Bitmap mImage;

    private static final int NBSTEP_MAX=9;
    private static final int NBING_MAX=15;
    private static final int NBCOM_MAX=20;
    private String TAG="CB_Recipe";
    private String DEFAULT_URL="https://wwww.cookbookfamily.com";
    private String UUIDNULL="00000000-0000-0000-0000-000000000000";



    public Recipe() {
        this(UUID.randomUUID());
    }

    public Recipe( UUID id){
```

```
 58          mId=id;
 59          mTitle="";
 60          mLastUpdateRecipe =new Date();
 61          Calendar c=Calendar.getInstance();
 62          c.set(2000,0,1,0,0, 0);
 63          mLastUpdateRecipe=c.getTime();
 64          mLastUpdatePhoto=c.getTime();
 65          mSteps = new String[NBSTEP_MAX];
 66          mIngredients= new String[NBING_MAX];
 67          for(int i=0;i<NBSTEP_MAX;i++){mSteps[i]="";}
 68          for(int i=0;i<NBING_MAX;i++){mIngredients[i]="";}
 69          mNbPers=4;
 70          mSeason=RecipeSeason.ALLYEAR;
 71          mType=RecipeType.MAIN;
 72          mDifficulty=RecipeDifficulty.UNDEFINED;
 73          mStatus=StatusRecipe.Visible;
 74          mComments=new ArrayList<Comment>();
 75          mNotes=new ArrayList<Note>();
 76          mSource="";
 77          mIdFrom=new User(UUID.fromString( UUIDNULL));
 78          try {mSource_url=new URL(DEFAULT_URL);
 79          } catch (MalformedURLException e) {}
 80          mTS_recipe=false;
 81          mTS_photo=false;
 82          mTS_comment=false;
 83          mTS_note=false;
 84      }
 85
 86      public Date getDate() {
 87          return mLastUpdateRecipe;
 88      }
 89
 90      public Date getDatePhoto() {
 91          return mLastUpdatePhoto;
 92      }
 93
 94      public String getTitle() {
 95          return mTitle;
 96      }
 97
 98      public void setDate(Date date) {mLastUpdateRecipe = date;}
 99
100      public void setDatePhoto(Date date) {mLastUpdatePhoto = date;}
101
102      public void setTitle(String title) {
103          mTitle = title;
104      }
105
106      public int getNbPers() {
107          return mNbPers;
108      }
109
110      public void setNbPers(int nbPers) {
111          mNbPers = nbPers;
112      }
113
114      public String getMessage() {
```

```
115         return mMessage;
116     }
117
118     public void setMessage(String message) {
119         mMessage=message;
120     }
121
122
123     // -----------------STEP-------------------
124     public void setStep(Integer i, String step) {
125         if ((i > 0) && (i <= NBSTEP_MAX)) {
126             mSteps[i - 1] = step;
127         }
128     }
129
130     public String getStep(Integer i){
131         if ((i>0)&&(i<=NBSTEP_MAX)) {return mSteps[i-1];}
132         else{ return "";}
133     }
134
135     public int getNbStep(){
136         int j=0;
137         for(int i=NBSTEP_MAX; i>0; i--){
138             if (!mSteps[i-1].isEmpty()) {j=i; break;}}
139         return j;
140     }
141
142     public int getNbStepMax(){return NBSTEP_MAX;}
143
144     //--------------Ingredients--------------------
145     public void setIngredient(Integer i, String ing) {
146         if ((i > 0) && (i <= NBING_MAX)) {
147             mIngredients[i-1] = ing;
148         }
149     }
150
151     public String getIngredient(Integer i){
152         if ((i>0)&&(i<=NBING_MAX)) {return mIngredients[i-1];}
153         else{ return "";}
154     }
155     public int getNbIng(){
156         int j=0;
157         for(int i=NBING_MAX; i>0; i--){
158             if (!mIngredients[i-1].isEmpty()) {j=i; break;}}
159         return j;
160     }
161
162     public int getNbIngMax(){return NBING_MAX;}
163
164     //------------------- Users  ---------------------------------
165     public UUID getId() {
166         return mId;
167     }
168
169     public User getUserFrom() {
170         return mIdFrom;
171     }
```

```
172
173     public void setUserFrom(User idFrom) {
174         mIdFrom = idFrom;
175     }
176
177     public void setId(UUID id) {mId = id; }
178
179     public User getOwner() {
180         return mOwner;
181     }
182
183     public String getOwnerIdString() {
184         return mOwner.getId().toString();
185     }
186
187     public void setOwner(User owner) {
188         mOwner = owner;
189     }
190
191     //------------------- Source ---------------------------------
192     public String getSource() {
193         if (mSource==null) return "";
194         return mSource;
195     }
196     public void setSource(String source) {
197         mSource = source;
198     }
199     public URL getSource_url() {
200         return mSource_url;
201     }
202
203     public String getSource_url_name() {
204         String ret=mSource_url.toString();
205         if (ret.equals("https:") || ret.equals("http:")){ret="";}
206         return ret;
207     }
208
209     public void setSource_url(URL source_url) {
210         mSource_url = source_url;
211     }
212
213     // ----------------------Enum ----------------------
214
215     public RecipeSeason getSeason() {
216         return mSeason;
217     }
218
219     public void setSeason(RecipeSeason season) {
220         mSeason = season;
221     }
222
223     public boolean IsSummer(){
224         switch (mSeason) {
225             case SUMMER:
226                 return true;
227             case ALLYEAR:
228                 return true;
```

```
229             default:
230                 return false;
231         }
232     }
233     public boolean IsWinter(){
234         switch (mSeason) {
235             case WINTER:
236                 return true;
237             case ALLYEAR:
238                 return true;
239             default:
240                 return false;
241         }
242     }
243     public RecipeType getType() {
244         return mType;
245     }
246     public void setType(RecipeType type) {
247         mType = type;
248     }
249 
250     public RecipeDifficulty getDifficulty() {
251         return mDifficulty;
252     }
253 
254     public void setDifficulty(RecipeDifficulty difficulty) {
255         mDifficulty = difficulty;
256     }
257 
258     public StatusRecipe getStatus() {return mStatus;}
259     public void setStatus(StatusRecipe s) {
260         mStatus = s;
261     }
262     public boolean IsVisible(){
263         if (mStatus==StatusRecipe.Visible) {return true;}
264         return false;
265     }
266     public boolean IsMessage(){
267         if (mStatus==StatusRecipe.Submitted) {return true;}
268         return false;
269     }
270     public boolean IsMarkedDeleted(){
271         if (mStatus==StatusRecipe.Deleted) {return true;}
272         return false;
273     }
274 
275     public void updateTS(AsynCallFlag asyn, Boolean b){
276         switch(asyn){
277             case NEWRECIPE:{mTS_recipe=b;return;}
278             case NEWPHOTO:{mTS_photo=b;return;}
279             case NEWRATING:{mTS_note=b;return;}
280             case NEWCOMMENT:{mTS_comment=b;return;}
281         }
282         return;
283     }
284 
285     public int getTS(AsynCallFlag asyn){
```

```
286         Boolean b=false;
287         switch(asyn){
288             case NEWRECIPE:{b=mTS_recipe; break;}
289             case NEWPHOTO:{b=mTS_photo; break;}
290             case NEWRATING:{b=mTS_note; break;}
291             case NEWCOMMENT:{b=mTS_comment; break;}
292         }
293         return (b ? 1:0);
294     }
295 
296     //-------------------- photo filename--------------------
297     public String getPhotoFilename(){
298         return "IMG"+getId().toString()+".jpg";
299     }
300 
301     //--------------------- Arraylist Comments et Notes
    ------------------------
302     public void addComment(Comment c){ mComments.add(c);}
303 
304     public ArrayList<Comment> getComments() {return mComments;}
305 
306     public Comment getComment(int i){
307         if (i<mComments.size()){
308             return mComments.get(i);
309         } else {return null;}
310 
311     }
312 
313     public int getNbComMax(){return NBCOM_MAX;} // nb max affiché
314 
315     public void addNote(Note note){ mNotes.add(note);}
316 
317     public ArrayList<Note> getNotes() {return mNotes;}
318 
319     public Note getNote(int i){
320         if (mNotes==null) return null;
321         if (i<mNotes.size()){ return mNotes.get(i);}
322         else {return null;}
323     }
324 
325     public double getNoteAvg() {
326         mNoteAvg=0;
327         if (mNotes==null) return mNoteAvg;
328         if (mNotes.size()!=0) {
329             for(Note n:mNotes){mNoteAvg+=n.getNote();}
330             mNoteAvg=mNoteAvg/mNotes.size();
331         } else {mNoteAvg=0;}
332         return mNoteAvg;
333     }
334     public int getNbNotes() {
335         if (mNotes==null) return 0;
336         if (mNotes.isEmpty()) return 0;
337         return mNotes.size();
338     }
339 
340     //--------------------- Serialisation ------------------------
341     public String getSerializedComments(){
```

```
342         Gson gson = new Gson();
343         return gson.toJson(mComments);
344     }
345
346     public void getCommentsDeserialised(String raw){
347         Gson gson=new Gson();
348         Type listOfNotesObject = new TypeToken<ArrayList<Comment
    >>() {}.getType();
349         mComments=gson.fromJson(raw, listOfNotesObject);
350     }
351     public String getSerializedOwner(){
352         Gson gson = new Gson();
353         return gson.toJson(mOwner);
354     }
355     public String getSerializedFrom(){
356         Gson gson = new Gson();
357         return gson.toJson(mIdFrom);
358     }
359     public void getOwnerDeserialized(String raw){
360         Gson gson=new Gson();
361         mOwner=gson.fromJson(raw, User.class);
362     }
363     public void getFromDeserialized(String raw){
364         Gson gson=new Gson();
365         mIdFrom=gson.fromJson(raw, User.class);
366     }
367
368     public String getSerializedNotes(){
369         Gson gson = new Gson();
370         return gson.toJson(mNotes);
371     }
372
373     public void getNotesDeserialised(String raw){
374         Gson gson=new Gson();
375         Type listOfNotesObject = new TypeToken<ArrayList<Note>>() {}.
    getType();
376         mNotes=gson.fromJson(raw, listOfNotesObject);
377     }
378
379     // ******************* tests **********************
380     public Boolean hasNotChangedSince(Recipe r){
381         if (!mId.toString().equals(r.getId().toString())) return
    false;
382         if (!mOwner.getId().toString().equals(r.getOwner().getId().
    toString())) return false;
383         if (!mTitle.equals(r.getTitle())) return false;
384         if (!mSource.equals(r.getSource())) return false;
385         if (!mSource_url.equals(r.getSource_url())) return false;
386         if (mNbPers!=r.getNbPers()) return false;
387         for (int i = 0; i < r.getNbStepMax(); i++) {
388             if (!mSteps[i].equals(r.getStep(i+1))) return false;
389         }
390         for (int i = 0; i < r.getNbIngMax(); i++) {
391             if (!mIngredients[i].equals(r.getIngredient(i + 1)))
    return false;
392         }
393         if (!mSeason.toString().equals(r.getSeason().toString()))
```

```
393 return false;
394         if (!mType.toString().equals(r.getType().toString())) return
    false;
395         if (!mDifficulty.toString().equals(r.getDifficulty().toString
    ())) return false;
396         return true;
397     }
398 
399     public boolean containQuery(String s){ //for search request
400         if (sc(mTitle, s)) return true;
401         if (sc(mOwner.getFamily().toString(), s)) return true;
402         for(String sloop:mIngredients){
403             if (sloop==null) break;
404             if (sc(sloop, s)) return true;
405         }
406         for(String sloop:mSteps){
407             if (sloop==null) break;
408             if (sc(sloop, s)) return true;
409         }
410         for(Comment c:mComments){
411             if (c==null) break;
412             if (sc(c.getTxt(),s)) return true;
413         }
414         return false;
415     }
416 
417     private boolean sc(String s1, String s2){
418         return s1.toLowerCase().contains(s2.toLowerCase());
419     }
420 
421     public boolean isTheSame(Recipe rtotest){
422         return (rtotest.getId().toString().equals(mId.toString()));
423     }
424     // ----------------------Image ----------------------
425 
426     public Bitmap getImage() {
427         return mImage;
428     }
429 
430     public void setImage(Bitmap image) {
431         mImage = image;
432     }
433 }
434 
```

```java
package com.fdx.cookbook;

import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;

public class Comment {
    private String mTxt;
    private User mUser;
    private Date mDate;

    public Comment(){
        mTxt="";
        mUser=new User("","");
        mDate=new Date();
    }

    public Comment(String s, User u){
        mTxt=s;
        mUser=u;
        mDate=new Date();
    }
    public Comment(String s, User u, Date d){
        mTxt=s;
        mUser=u;
        mDate=d;
    }

    public String getTxt() {
        return mTxt;
    }

    public User getUser() {
        return mUser;
    }

    public Date getDate() {
        return mDate;
    }

    public void setDate(Date date) {
        mDate = date;
    }

    public String toTxt(){
        DateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy");
        String s= dateFormat.format(mDate);
        return mTxt+" ("+mUser.getNameComplete()+") - " + s;
    }

    @Override
    public boolean equals(Object o){
        if (o == null) return false;
        if (o == this) return true;
        if (getClass() != o.getClass()) return false;
        Comment c=(Comment) o;
        boolean b=(this.mTxt.equals(c.getTxt()));
```

```
58          b=b && (mDate.toString().equals(c.getDate().toString()));
59          b=b && (mUser.equals(c.getUser()));
60          return b;
61      }
62  }
63
```

File - D:\4. Softwares\AndroidStudioProjects\CookBook\app\src\main\java\com\fdx\cookbook\CookBook.java

```java
package com.fdx.cookbook;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.util.Log;


import java.io.File;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.UUID;

public class CookBook {
    private static CookBook ourInstance ;
    private Context mContext;
    private SQLiteDatabase mDatabase;
    private static final String TAG = "CB_Cookbook";

    public static CookBook get(Context context) {
        if (ourInstance==null){
            ourInstance= new CookBook(context);
        }
        return ourInstance;
    }

    private CookBook(Context context) {
          mContext=context.getApplicationContext();
          mDatabase=new RecipeBaseHelper(mContext)
                  .getWritableDatabase();
    }

    public Recipe getRecipe(UUID id) {
        RecipeCursorWrapper cursor=queryRecipes(
                RecipeDbSchema.RecipeTable.Cols.UUID+" =?",
                new String[] {id.toString()}
        );
        try{
            if (cursor.getCount()==0) {return null;}
            cursor.moveToFirst();
            return cursor.getRecipe();
        } finally {cursor.close();}
    }

    public void updateRecipe(Recipe r) {
        String uuidString=r.getId().toString();
        ContentValues values=getContentValues(r);
        mDatabase.update(RecipeDbSchema.RecipeTable.NAME, values,
                RecipeDbSchema.RecipeTable.Cols.UUID+" =?",
                new String[] {uuidString});
    }

    private RecipeCursorWrapper queryRecipes(String whereClause,
String[] whereArgs){
        Cursor cursor=mDatabase.query(
```

```
 57                 RecipeDbSchema.RecipeTable.NAME,
 58                 null, // selct all columns
 59                 whereClause,
 60                 whereArgs,
 61                 null, null, null
 62         );
 63         return new RecipeCursorWrapper(cursor);
 64     }
 65
 66     public void addRecipe(Recipe r){
 67         if (this.getRecipe(r.getId())==null) {
 68             ContentValues values=getContentValues(r);
 69             mDatabase.insert(RecipeDbSchema.RecipeTable.NAME, null,
    values);
 70         }
 71     }
 72
 73     public void removeRecipe(Recipe r){
 74         String uuidString=r.getId().toString();
 75         mDatabase.delete(RecipeDbSchema.RecipeTable.NAME,
 76                 RecipeDbSchema.RecipeTable.Cols.UUID+" =?",
 77                 new String[] {uuidString});
 78         return;
 79     }
 80
 81     public void markRecipeToDelete(Recipe r){
 82         r.setStatus(StatusRecipe.Deleted);
 83         updateRecipe(r);
 84         return;
 85     }
 86
 87     public List<Recipe> getRecipes(){
 88           List<Recipe> recipes=new ArrayList<>();
 89           RecipeCursorWrapper cursor=queryRecipes(null, null);
 90           try{
 91               cursor.moveToFirst();
 92               while (!cursor.isAfterLast()){
 93                   recipes.add(cursor.getRecipe());
 94                   cursor.moveToNext();
 95               }
 96           } finally { cursor.close();}
 97           return recipes;
 98     }
 99
100     public List<Recipe> getRecipesVisibles(){
101         List<Recipe> recipes=this.getRecipes();
102         List<Recipe> recipeloop=this.getRecipes();
103         for(Recipe r:recipeloop){
104             if (!r.IsVisible()) recipes.remove(r);
105         }
106         return recipes;
107     }
108
109     private static ContentValues getContentValues(Recipe recipe) {
110         ContentValues values=new ContentValues();
111         values.put(RecipeDbSchema.RecipeTable.Cols.UUID, recipe.getId
    ().toString());
```

```
112         values.put(RecipeDbSchema.RecipeTable.Cols.OWNER, recipe.
    getSerializedOwner());
113         values.put(RecipeDbSchema.RecipeTable.Cols.TITLE, recipe.
    getTitle());
114         values.put(RecipeDbSchema.RecipeTable.Cols.SOURCE, recipe.
    getSource());
115         values.put(RecipeDbSchema.RecipeTable.Cols.SOURCE_URL, recipe
    .getSource_url().toString());
116         values.put(RecipeDbSchema.RecipeTable.Cols.DATE, recipe.
    getDate().getTime());
117         if (recipe.getDatePhoto()!=null){
118         values.put(RecipeDbSchema.RecipeTable.Cols.DATE_PHOTO, recipe
    .getDatePhoto().getTime());}
119         values.put(RecipeDbSchema.RecipeTable.Cols.NBPERS, recipe.
    getNbPers());
120         for(int i=0;i<recipe.getNbStepMax();i++){
121             values.put(RecipeDbSchema.RecipeTable.Cols.STEP[i],
    recipe.getStep(i+1));
122         }
123         for(int i=0;i<recipe.getNbIngMax();i++){
124             values.put(RecipeDbSchema.RecipeTable.Cols.ING[i], recipe
    .getIngredient(i+1));
125         }
126         values.put(RecipeDbSchema.RecipeTable.Cols.SEASON, recipe.
    getSeason().name());
127         values.put(RecipeDbSchema.RecipeTable.Cols.DIFFICULTY, recipe
    .getDifficulty().name());
128         values.put(RecipeDbSchema.RecipeTable.Cols.TYPE, recipe.
    getType().name());
129         values.put(RecipeDbSchema.RecipeTable.Cols.COMMENTS, recipe.
    getSerializedComments());
130         values.put(RecipeDbSchema.RecipeTable.Cols.STATUS, recipe.
    getStatus().toString());
131         values.put(RecipeDbSchema.RecipeTable.Cols.NOTES, recipe.
    getSerializedNotes());
132         values.put(RecipeDbSchema.RecipeTable.Cols.MESSAGE, recipe.
    getMessage());
133         values.put(RecipeDbSchema.RecipeTable.Cols.MESSAGE_FROM,
    recipe.getSerializedFrom());
134         values.put(RecipeDbSchema.RecipeTable.Cols.TS_RECIPE, recipe.
    getTS(AsynCallFlag.NEWRECIPE));
135         values.put(RecipeDbSchema.RecipeTable.Cols.TS_PHOTO, recipe.
    getTS(AsynCallFlag.NEWPHOTO));
136         values.put(RecipeDbSchema.RecipeTable.Cols.TS_COMMENT, recipe
    .getTS(AsynCallFlag.NEWCOMMENT));
137         values.put(RecipeDbSchema.RecipeTable.Cols.TS_NOTE, recipe.
    getTS(AsynCallFlag.NEWRATING));
138         return values;
139     }
140
141     public File getPhotoFile(Recipe r){
142         if (r==null) {return null;}
143         File filesDir= mContext.getFilesDir();
144         return new File(filesDir, r.getPhotoFilename());
145     }
146
147     public Boolean deleteImage(Recipe r){
```

```
148         Boolean success=false;
149         File filesDir= mContext.getFilesDir();
150         File im=new File(filesDir, r.getPhotoFilename());
151         if (im.exists()){
152             im.delete();
153             success=true;
154         }
155         return success;
156     }
157 
158     public void clearCookBook(){
159         List<Recipe> recipes=new ArrayList<>();
160         recipes=getRecipes();
161         for (Recipe r:recipes){
162             deleteImage(r);
163             removeRecipe(r);
164         }
165     }
166 
167     public void fillCookBook(List<Recipe> recipes){
168         for(Recipe r:recipes){
169             addRecipe(r);
170         }
171     }
172 
173     public Boolean isThereMail(){
174         List<Recipe> recipes=this.getRecipes();
175         for(Recipe r:recipes){
176             if (r.IsMessage()) return true;
177         }
178         return false;
179     }
180 }
181 
```

```java
package com.fdx.cookbook;

import java.util.UUID;

public class ListMask {
    private boolean SeasonFiltered;
    private RecipeSeason SeasonState;
    private boolean DifficultyFiltered;
    private RecipeDifficulty DifficultyState;
    private boolean TypeFiltered;
    private RecipeType TypeState;
    private boolean UserFiltered;
    private User UserState;
    private boolean IsNoteSorted;
    private boolean IsSearched;
    private boolean IsTitleAlphaSorted;
    private String Query;

    public ListMask(User u){
        SeasonFiltered=false;
        DifficultyFiltered=false;
        TypeFiltered=false;
        UserFiltered=false;
        IsNoteSorted=false;
        IsSearched=false;
        IsTitleAlphaSorted=false;
        Query="";
        UserState=u;
        SeasonState=RecipeSeason.ALLYEAR;
        DifficultyState=RecipeDifficulty.UNDEFINED;
        TypeState=RecipeType.MAIN;
    }

    public void reset(){
        SeasonFiltered=false;
        DifficultyFiltered=false;
        TypeFiltered=false;
        UserFiltered=false;
        IsNoteSorted=false;
        IsSearched=false;
        IsTitleAlphaSorted=false;
        Query="";
        SeasonState=RecipeSeason.ALLYEAR;
        DifficultyState=RecipeDifficulty.UNDEFINED;
        TypeState=RecipeType.MAIN;
    }

    public Integer toggleSun(){
        Integer i;
        if (SeasonFiltered){
            SeasonFiltered=false;
            SeasonState=RecipeSeason.ALLYEAR;
            i=R.string.TSOFF;
        } else {
            SeasonFiltered=true;
            SeasonState=RecipeSeason.SUMMER;
            i=R.string.TSON;
```

```
58          }
59          return i;
60      }
61
62      public Integer toggleWinter(){
63          Integer i;
64          if (SeasonFiltered){
65              SeasonFiltered=false;
66              SeasonState=RecipeSeason.ALLYEAR;
67              i=R.string.TWOFF;
68          } else {
69              SeasonFiltered=true;
70              SeasonState=RecipeSeason.WINTER;
71              i=R.string.TWON;
72          }
73          return i;
74      }
75
76      public Integer toggleTitle(){
77          Integer i;
78          if (IsTitleAlphaSorted){
79              IsTitleAlphaSorted=false;
80              i=R.string.TTIOFF;
81          } else {
82              IsTitleAlphaSorted=true;
83              i=R.string.TTION;
84          }
85          return i;
86      }
87      public boolean isTitleSorted(){return IsTitleAlphaSorted;}
88
89      public Integer toggleUser(User u){
90          Integer i;
91          if (UserFiltered){
92              UserFiltered=false;
93              i=R.string.TROFF;
94          }else{
95              UserState=u;
96              UserFiltered=true;
97              i=R.string.TRON;
98          }
99          return i;
100     }
101
102     public Integer toggleSearch(String s){
103         Integer i;
104         if ((s==null)||(s.equals(""))) {
105             IsSearched=false;
106             Query="";
107             i=R.string.TSEOFF;
108         } else {
109             IsSearched=true;
110             Query=s;
111             i=R.string.TSEON;
112         }
113         return i;
114     }
```

```
115
116     public Integer toggleDifficulty(RecipeDifficulty rd){
117         Integer i;
118         if (DifficultyFiltered){
119             DifficultyFiltered=false;
120             DifficultyState=RecipeDifficulty.UNDEFINED;
121             i=R.string.TDOFF;
122         }else{
123             DifficultyFiltered=true;
124             DifficultyState=rd;
125             i=R.string.TDON;
126         }
127         return i;
128     }
129
130     public Integer toggleType(RecipeType rt){
131         Integer i;
132         if (TypeFiltered){
133             TypeFiltered=false;
134             TypeState=RecipeType.MAIN;
135             i=R.string.TTOFF;
136         }else{
137             TypeFiltered=true;
138             TypeState=rt;
139             i=R.string.TTON;
140         }
141         return i;
142     }
143
144     public Integer toggleSortNote(){
145         Integer i;
146         if (IsNoteSorted){
147             IsNoteSorted=false;
148             i=R.string.TNOFF;
149         }else{
150             IsNoteSorted=true;
151             i=R.string.TNON;
152         }
153         return i;
154     }
155
156     public boolean isNoteSorted(){return IsNoteSorted;}
157
158     public boolean isRecipeSelected(Recipe r){
159         boolean b=false;
160         if (!r.IsVisible()) return false;
161         if (SeasonFiltered) {
162             if (SeasonState==RecipeSeason.SUMMER) b=(r.getSeason()==
    RecipeSeason.WINTER);
163             if (SeasonState==RecipeSeason.WINTER) b=(r.getSeason()==
    RecipeSeason.SUMMER);
164             if (b) return false;
165         }
166         if (DifficultyFiltered){
167             if (r.getDifficulty()!=DifficultyState) return false;
168         }
169         if (TypeFiltered){
```

```
170             if (r.getType()!=TypeState) return false;
171         }
172         if (UserFiltered){
173             if (!r.getOwner().IsEqual(UserState)) return false;
174         }
175         if (IsSearched){
176             if (!r.containQuery(Query)) return false;
177         }
178         return true;
179     }
180 
181     public String convertToString(){
182         String s,sep=";";
183         s=(SeasonFiltered ? "Y":"N")+sep;   //0
184         s+=SeasonState.toString()+sep;      //1
185         s+=(DifficultyFiltered ? "Y":"N")+sep;  //2
186         s+=DifficultyState.toString()+sep;      //3
187         s+=(TypeFiltered ? "Y":"N")+sep;        //4
188         s+=TypeState.toString()+sep;            //5
189         s+=(IsNoteSorted ? "Y":"N")+sep;        //6
190         s+=(IsTitleAlphaSorted ? "Y":"N")+sep;  //7
191         s+=(IsSearched ? "Y":"N")+sep;          //8
192         s+=Query+sep;                           //9
193         s+=(UserFiltered ? "Y":"N")+sep;        //10
194         s+=UserState.getFamily()+sep;           //11
195         s+=UserState.getName()+sep;             //12
196         s+=UserState.getId().toString();        //13
197         return s;
198     }
199 
200     public void updateFromString(String s){
201         String sep=";";
202         String[] tokens = s.split(sep);
203         if (tokens.length<14) return;
204         SeasonFiltered=( tokens[0].equals("Y")? true:false);
205         SeasonState=RecipeSeason.valueOf(tokens[1]);
206         DifficultyFiltered=( tokens[2].equals("Y")? true:false);
207         DifficultyState=RecipeDifficulty.valueOf(tokens[3]);
208         TypeFiltered=( tokens[4].equals("Y")? true:false);
209         TypeState=RecipeType.valueOf(tokens[5]);
210         IsNoteSorted=( tokens[6].equals("Y")? true:false);
211         IsTitleAlphaSorted=( tokens[7].equals("Y")? true:false);
212         IsSearched=( tokens[8].equals("Y")? true:false);
213         Query=tokens[9];
214         UserFiltered=( tokens[10].equals("Y")? true:false);
215         UserState=new User(tokens[11], tokens[12]);
216         UserState.setId(UUID.fromString(tokens[13]));
217     }
218 }
219 
```

File - D:\4. Softwares\AndroidStudioProjects\CookBook\app\src\main\java\com\fdx\cookbook\MailCard.java

```java
package com.fdx.cookbook;

import java.util.UUID;

public class MailCard {
    private Boolean mIsReceived;
    private Boolean mIsRequest;
    private String mMessage;
    private User mUser; // from or to
    private Integer mStatus;
    private Integer SUBMITTED=1;
    private Integer ACCEPTED=2;
    private Integer REFUSED=3;
    private UUID mIdRecipe;
    private Integer mRequestId;
    private String mTitle;
    private String UUIDNULL="00000000-0000-0000-0000-000000000000";


    public MailCard(Recipe r){
        this();
        if ((r!=null)&&(r.IsMessage())){
            mUser=r.getUserFrom();
            mMessage=r.getMessage();
            mStatus=SUBMITTED;
            mIdRecipe=r.getId();
            mTitle=r.getTitle();
            mRequestId=0;
        }
    }

    public MailCard(){
        mUser=new User();
        mIsReceived=true;
        mIsRequest=false;
        mMessage="";
        mStatus=0;
        mIdRecipe=UUID.fromString(UUIDNULL);
        mTitle="";
    }

    public Boolean isReceived() {
        return mIsReceived;
    }
    public Boolean isRequest() {
        return mIsRequest;
    }

    public void setRequest() {
        mIsRequest = true;
        mIsReceived=false;
    }

    public UUID getRecipeId(){return mIdRecipe;}
    public void setIdRecipe(UUID idRecipe) {
        mIdRecipe = idRecipe;
    }
```

```
58
59      public Boolean isSubmitted() {return mStatus==SUBMITTED;}
60      public void setSubmitted(){mStatus=SUBMITTED;}
61      public String getTitle() {
62          return mTitle;
63      }
64
65      public void setTitle(String title) {
66          mTitle = title;
67      }
68
69      public String getMessage() {
70          return mMessage;
71      }
72
73      public void setMessage(String message) {
74          mMessage = message;
75      }
76
77      public User getUser() {
78          return mUser;
79      }
80
81      public void setUser(User user) {
82          mUser = user;
83      }
84
85      public void setRefused() {mStatus=REFUSED;}
86      public void setAccepted() {mStatus=ACCEPTED;}
87
88      public Integer getRequestId() {
89          return mRequestId;
90      }
91
92      public void setRequestId(Integer requestId) {
93          mRequestId = requestId;
94      }
95
96      public Integer getStatus() {
97          return mStatus;
98      }
99  }
100
```

```java
package com.fdx.cookbook;

public enum RecipeType {
    APERITIF,STARTER,MAIN,DESSERT,SIDE,OTHER;

        private static RecipeType[] list=RecipeType.values();

        public static RecipeType getType(int i){
            return list[i];
        }

        public static int getIndex(RecipeType r){
            for(int i=0;i<list.length;i++) {
                if (list[i].equals(r)){ return i;}
            }
            return 0;
        }
}

```

```java
package com.fdx.cookbook;

import android.content.Context;
import android.content.SharedPreferences;
import android.preference.PreferenceManager;

import java.util.UUID;

public class SessionInfo {
    private static SessionInfo ourInstance;
    private User mUser;
    private Context mContext;
    private Boolean mIsConnected;
    private Boolean mReqNewSession;
    private Boolean mIsRecipeRequest;
    private String CB_FAMILY="family";
    private String CB_NAME="name";
    private String CB_ID="iduser";
    private String NOT_FOUND="Not found";
    private String mMaskSerialized;
    private String mDevice;
    private static final String TAG = "DebugSessionInfo";
    private static String URLPATH="http://82.66.37.73:8085/cb/";
    public static int CONNECT_TIMEOUT = 10000;
    public static int READ_TIMEOUT = 10000;

    public static SessionInfo get(Context context) {
        if (ourInstance==null){
            ourInstance= new SessionInfo(context);
        }
        return ourInstance;
    }

    private SessionInfo(Context context) {
        mContext=context.getApplicationContext();
        mIsConnected=false;
        mReqNewSession=false;
        mIsRecipeRequest=false;
        mMaskSerialized="";
        mDevice="Device " + android.os.Build.DEVICE+ " model "
                +android.os.Build.MODEL + " ("+ android.os.Build.
PRODUCT + ")";
        String sharedPref;
        sharedPref= PreferenceManager.getDefaultSharedPreferences(
context)
                .getString(CB_ID, NOT_FOUND);
        if (sharedPref.equals(NOT_FOUND)){
            mUser = new User(NOT_FOUND, NOT_FOUND);
        } else {
            mUser = new User(PreferenceManager.
getDefaultSharedPreferences(context)
                    .getString(CB_FAMILY, NOT_FOUND),
PreferenceManager.getDefaultSharedPreferences(context)
                    .getString(CB_NAME, NOT_FOUND));
            mUser.setId(UUID.fromString(sharedPref));
        }
    }
```

```
 54
 55
 56     public User getUser() {
 57         return mUser;
 58     }
 59
 60     public void setStoredUser(User user){
 61         PreferenceManager.getDefaultSharedPreferences(mContext)
 62                 .edit()
 63                 .putString(CB_FAMILY, user.getFamily())
 64                 .putString(CB_NAME, user.getName())
 65                 .putString(CB_ID, user.getId().toString())
 66                 .apply();
 67         mUser=user;
 68     }
 69     public void clearStoredUser(){
 70         SharedPreferences settings = mContext.getSharedPreferences("
    PreferencesName", Context.MODE_PRIVATE);
 71         settings.edit().remove("CB_FAMILY").commit();
 72         settings.edit().remove("CB_NAME").commit();
 73         settings.edit().remove("CB_ID").commit();
 74     }
 75
 76     public Boolean IsEmpty(){
 77         if (mUser.getName().equals(NOT_FOUND)){return true;}
 78         return false;
 79     }
 80
 81     public Boolean IsReqNewSession() {
 82         return mReqNewSession;
 83     }
 84
 85     public void setReqNewSession(Boolean reqNewSession) {
 86         mReqNewSession = reqNewSession;
 87     }
 88
 89     public void setConnection(Boolean b){
 90         mIsConnected=b;
 91     }
 92     public Boolean IsConnected(){
 93         return mIsConnected;
 94     }
 95
 96     public Context getContext(){return mContext;}
 97
 98     public String getURLPath(){return URLPATH;}
 99     public int getConnectTimeout(){return CONNECT_TIMEOUT;}
100     public int getReadTimeout(){return READ_TIMEOUT;}
101     public String getListMask(){return mMaskSerialized;}
102     public void setListMask(String s){mMaskSerialized=s;}
103     public Boolean IsRecipeRequest(){return mIsRecipeRequest;}
104     public void setIsRecipeRequest(Boolean b){mIsRecipeRequest=b;}
105
106     public String getDevice() {
107         return mDevice;
108     }
109 }
```

File - D:\4. Softwares\AndroidStudioProjects\CookBook\app\src\main\java\com\fdx\cookbook\NetworkUtils.java

```java
package com.fdx.cookbook;

import android.content.Context;
import android.graphics.Bitmap;
import android.util.Log;

import org.json.JSONArray;
import org.json.JSONObject;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLEncoder;
import java.text.DecimalFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.UUID;

public class NetworkUtils {
    private SessionInfo mSession;
    private static final String TAG = "CB_NetworkUtils";
    private static final String MYSQLDATEFORMAT="yyyy-MM-dd HH:mm:ss";

    public NetworkUtils(Context c){
        mSession=SessionInfo.get(c);
    }

    public String sendPostRequestJson(String requestURL,
                                      HashMap<String, String>
postDataParams) {

        URL url;
        try {
            url = new URL(requestURL);
            HttpURLConnection conn = (HttpURLConnection) url.
openConnection();
            conn.setReadTimeout(mSession.getReadTimeout());
            conn.setConnectTimeout(mSession.getConnectTimeout());
            conn.setRequestMethod("POST");
            conn.setDoInput(true);
            conn.setDoOutput(true);
            OutputStream os = conn.getOutputStream();
            BufferedWriter writer = new BufferedWriter(
                    new OutputStreamWriter(os, "UTF-8"));
```

```
 56              writer.write(this.getPostDataString(postDataParams));
 57              writer.flush();
 58              writer.close();
 59              os.close();
 60              StringBuilder sb = new StringBuilder();
 61              BufferedReader bufferedReader = new BufferedReader(new
    InputStreamReader(conn.getInputStream()));
 62              String json;
 63              while ((json = bufferedReader.readLine()) != null) {
 64                  sb.append(json + "\n");
 65              }
 66              return sb.toString().trim();
 67          } catch (Exception e) {
 68              deBugShow( "SendPostRequest error>"+e);
 69              return null;
 70          }
 71      }
 72      private String getPostDataString(HashMap<String, String> params
    ) {
 73          StringBuilder result = new StringBuilder();
 74          boolean first = true;
 75          for (Map.Entry<String, String> entry : params.entrySet()) {
 76              if (first)
 77                  first = false;
 78              else
 79                  result.append("&");
 80              try {result.append(URLEncoder.encode(entry.getKey(), "UTF
    -8"));
 81              result.append("=");
 82              result.append(URLEncoder.encode(entry.getValue(), "UTF-8"
    ));
 83              } catch (Exception e){
 84                  deBugShow( "Pb with >"+entry.getKey()+":"+entry.
    getValue()+"<");
 85              }
 86          }
 87          return result.toString();
 88      }
 89      public List<Comment> parseCommentsOfRecipe(String json){
 90          Comment c;
 91          List<Comment> cs=new ArrayList<>();
 92          if ((json==null)||(json.equals(""))) return null;
 93          try {
 94              JSONArray jarr1=new JSONArray(json);
 95              for (int i=0; i<jarr1.length(); i++){
 96                  JSONObject obj = jarr1.getJSONObject(i);
 97                  c=parseObjectComment(obj);
 98                  if (c!=null) cs.add(c);
 99              }
100          } catch (Exception e){
101              deBugShow( "Failure in parsing JSON Array Comments "+e);
102              return null;
103          }
104          return cs;
105      }
106
107      public Comment parseObjectComment(JSONObject obj){
```

```
108         try {
109         UUID uuid=UUID.fromString(obj.getString("id_user"));
110         User u=new User(obj.getString("family"), obj.getString("name"
    ) );
111         u.setId(uuid);
112         String s=obj.getString("date_comment");
113         Date date=new SimpleDateFormat(MYSQLDATEFORMAT).parse(s);
114         return new Comment(obj.getString("comment"),u,date);}
115         catch (Exception e){
116             deBugShow("Failure in parsing JSONObject Comments "+e);
117             return null;
118         }
119     }
120     public List<Note> parseNotesOfRecipe(String json){
121         Note n;
122         List<Note> ns=new ArrayList<>();
123         if ((json==null)||(json.equals(""))) return null;
124         try {
125             JSONArray jarr1=new JSONArray(json);
126             for (int i=0; i<jarr1.length(); i++){
127                 JSONObject obj = jarr1.getJSONObject(i);
128                 n=parseObjectNote(obj);
129                 if (n!=null) ns.add(n);
130             }
131         } catch (Exception e){
132             deBugShow("Failure in parsing JSON Array Comments "+e);
133             return null;
134         }
135         return ns;
136     }
137
138     public Note parseObjectNote(JSONObject obj){
139         try {
140             UUID uuid=UUID.fromString(obj.getString("id_user"));
141             User u=new User(obj.getString("family"), obj.getString("
    name") );
142             u.setId(uuid);
143             String s=obj.getString("date_note");
144             Date date=new SimpleDateFormat(MYSQLDATEFORMAT).parse(s);
145             return new Note(obj.getInt("note"),u,date);}
146         catch (Exception e){
147             deBugShow("Failure in parsing JSONObject Note : "+e);
148             return null;
149         }
150     }
151     public Recipe parseObjectRecipeStamp(JSONObject obj){
152         try {
153             //---------------- uuid and users
154             UUID uuid = UUID.fromString(obj.getString("id_recipe"));
155             Recipe r = new Recipe(uuid);
156             uuid = UUID.fromString(obj.getString("id_owner"));
157             User u = new User(uuid);
158             r.setOwner(u);
159             //dates
160             Date date;
161             String s1 = obj.getString("lastupdate_recipe");
162             if (s1==null) return null;
```

```
163             if ((!s1.equals("null"))&&(s1.length()>5)) {
164                 date = new SimpleDateFormat(MYSQLDATEFORMAT).parse(s1
    );
165                 r.setDate(date);
166             } else {return null;}
167             s1 = obj.getString("lastupdate_photo");
168             if ((!s1.equals("null"))&&(s1.length()>5)) {
169                 date = new SimpleDateFormat(MYSQLDATEFORMAT).parse(s1
    );
170                 r.setDatePhoto(date);
171             }
172             s1=obj.getString("message");
173             if ((s1==null)||(s1.equals("null"))) s1="";
174             r.setMessage(s1);
175             r.setStatus(StatusRecipe.valueOf(obj.getString("status"
    )));
176             s1 = obj.getString("id_from");
177             if ((!s1.equals("null"))&&(s1.length()>5)) {
178                 uuid = UUID.fromString(s1);
179                 u = new User(uuid);
180                 r.setOwner(u);
181             }
182             return r;
183         }
184         catch (Exception e){
185             deBugShow( "Failure in parsing JSONObject Recette Stamp
     : "+e);
186             return null;
187         }
188     }
189
190     public List<Recipe> parseStampsTiers(String json){
191         Recipe r;
192         List<Recipe> rs=new ArrayList<>();
193         if ((json==null)||(json.equals(""))) return null;
194         try {
195             JSONArray jarr1=new JSONArray(json);
196             for (int i=0; i<jarr1.length(); i++){
197                 JSONObject obj = jarr1.getJSONObject(i);
198                 r=parseObjectRecipeStamp(obj);
199                 if (r!=null) rs.add(r);
200             }
201         } catch (Exception e){
202             deBugShow( "Failure in parsing JSON Array Comments "+e);
203             return null;
204         }
205         return rs;
206     }
207
208     public boolean parseObjectRecipe(Recipe r, JSONObject obj,
    boolean withphoto, boolean full){
209         try {
210             UUID uuid;
211             String s1, s2;
212             User u;
213             if (full) {
214                 uuid = UUID.fromString(obj.getString("id_owner"));
```

```
215                 s1 = obj.getString("owner_family");
216                 s2 = obj.getString("owner_name");
217                 u = new User(s1, s2);
218                 u.setId(uuid);
219                 r.setOwner(u);
220                 s1=obj.getString("message");
221                 if ((s1==null)||(s1.equals("null"))) s1="";
222                 r.setMessage(s1);
223                 r.setStatus(StatusRecipe.valueOf(obj.getString("
    status")));
224                 s1 = obj.getString("id_from");
225                 if ((!s1.equals("null"))&&(s1.length()>5)&&(s1!=null
    )) {
226                     uuid = UUID.fromString(s1);
227                     s1 = obj.getString("from_family");
228                     s2 = obj.getString("from_name");
229                     u = new User(s1, s2);
230                     u.setId(uuid);
231                     r.setUserFrom(u);
232                 }
233             }
234             //------------ dates
235             Date date;
236             s1 = obj.getString("lastupdate_recipe");
237             if ((!s1.equals("null"))&&(s1.length()>5)&&(s1!=null)) {
238                 date = new SimpleDateFormat(MYSQLDATEFORMAT).parse(s1
    );
239                 r.setDate(date);
240             } else {return false;}
241             s1 = obj.getString("lastupdate_photo");
242             if ((!s1.equals("null"))&&(s1.length()>5)&&(s1!=null)) {
243                 date = new SimpleDateFormat(MYSQLDATEFORMAT).parse(s1
    );
244                 r.setDatePhoto(date);
245             }
246             //----------------- titre, source x2, nbpers
247             s1 = obj.getString("title");
248             URL url;
249             if (s1==null) return false;
250             if (s1.equals("null")) return false;
251             r.setTitle(s1);
252             r.setSource(obj.getString("source"));
253             s1 = obj.getString("source_url");
254             if (!s1.equals("null")&&(s1.length()>5)&&(s1!=null)) {
255                 try {
256                     url = new URL(s1);
257                     r.setSource_url(url);
258                 } catch (MalformedURLException e) {
259                     deBugShow("getURL from cookbook download failed
    in parse recipe "+e);
260                 }
261             }
262             int nbp=obj.getInt("nb_pers");
263             if (nbp==0) return false;
264             r.setNbPers(nbp);
265             //---------------- Etapes and ing
266             DecimalFormat formatter = new DecimalFormat("00");
```

```
267             for (int i = 0; i < r.getNbStepMax(); i++) {
268                 s1 = "etape" + formatter.format(i + 1);
269                 s2=obj.getString(s1);
270                 if (!s2.equals("null")) {
271                     r.setStep(i + 1, s2);}
272             }
273             for (int i = 0; i < r.getNbIngMax(); i++) {
274                 s1 = "ing" + formatter.format(i + 1);
275                 s2=obj.getString(s1);
276                 if (!s2.equals("null")) {
277                     r.setIngredient(i + 1, s2);}
278             }
279             //----------------- enum
280             r.setSeason(RecipeSeason.valueOf(obj.getString("season"
    )));
281             r.setDifficulty(RecipeDifficulty.valueOf(obj.getString("
    difficulty")));
282             r.setType(RecipeType.valueOf(obj.getString("type")));
283             // --------------- photo
284             CookBook cb = CookBook.get(mSession.getContext());
285             File f=cb.getPhotoFile(r);
286             if (withphoto) {
287                 s1 = obj.getString("photo");
288                 if ((!s1.equals("null"))&&(!s1.equals(""))&&(s1!=null
    )) {
289                     Bitmap bm=PictureUtils.getBitmapFromString(s1);
290                     if (f.exists()){
291                         deBugShow( "file  " + f.toString()+" ecrasée
    dans parsing recipe !");
292                         f.delete();
293                     }
294                     try {
295                         if(!f.createNewFile()) {
296                             deBugShow( "Error in creating file  "+f.
    toString());
297                         }
298                     } catch (IOException e) {
299                         deBugShow( "Error in creating file "+f.
    toString()+":" +e);
300                     }
301                     try {
302                         FileOutputStream out = new FileOutputStream(f
    );
303                         bm.compress(Bitmap.CompressFormat.JPEG, 100,
    out);
304                         out.flush();
305                         out.close();
306                     } catch (Exception e) {
307                         deBugShow("Storing bitmap error  " + e);
308                     }
309
310                 }
311             }
312
313             return true;
314         }
315         catch (Exception e){
```

```
316             deBugShow( "Failure in parsing JSONObject Recette : "+e);
317             return false;
318         }
319     }
320
321     public boolean saveBmpInRecipe(Bitmap bmp, Recipe r){
322         CookBook cb = CookBook.get(mSession.getContext());
323         File f=cb.getPhotoFile(r);
324         if (f.exists()) f.delete();
325         try {
326             if(!f.createNewFile()) {
327                 //fdx Log.d(TAG, "Error in creating file  "+f.
    toString());
328             }
329             FileOutputStream out = new FileOutputStream(f);
330             bmp.compress(Bitmap.CompressFormat.JPEG, 100, out);
331             out.flush();
332             out.close();
333             return true;
334         } catch (Exception e) {
335             deBugShow( "Storing bitmap error  " + e);
336             return false;
337         }
338     }
339
340     public List<Recipe> parseRecipesOfCommunity(String json, boolean
    withphoto){
341         Recipe r;
342         List<Recipe> rs=new ArrayList<>();
343         if ((json==null)||(json.equals(""))) return null;
344         try {
345             JSONArray jarr1=new JSONArray(json);
346             for (int i=0; i<jarr1.length(); i++){
347                 JSONObject obj = jarr1.getJSONObject(i);
348                 r=new Recipe();
349                 if (!parseObjectRecipeShort(r,obj, withphoto)) {
350                     deBugShow("Error in parsing");
351                     return null;}
352                 if (r!=null) rs.add(r);
353             }
354         } catch (Exception e){
355             deBugShow("Failure in parsing JSON Array Recipes for
    community "+e);
356             return null;
357         }
358         return rs;
359     }
360
361     public Boolean parseObjectRecipeShort(Recipe r,JSONObject obj,
    boolean withphoto){
362         try {
363             UUID uuid;
364             String s1, s2;
365             User u;
366             uuid = UUID.fromString(obj.getString("id_recipe"));
367             r.setId(uuid);
368             uuid = UUID.fromString(obj.getString("id_user"));
```

```
369             s1 = obj.getString("family");
370             s2 = obj.getString("name");
371             u = new User(s1, s2);
372             u.setId(uuid);
373             r.setOwner(u);
374             //------------ dates
375             Date date;
376             s1 = obj.getString("lastupdate_recipe");
377             if ((!s1.equals("null"))&&(s1.length()>5)&&(s1!=null)) {
378                 date = new SimpleDateFormat(MYSQLDATEFORMAT).parse(s1
    );
379                 r.setDate(date);
380             } else {return false;}
381             //----------------- titre, source x2, nbpers
382             s1 = obj.getString("title");
383             r.setTitle(s1);
384             //----------------- enum
385             r.setSeason(RecipeSeason.valueOf(obj.getString("season"
    )));
386             r.setDifficulty(RecipeDifficulty.valueOf(obj.getString("
    difficulty")));
387             r.setType(RecipeType.valueOf(obj.getString("type")));
388             // --------------- photo
389             if (withphoto) {
390                 s1 = obj.getString("photo");
391                 if ((s1!=null)&&(!s1.equals("")))
392                     r.setImage(PictureUtils.getBitmapFromString(s1));
393                 }
394             return true;
395         }
396         catch (Exception e){
397             deBugShow( "Failure in parsing JSONObject Recette : "+e);
398             return false;
399         }
400     }
401
402     public MailCard parseObjectRequest(JSONObject obj){
403         MailCard mc=new MailCard();
404         mc.setRequest();
405         String s1, s2;
406         User u;
407         Integer n;
408         try {
409             UUID uuid=UUID.fromString(obj.getString("id_recipe"));
410             mc.setIdRecipe(uuid);
411             uuid=UUID.fromString(obj.getString("id_from"));
412             s1 = obj.getString("family");
413             s2 = obj.getString("name");
414             u = new User(s1, s2);
415             u.setId(uuid);
416             mc.setUser(u);
417             s1 = obj.getString("message");
418             mc.setMessage(s1);
419             n=obj.getInt("pknum");
420             mc.setRequestId(n);
421             s1 = obj.getString("title");
422             mc.setTitle(s1);
```

```
423             s1 = obj.getString("status");
424             if (s1.equals("ISSUED")) mc.setSubmitted();
425             if (s1.equals("DENIED")) mc.setRefused();
426             if (s1.equals("ACCEPTED")) mc.setAccepted();
427             return mc;
428         }
429         catch (Exception e){
430             deBugShow( "Failure in parsing JSONObject Request : "+e);
431             return null;
432         }
433     }
434 
435     public Boolean test204() {
436         String PHP204 = "return204.php";
437         try {
438             URL url = new URL(mSession.getURLPath() + PHP204);
439             HttpURLConnection conn = (HttpURLConnection) url.
    openConnection();
440             conn.setConnectTimeout(mSession.getConnectTimeout());
441             conn.setReadTimeout(mSession.getReadTimeout());
442             conn.setRequestMethod("HEAD");
443             InputStream in = conn.getInputStream();
444             int status = conn.getResponseCode();
445             in.close();
446             conn.disconnect();
447             return (status == HttpURLConnection.HTTP_NO_CONTENT);
448         } catch (Exception e) {
449             deBugShow("Test 204 : " + e);
450             return false;
451         }
452     }
453 
454     private void deBugShow(String s){
455         //Log.d(TAG, s);
456     }
457 
458 }
459 
```

File - D:\4. Softwares\AndroidStudioProjects\CookBook\app\src\main\java\com\fdx\cookbook\PictureUtils.java

```java
package com.fdx.cookbook;

import android.app.Activity;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Point;
import android.util.Base64;


import java.io.ByteArrayOutputStream;

public class PictureUtils {
    public static Bitmap getScaledBitmap(String path, int destWidth, int destHeight) {
        BitmapFactory.Options options=new BitmapFactory.Options();
        options.inJustDecodeBounds=true;
        BitmapFactory.decodeFile(path, options);
        float srcWidth=options.outWidth;
        float srcHeight=options.outHeight;
        int inSampleSize=1;
        if (srcHeight>destHeight || srcWidth>destWidth) {
            float heightScale=srcHeight / destHeight;
            float widthScale=srcWidth/destWidth;
            inSampleSize=Math.round(heightScale > widthScale ? heightScale : widthScale);
        }
        options=new BitmapFactory.Options();
        options.inSampleSize=inSampleSize;
        return BitmapFactory.decodeFile(path, options);
    }

    public static Bitmap getScaledBitmap(Bitmap bmpin, int w) {
        Integer h=(Integer)(bmpin.getHeight()*w/bmpin.getWidth());
        return bmpin.createScaledBitmap(bmpin, w, h, true);
    }

    public static Bitmap getBitmap(String path) {
        BitmapFactory.Options options=new BitmapFactory.Options();
        options.inJustDecodeBounds=true;
        BitmapFactory.decodeFile(path, options);
        int inSampleSize=1;
        options=new BitmapFactory.Options();
        options.inSampleSize=inSampleSize;
        return BitmapFactory.decodeFile(path, options);
    }
    @SuppressWarnings("deprecation")
    public static Bitmap getScaledBitmap(String path, Activity activity){
        Point size=new Point();
        activity.getWindowManager().getDefaultDisplay().getSize(size);
        return getScaledBitmap(path, size.x, size.y);
    }
    public static Bitmap getScaledBitmap(String path, Activity activity, int width){
        Point size=new Point();
        activity.getWindowManager().getDefaultDisplay().getSize(size);
        int height=(Integer) Math.round(width*size.y/size.x);
```

```
54          return getScaledBitmap(path, width, height);
55      }
56
57      public static String getStringImage(Bitmap bmp){
58          ByteArrayOutputStream baos = new ByteArrayOutputStream();
59          bmp.compress(Bitmap.CompressFormat.JPEG, 100, baos);
60          byte[] imageBytes = baos.toByteArray();
61          String encodedImage = Base64.encodeToString(imageBytes,
    Base64.DEFAULT);
62          return encodedImage;
63      }
64
65      public static Bitmap getBitmapFromString(String s){
66          byte[] decodedString = Base64.decode(s, Base64.DEFAULT);
67          Bitmap decodedByte = BitmapFactory.decodeByteArray(
    decodedString, 0, decodedString.length);
68          return decodedByte;
69      }
70  }
71
```

File - D:\4. Softwares\AndroidStudioProjects\CookBook\app\src\main\java\com\fdx\cookbook\RecipeSeason.java

```java
package com.fdx.cookbook;

public enum RecipeSeason {
    WINTER,SUMMER,ALLYEAR;

    private static RecipeSeason[] list=RecipeSeason.values();

    public static RecipeSeason getSeason(int i){
        return list[i];
    }

    public static int getIndex(RecipeSeason r){
        for(int i=0;i<list.length;i++) {
            if (list[i].equals(r)){ return i;}
        }
        return 0;
    }
}

```

```java
package com.fdx.cookbook;

import android.app.Activity;
import android.content.Context;
import android.graphics.Bitmap;
import android.os.AsyncTask;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.widget.Toast;

import org.json.JSONArray;
import org.json.JSONObject;

import java.io.File;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.URL;
import java.text.DateFormat;
import java.text.DecimalFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.UUID;

enum AsynCallFlag { NEWRECIPE, NEWPHOTO, NEWCOMMENT, NEWRATING, GLOBALSYNC, DELETERECIPE}

class AsyncCallClass extends AsyncTask<Void, Integer, Boolean> {
    private static final String TAG = "CB_AsynCalls";
    private static final String PHP204 = "return204.php";
    private static final String PHPUPDATECREATE = "updateorcreaterecipe.php";
    private static final String PHPUPUPLOADPHOTO = "uploadphotointorecipe.php";
    private static final String PHPDELETERECIPE = "deleterecipe.php";
    private static final String PHPGETCOMMENTSOFRECIPE = "getcommentsofrecipe.php";
    private static final String PHPADDCOMMENTTORECIPE = "addcommentwithdate.php";
    private static final String PHPADDNOTETORECIPE = "addnotewithdate.php";
    private static final String PHPGETNOTESOFRECIPE = "getnotesofrecipe.php";
    private static final String PHPGETSTAMPSTIERS = "getrecipestampstiers.php";
    private static final String PHPGETRECIPEFROMCB = "getrecipefromcb.php";
    private static final String PHPGETRECIPEFROMCBWITHPHOTO = "getrecipefromcbwithphoto.php";
    private static final String PHPACCEPTRECIPE = "acceptrecipe.php";
    private static final String PHPCHECKREQUESTS = "checkrequests.php";
    private static final String MYSQLDATEFORMAT="yyyy-MM-dd HH:mm:ss";
    private static Context mContext;
```

```
47      private SessionInfo mSession;
48      private NetworkUtils mNetUtils;
49      private CookBook mCookbook;
50
51      public AsyncCallClass(Context context){
52          mContext=context;
53          mCookbook = CookBook.get(mContext);
54          mSession = SessionInfo.get(mContext);
55          mNetUtils = new NetworkUtils(mContext);
56      }
57
58      @Override
59      protected Boolean doInBackground(Void... params) {
60          Boolean isChanged=false;
61          if (!test204()) {
62              return isChanged;
63          }
64          deBugShow("----------- Start Sync-------------");
65          if (syncTiersRecipe(mSession.getUser())) isChanged=true;
66          List<Recipe> mrecipes=mCookbook.getRecipes();
67          for(Recipe r:mrecipes){
68              if (r.getOwnerIdString().equals(mSession.getUser().getId
    ().toString())) {
69                  if (r.getTS(AsynCallFlag.NEWRECIPE) == 1) {
70                      if (uploadRecipe(r)) {
71                          deBugShow("Recette " + r.getTitle() + " mise
    à jour");
72                          r.updateTS(AsynCallFlag.NEWRECIPE, false);
73                          mCookbook.updateRecipe(r);
74                          isChanged=true;
75                      } else {
76                          deBugShow( "Recette " + r.getTitle() + " non
    mise à jour");
77                      }
78                  }
79                  if (r.getTS(AsynCallFlag.NEWPHOTO) == 1) {
80                      if (uploadPhoto(r)) {
81                          deBugShow( "Recette Photo " + r.getTitle() +
    " mise à jour");
82                          r.updateTS(AsynCallFlag.NEWPHOTO, false);
83                          mCookbook.updateRecipe(r);
84                          isChanged=true;
85                      } else {
86                          deBugShow( "Recette Photo " + r.getTitle() +
    " non mise à jour");
87                      }
88                  }
89              }
90              if (syncCommentsRecipe(r)) {
91                  deBugShow( "Comments de " + r.getTitle()+" updaté");
92                  isChanged=true;
93                  r.updateTS(AsynCallFlag.NEWCOMMENT,false);
94                  mCookbook.updateRecipe(r);
95              }
96              if (syncNotesRecipe(r)) {
97                  deBugShow( "Notes de " + r.getTitle()+" updaté");
98                  isChanged=true;
```

```
 99                 r.updateTS(AsynCallFlag.NEWRATING,false);
100                 mCookbook.updateRecipe(r);
101             }
102             if (r.IsMarkedDeleted()){
103                 if (deleteRecipeFromCB(r)) {
104                     deBugShow( "Recette effacée : " + r.getTitle());
105                     mCookbook.removeRecipe(r);
106                     mCookbook.deleteImage(r);
107                     isChanged=true;
108                 } else{
109                     deBugShow("Recette " + r.getTitle()+" non effacée
    ");
110                 }
111             }
112         }
113         mSession.setIsRecipeRequest(checkRecipeRequest());
114         return isChanged;
115     }
116 
117     @Override
118     protected void onPreExecute() {
119         super.onPreExecute();
120     }
121 
122     @Override
123     protected void onPostExecute(Boolean isChanged) {
124         super.onPostExecute(isChanged);
125         if (isChanged)
126         Toast.makeText(mContext, mContext.getString(R.string.P1_sync
    ), Toast.LENGTH_SHORT).show();
127     }
128 
129     private void deBugShow(String s){
130         //Log.d(TAG, s);
131     }
132 
133     private Boolean test204() {
134         try {
135             URL url = new URL(mSession.getURLPath() + PHP204);
136             HttpURLConnection conn = (HttpURLConnection) url.
    openConnection();
137             conn.setConnectTimeout(mSession.getConnectTimeout());
138             conn.setReadTimeout(mSession.getReadTimeout());
139             conn.setRequestMethod("HEAD");
140             InputStream in = conn.getInputStream();
141             int status = conn.getResponseCode();
142             in.close();
143             conn.disconnect();
144             return (status == HttpURLConnection.HTTP_NO_CONTENT);
145         } catch (Exception e) {
146             deBugShow( "Test 204 : " + e);
147             return false;
148         }
149     }
150 
151     private Boolean uploadRecipe(Recipe r) {
152         HashMap<String, String> data = new HashMap<>();
```

```
153         String s1;
154         DateFormat dateFormat = new SimpleDateFormat(MYSQLDATEFORMAT
    );
155         data.put("idrecipe", r.getId().toString());
156         data.put("iduser", r.getOwner().getId().toString());
157         data.put("title", r.getTitle());
158         data.put("source", r.getSource());
159         data.put("sourceurl", r.getSource_url_name());
160         DecimalFormat formatter = new DecimalFormat("00");
161         data.put("nbpers", formatter.format(r.getNbPers()));
162         s1 = dateFormat.format(r.getDate());
163         data.put("date", s1);
164         for (int i = 0; i < r.getNbStepMax(); i++) {
165             s1 = "etape" + formatter.format(i + 1);
166             data.put(s1, r.getStep(i + 1));
167         }
168         for (int i = 0; i < r.getNbIngMax(); i++) {
169             s1 = "ing" + formatter.format(i + 1);
170             data.put(s1, r.getIngredient(i + 1));
171         }
172         data.put("season", r.getSeason().toString());
173         data.put("difficulty", r.getDifficulty().toString());
174         data.put("type", r.getType().toString());
175         String result = mNetUtils.sendPostRequestJson(mSession.
    getURLPath() + PHPUPDATECREATE, data);
176         if (result==null) return false;
177         if (!result.trim().equals("1")) {
178             deBugShow( "Retour de "+PHPUPDATECREATE+" =>"+result+"<"
    );
179         return false;}
180         return true;
181     }
182 
183     private Boolean uploadPhoto(Recipe r) {
184         String s1;
185         File file = mCookbook.getPhotoFile(r);
186         Bitmap bitmap = PictureUtils.getBitmap(file.getPath());
187         if (bitmap == null) {
188            deBugShow( "Pas de bitmap pour " + r.getTitle());
189             return false;
190         }
191         String uploadImage = PictureUtils.getStringImage(bitmap);
192         HashMap<String, String> data = new HashMap<>();
193         data.put("idrecipe", r.getId().toString());
194         if (uploadImage==null) return false;
195         data.put("image", uploadImage);
196         DateFormat dateFormat = new SimpleDateFormat(MYSQLDATEFORMAT
    );
197         s1 = dateFormat.format(r.getDatePhoto());
198         data.put("date", s1);
199         String result = mNetUtils.sendPostRequestJson(mSession.
    getURLPath() + PHPUPUPLOADPHOTO, data);
200         if (result==null) return false;
201         if (!result.trim().equals("1")) {
202             deBugShow( "Retour de "+ PHPUPUPLOADPHOTO+" = "+result);
203             return false;}
204         return true;
```

```
205     }
206     private Boolean deleteRecipeFromCB(Recipe r) {
207         HashMap<String, String> data = new HashMap<>();
208         data.put("idrecipe", r.getId().toString());
209         data.put("iduser", mSession.getUser().getId().toString());
210         String result = mNetUtils.sendPostRequestJson(mSession.
    getURLPath() + PHPDELETERECIPE, data);
211         if (result==null) return false;
212         if (!result.trim().equals("1")) {
213             deBugShow( "Retour de "+ PHPDELETERECIPE+" = "+result);
214             return false;}
215         return true;
216     }
217
218     private Boolean syncCommentsRecipe(Recipe r) {
219         boolean ret=false;
220         HashMap<String, String> data = new HashMap<>();
221         data.put("idrecipe", r.getId().toString());
222         String result = mNetUtils.sendPostRequestJson(mSession.
    getURLPath() + PHPGETCOMMENTSOFRECIPE, data);
223         List<Comment> downloadedComments=mNetUtils.
    parseCommentsOfRecipe(result);
224         if (downloadedComments==null) downloadedComments=new
    ArrayList<>();
225         List<Comment> recipeComments=r.getComments();
226         if (recipeComments==null) recipeComments=new ArrayList<>();
227         //----------- boucle local
228         List<Comment> c1oc= new ArrayList<>();
229         Comment c,ci;
230         if (!recipeComments.isEmpty()){
231             for (int i = 0; i < recipeComments.size(); i++){
232                 ci=recipeComments.get(i);
233                 c=new Comment(ci.getTxt(),ci.getUser(),ci.getDate());
234                 if (downloadedComments.isEmpty()) {c1oc.add(c);} else
     {
235                 if (downloadedComments.indexOf(c)==-1) c1oc.add(c);}
236             }
237         }
238         //----------- boucle serveur
239         List<Comment> cser= new ArrayList<>();
240         if (!downloadedComments.isEmpty()){
241             for (int i = 0; i < downloadedComments.size(); i++){
242                 ci=downloadedComments.get(i);
243                 c=new Comment(ci.getTxt(),ci.getUser(),ci.getDate());
244                 if (recipeComments.isEmpty()) {cser.add(c);} else {
245                 if (recipeComments.indexOf(c)==-1) cser.add(c);}
246             }
247         }
248         if (!cser.isEmpty()){
249             for (int i = 0; i < cser.size(); i++){
250                 r.addComment(cser.get(i));
251             }
252             if (cser.size()>0) {
253                 mCookbook.updateRecipe(r);
254                 ret=true;
255             }
256         }
```

```
257         if (!c1oc.isEmpty()) {
258             if (!uploadComments(r,c1oc)) {ret=false;}
259             else {ret=true;}
260         }
261         return ret;
262     }
263
264     private Boolean uploadComments(Recipe r, List<Comment> cs){
265         if (cs==null) return false;
266         if (cs.size()==0) return true;
267         HashMap<String, String> data = new HashMap<>();
268         DateFormat dateFormat = new SimpleDateFormat(MYSQLDATEFORMAT
    );
269         String s1;
270         boolean b=true;
271         for (Comment c:cs) {
272             data.clear();
273             data.put("idrecipe", r.getId().toString());
274             data.put("idfrom", c.getUser().getId().toString());
275             data.put("comment", c.getTxt());
276             s1 = dateFormat.format(c.getDate());
277             data.put("date",s1  );
278             String result = mNetUtils.sendPostRequestJson(mSession.
    getURLPath() + PHPADDCOMMENTTORECIPE, data);
279             if (result==null) b=false;
280             if (!result.equals("1")) {
281                 deBugShow( "Retour de "+ PHPADDCOMMENTTORECIPE+" = "+
    result);
282                 b=false;}
283         }
284         return b;
285     }
286     private Boolean syncNotesRecipe(Recipe r) {
287         boolean ret=false;
288         HashMap<String, String> data = new HashMap<>();
289         data.put("idrecipe", r.getId().toString());
290         String result = mNetUtils.sendPostRequestJson(mSession.
    getURLPath() + PHPGETNOTESOFRECIPE, data);
291         List<Note> downloadedNotes=mNetUtils.parseNotesOfRecipe(
    result);
292         if (downloadedNotes==null) downloadedNotes=new ArrayList<>();
293         List<Note> recipeNotes=r.getNotes();
294         if (recipeNotes==null) recipeNotes=new ArrayList<>();
295         //----------- boucle local
296         List<Note> c1oc= new ArrayList<>();
297         Note c,ci;
298         if (!recipeNotes.isEmpty()){
299             for (int i = 0; i < recipeNotes.size(); i++){
300                 ci=recipeNotes.get(i);
301                 c=new Note(ci.getNote(),ci.getUser(),ci.getDate());
302                 if (downloadedNotes.isEmpty()) {
303                     c1oc.add(c);}
304                 else {
305                     if (downloadedNotes.indexOf(c)==-1) c1oc.add(c);}
306             }
307         }
308         //----------- boucle serveur
```

```
309         List<Note> cser= new ArrayList<>();
310         if (!downloadedNotes.isEmpty()){
311             for (int i = 0; i < downloadedNotes.size(); i++){
312                 ci=downloadedNotes.get(i);
313                 c=new Note(ci.getNote(),ci.getUser(),ci.getDate());
314                 if (recipeNotes.isEmpty()){
315                     cser.add(c);}
316                 else{
317                     if (recipeNotes.indexOf(c)==-1) cser.add(c);}
318             }
319         }
320         if (!cser.isEmpty()){
321             for (int i = 0; i < cser.size(); i++){
322                 r.addNote(cser.get(i));
323             }
324             if (cser.size()>0) {
325                 mCookbook.updateRecipe(r);
326                 ret=true;
327             }
328         }
329         if (!c1oc.isEmpty()) {
330             if (!uploadNotes(r,c1oc)) {ret=false;}
331             else {ret=true;}
332         }
333         return ret;
334     }
335     private Boolean uploadNotes(Recipe r, List<Note> cs){
336         if (cs==null) return false;
337         if (cs.size()==0) return true;
338         HashMap<String, String> data = new HashMap<>();
339         DateFormat dateFormat = new SimpleDateFormat(MYSQLDATEFORMAT
    );
340         DecimalFormat formatter = new DecimalFormat("00");
341         String s1;
342         boolean b=true;
343         for (Note c:cs) {
344             data.clear();
345             data.put("idrecipe", r.getId().toString());
346             data.put("idfrom", c.getUser().getId().toString());
347             data.put("note", formatter.format(c.getNote()));
348             s1 = dateFormat.format(c.getDate());
349             data.put("date",s1  );
350             String result = mNetUtils.sendPostRequestJson(mSession.
    getURLPath() + PHPADDNOTETORECIPE, data);
351             if (result==null) b=false;
352             if (!result.equals("1")) {
353                 deBugShow( "Retour de "+ PHPADDNOTETORECIPE+" = "+
    result);
354                 b=false;}
355         }
356         return b;
357     }
358     private Boolean syncTiersRecipe(User user) {
359         boolean ret=false;
360         HashMap<String, String> data = new HashMap<>();
361         data.put("iduser", user.getId().toString());
362         String result = mNetUtils.sendPostRequestJson(mSession.
```

```
362 getURLPath() + PHPGETSTAMPSTIERS, data);
363         if (result==null)return ret;
364         List<Recipe> tiersRecipe=mNetUtils.parseStampsTiers(result);
365         if (tiersRecipe==null) return ret;
366         boolean withphoto=false,update=false;
367         Recipe rloc, rnew;
368         for(Recipe r:tiersRecipe) {
369             rloc=mCookbook.getRecipe(r.getId());
370             if (rloc==null) { // case recipe tiers not found locally
     => download
371                 rnew=downloadRecipe(r);
372                 if (rnew==null) {
373                     deBugShow( "Failure in down loading recipe Id "+
    r.getId());
374                     ret=false;
375                 } else {
376                     mCookbook.addRecipe(rnew);
377                     deBugShow( "Recette "+ rnew.getTitle()+"
    downloadée");
378                     ret=true;
379                 }
380             } else {
381                 deBugShow("Exists localy. Remote status : status "+r.
    getStatus().toString()+
382                         " ("+r.getDate().toString()+") photo :"+r.
    getDatePhoto().toString());
383                 if (rloc.IsVisible()){ // if local recipe submitted
    then skip
384                     withphoto=IsAfterAndNonNull(r.getDatePhoto(),
    rloc.getDatePhoto());
385                     update=IsAfterAndNonNull(r.getDate(), rloc.
    getDate());
386                     deBugShow("Test : Server more recent ? "+update+
    " and photo ?"+withphoto);
387                     if (r.IsMessage()){// case recipe tiers active
    and status needs update
388                         if (recipeAccepted(r)){
389                             deBugShow("Recette "+ rloc.getTitle()+"
    made visible on server");
390                         } else deBugShow("Error in accepting server
    recipe");
391                     }
392                     if(update || withphoto){
393                         if (updateRecipe(rloc, withphoto)){
394                             mCookbook.updateRecipe(rloc);
395                             deBugShow( "Recette "+ rloc.getTitle()+"
    updatée (with photo :"+withphoto+")");
396                             ret=true;
397                         }
398                     }
399                     else continue;
400                 }
401             }
402         }
403         return ret;
404     }
405     private boolean IsAfterAndNonNull(Date ds, Date dl){
```

```
406         Calendar c=Calendar.getInstance();
407         c.set(2000,0,1,0,0, 0);
408         Date d0=c.getTime();
409         boolean morerecent=(ds.compareTo(dl)==1);
410         boolean isnotnull=(ds.compareTo(d0)==1);
411         return morerecent && isnotnull;
412     }
413 
414     private Recipe downloadRecipe(Recipe ref){
415         if (ref==null) return null;
416         HashMap<String,String> data = new HashMap<>();
417         data.put("idrecipe", ref.getId().toString().trim());
418         data.put("iduser", mSession.getUser().getId().toString().trim
    ());
419         String result = mNetUtils.sendPostRequestJson(mSession.
    getURLPath()+PHPGETRECIPEFROMCBWITHPHOTO,data);
420         Recipe r=new Recipe();
421         try {
422             JSONArray jarr1 = new JSONArray(result);
423             if (jarr1.length()==0){
424                 deBugShow( " No json objects from download <"+ref.
    getId()+">");
425                 return null;
426             } else {
427                 if (jarr1.length()>1) deBugShow( " Recette non unique
     dans cookbook : <"+ref.getId()+">");
428                 JSONObject obj = jarr1.getJSONObject(0);
429                 r = new Recipe(UUID.fromString(obj.getString("
    id_recipe")));
430                 if (!mNetUtils.parseObjectRecipe(r,obj, true, true))
    return null;
431             }
432         } catch (Exception e) {
433             deBugShow( " Error parsing CB in downloadRecipe" + e);
434         }
435         return r;
436     }
437     private Boolean updateRecipe(Recipe ref, boolean withphoto){
438         if (ref==null) return null;
439         HashMap<String,String> data = new HashMap<>();
440         data.put("idrecipe", ref.getId().toString().trim());
441         data.put("withphoto", (withphoto)?"1":"0");
442         String result = mNetUtils.sendPostRequestJson(mSession.
    getURLPath()+ PHPGETRECIPEFROMCB,data);
443         Recipe r=new Recipe();
444         try {
445             JSONArray jarr1 = new JSONArray(result);
446             if (jarr1.length()!=1){
447                 deBugShow( " Number of json objects from
    downloadRecipes =" + jarr1.length()+" !");
448                 return null;
449             } else {
450                 JSONObject obj = jarr1.getJSONObject(0);
451                 if (!mNetUtils.parseObjectRecipe(ref,obj, withphoto,
    false)) {
452                     deBugShow( " Null recipe from JSOn object" +
    result);
```

```
453                     return false;
454                 }
455             }
456         } catch (Exception e) {
457             deBugShow( " Error parsing CB in downloadRecipe" + e);
458             return false;
459         }
460         return true;
461     }
462 
463     private Boolean recipeAccepted(Recipe r){
464         if (r==null) return null;
465         HashMap<String,String> data = new HashMap<>();
466         data.put("idrecipe", r.getId().toString().trim());
467         data.put("iduser", mSession.getUser().getId().toString());
468         String result = mNetUtils.sendPostRequestJson(mSession.
    getURLPath()+ PHPACCEPTRECIPE,data);
469         if (!result.equals("1")) {
470             deBugShow( "Retour de "+ PHPACCEPTRECIPE+" = "+result);
471             return false;}
472         return true;
473     }
474 
475     private Boolean checkRecipeRequest(){
476         String iduser=mSession.getUser().getId().toString().trim();
477         if ((iduser==null)||(iduser.equals(""))) return false;
478         HashMap<String,String> data = new HashMap<>();
479         data.put("iduser", iduser);
480         String result = mNetUtils.sendPostRequestJson(mSession.
    getURLPath()+ PHPCHECKREQUESTS,data);
481         return result.equals("1");
482     }
483 }
484 
```

File - D:\4. Softwares\AndroidStudioProjects\CookBook\app\src\main\java\com\fdx\cookbook\CookBooksShort.java

```java
package com.fdx.cookbook;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.util.Log;

import org.json.JSONArray;
import org.json.JSONObject;

import java.util.ArrayList;
import java.util.List;
import java.util.UUID;

public class CookBooksShort {
    private static CookBooksShort ourInstance ;
    private Context mContext;
    private List<Recipe> mCookBook;
    private Integer mSelectType;
    private Integer mDLStatus;
    private static final Integer COMPLETED=2;
    private static final Integer STARTED=1;
    private static final Integer INACTIVE=0;
    private static final Integer RECENT=1;
    private static final Integer POPULAR=2;
    private static final Integer BESTNOTE=3;
    private static final String TAG = "CB_Community";

    public static CookBooksShort get(Context context) {
        if (ourInstance==null){
            ourInstance= new CookBooksShort(context);
        }
        return ourInstance;
    }
    private CookBooksShort(Context context) {
        mContext=context.getApplicationContext();
        mCookBook=new ArrayList<>();
        mDLStatus=INACTIVE;
        mSelectType=RECENT;
    }

    public void addRecipe(Recipe r){
        if (r!=null){
            mCookBook.add(r);
        }
    }

    public List<Recipe> getRecipes(){
        return mCookBook;
    }

    public void clear(){
        mCookBook.clear();
    }
    public void clearCompletely (){
        mDLStatus=INACTIVE;
        mSelectType=RECENT;
        mCookBook.clear();
```

```
 58     }
 59
 60     public void fill(List<Recipe> recipes){
 61         if (recipes==null) return;
 62         for(Recipe r:recipes){
 63             if (r!=null) mCookBook.add(r);
 64         }
 65     }
 66
 67     public void updatePhoto(Recipe rtoupdate){
 68         for(Recipe r:mCookBook){
 69             if(r.isTheSame(rtoupdate)) {
 70                 r.setImage(rtoupdate.getImage());
 71                 return;
 72             }
 73         }
 74     }
 75
 76     public Integer getsize() {
 77         if (mCookBook == null) return 0;
 78         if (mCookBook.isEmpty()) return 0;
 79         return mCookBook.size();
 80     }
 81
 82     public Integer getSelectType() {
 83         return mSelectType;
 84     }
 85
 86     public void setSelectType(Integer selectType) {
 87         mSelectType = selectType;
 88     }
 89
 90     public Integer getDLStatus() {
 91         return mDLStatus;
 92     }
 93
 94     public void getDLStarted() {
 95         mDLStatus = STARTED;
 96     }
 97     public void setDLCompleted() {
 98         mDLStatus = COMPLETED;
 99     }
100     public Boolean isDownloading(){return mDLStatus==STARTED;}
101     public Boolean isNew(){return mDLStatus==INACTIVE;}
102 }
103
```

```java
package com.fdx.cookbook;

import android.content.Context;
import android.content.Intent;
import android.support.v4.app.Fragment;

import java.util.UUID;

public class RecipeActivity extends SingleFragmentActivity {
    private static final String EXTRA_RECIPE_ID="com.fdx.cookbook.recipe_id";

    public static Intent newIntent(Context packageContexte, UUID recipeId) {
        Intent intent=new Intent(packageContexte, RecipeActivity.class);
        intent.putExtra(EXTRA_RECIPE_ID, recipeId);
        return intent;
    }

    @Override
    protected Fragment createFragment(){
        UUID recipeId=(UUID) getIntent().getSerializableExtra(EXTRA_RECIPE_ID);
        return RecipeEditFragment.newInstance(recipeId);
    }

}

```

File - D:\4. Softwares\AndroidStudioProjects\CookBook\app\src\main\java\com\fdx\cookbook\RecipeDbSchema.java

```java
package com.fdx.cookbook;

public class RecipeDbSchema {
    public static final class RecipeTable{
        public static final String NAME="recipes";
        public static final class Cols {
            public static final String UUID="uuid";
            public static final String OWNER="owner";
            public static final String TITLE="title";
            public static final String SOURCE="source";
            public static final String SOURCE_URL="source_url";
            public static final String DATE="date";
            public static final String DATE_PHOTO="date_photo";
            public static final String NBPERS="nbpers";
            public static final String[] STEP={"etape1","etape2","
etape3","etape4",
                    "etape5","etape6","etape7","etape8","etape9"};
            public static final String[] ING={"ing01","ing02","ing03",
"ing04",
                    "ing05","ing06","ing0e7","ing08","ing09","ing10","
ing11",
                    "ing12","ing13","ing14","ing15"};
            public static final String SEASON="season";
            public static final String DIFFICULTY="difficulty";
            public static final String TYPE="type";
            public static final String COMMENTS="comments";
            public static final String STATUS="status";
            public static final String NOTES="notes";
            public static final String MESSAGE="message";
            public static final String MESSAGE_FROM="messagefrom";
            public static final String TS_RECIPE="tsrecipe";
            public static final String TS_PHOTO="tsphoto";
            public static final String TS_COMMENT="tscomment";
            public static final String TS_NOTE="tsnote";
        }
    }
}

```

```java
package com.fdx.cookbook;

import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.graphics.drawable.AnimationDrawable;
import android.os.AsyncTask;
import android.os.Bundle;
import android.support.v4.content.ContextCompat;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.text.Editable;
import android.text.TextWatcher;
import android.text.method.HideReturnsTransformationMethod;
import android.text.method.PasswordTransformationMethod;
import android.util.Log;
import android.view.View;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;

import org.json.JSONArray;
import org.json.JSONObject;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.UUID;
import java.util.regex.Pattern;

public class SplashActivity extends AppCompatActivity {
    private TextView mMoto;
    private String mFamilyEntered;
    private String mMemberEntered;
    private String mPwdEntered;
    private String mPwdRead;
    private TextView mEnterFamilyLbl;
    private EditText mEnterFamily;
    private TextView mEnterMemberLbl;
    private EditText mEnterMember;
    private TextView mEnterPwdLbl;
    private EditText mEnterPwd;
    private TextView mEnterMessage;
    private Button mNewSession;
    private Button mNewMember;
    private Button mNewFamily;
    private ProgressBar mProgressBar;
    private ImageView mNetAnim;
    private AnimationDrawable frameNetAnim;
    private ArrayList<User> memberFamily;
    private ArrayList<Recipe> mRecipes;
```

```
 58      private SessionInfo mSession;
 59      private NetworkUtils mNetUtils;
 60      private User mUser;
 61      private TestConnection tc;
 62      private int mState;
 63      private AlertDialog.Builder mBuilder;
 64      private static final String TAG = "CB_Splash";
 65      private final static int NEW_FAMILY=1;
 66      private final static int NEW_MEMBER=2;
 67      private final static int NEW_SESSION=3;
 68      private final static int NEW_PWD=3;
 69      private final static Integer MINMAX[][]={{8,45},{1,25},{3,25}};
    // min max pour family, member, pwd strings
 70      private static final String REGEX_FAMILY="[-_!?\\w\\p{
    javaLowerCase}\\p{javaUpperCase}()\\p{Space}]*";
 71      private static final String REGEX_MEMBER="[-_\\w\\p{javaLowerCase
    }\\p{javaUpperCase}]*";
 72      private static final String REGEX_PWD="[-_!?\\w\\p{javaLowerCase}
    \\p{javaUpperCase}()]*";
 73      private static final String REGEX[]={REGEX_FAMILY, REGEX_MEMBER,
    REGEX_PWD};
 74      private static final String PHPREQFAMILYCOMP="
    getfamilycomposition.php";
 75      private static final String PHPREQNEWMEMBER="createnewmember.php"
    ;
 76      private static final String PHPREQNEWFAMILY="createnewfamily.php"
    ;
 77      private static final String PHPREQUPLOADPHOTO="
    uploadphotointorecipe.php";
 78      private static final String PHPREQGETCB="getcookbook.php";
 79      private static final String PHPREQGETNOTES="getnotes.php";
 80      private static final String PHPREQGETCOMMENTS="getcomments.php";
 81      private static final String MYSQLDATEFORMAT="yyyy-MM-dd HH:mm:ss"
    ;
 82
 83      @Override
 84      protected void onCreate(Bundle savedInstanceState) {
 85          super.onCreate(savedInstanceState);
 86
 87          mSession= SessionInfo.get(getApplicationContext());
 88          if ((!mSession.IsEmpty())&&(!mSession.IsReqNewSession())){
 89              Intent intent=new Intent(getApplicationContext(),
    RecipeListActivity.class);
 90              startActivity(intent);
 91              finish();
 92          }
 93          mBuilder=new AlertDialog.Builder(this);
 94          mNetUtils=new NetworkUtils(getApplicationContext());
 95          memberFamily=new ArrayList<>();
 96          mRecipes=new ArrayList<>();
 97          tc = new TestConnection();
 98          tc.setTestConnexion(getApplicationContext());
 99          tc.testGo();
100
101          getWindow().setFlags(
102                  WindowManager.LayoutParams.FLAG_FULLSCREEN,
103                  WindowManager.LayoutParams.FLAG_FULLSCREEN);
```

```
104         setContentView(R.layout.activity_splash);
105         mMoto=(TextView) findViewById(R.id.splash_moto);
106         mMoto.setText(R.string.P0_moto);
107         mProgressBar=(ProgressBar) findViewById(R.id.splash_progBar);
108         mProgressBar.setProgress(0);
109         mEnterMessage=(TextView) findViewById(R.id.
    splash_edit_message);
110         mEnterMessage.setText("");
111         mEnterFamilyLbl=(TextView) findViewById(R.id.
    splash_edit_family_lbl);
112         mEnterFamilyLbl.setText(R.string.P0LF);
113         mEnterFamily=(EditText) findViewById(R.id.splash_edit_family
    );
114         mEnterMemberLbl=(TextView) findViewById(R.id.
    splash_edit_member_lbl);
115         mEnterMemberLbl.setText(R.string.P0LM);
116         mEnterMember=(EditText) findViewById(R.id.splash_edit_member
    );
117         mEnterPwdLbl=(TextView) findViewById(R.id.splash_edit_pwd_lbl
    );
118         mEnterPwd=(EditText) findViewById(R.id.splash_edit_pwd);
119         mEnterPwdLbl.setText(R.string.P0LP);
120         mNewSession=(Button) findViewById(R.id.splash_button_session
    );
121         mNewMember=(Button) findViewById(R.id.splash_button_member);
122         mNewFamily=(Button) findViewById(R.id.splash_button_family);
123         if (mSession.IsReqNewSession()){
124             mFamilyEntered=mSession.getUser().getFamily();
125             mEnterFamily.setText(mFamilyEntered);
126             mMemberEntered=mSession.getUser().getName();
127             mEnterMember.setText(mMemberEntered);
128             mPwdEntered="";
129             mEnterPwd.setText(mPwdEntered);
130         } else {
131             mFamilyEntered="";
132             mEnterFamily.setText(mFamilyEntered);
133             mEnterFamily.setHint(R.string.P0HF);
134             mMemberEntered="";
135             mEnterMember.setText(mMemberEntered);
136             mEnterMember.setHint(R.string.P0HM);
137             mPwdEntered="";
138             mEnterPwd.setText(mPwdEntered);
139             mEnterPwd.setHint(R.string.P0HP);
140         }
141         mEnterMessage.setText("");
142         mNewSession.setText(R.string.P0BP);
143         mNewMember.setText(R.string.P0BM);
144         mNewFamily.setText(R.string.P0BF);
145         //updateTop();
146         mNetAnim=(ImageView) findViewById(R.id.net_anim);
147         mNetAnim.setBackgroundResource(R.drawable.network_animation);
148         mNetAnim.setVisibility(View.INVISIBLE);
149         frameNetAnim = (AnimationDrawable) mNetAnim.getBackground();
150         updateDisplayOnAsync(false);
151
152         mEnterFamily.addTextChangedListener(new TextWatcher() {
153             @Override
```

```
154             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) {}
155
156             @Override
157             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
158                 String z=s.toString();
159                 mFamilyEntered=z;
160                 Integer d= getColorOnCondition(z,NEW_FAMILY);
161                 mEnterFamily.setBackgroundColor(ContextCompat.
    getColor(getApplicationContext(), d));
162                 buttonEnabled((d==R.color.bg_enter_val));
163             }
164
165             @Override
166             public void afterTextChanged(Editable s) {}
167         });
168         mEnterMember.addTextChangedListener(new TextWatcher() {
169             @Override
170             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) {}
171
172             @Override
173             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
174                 String z=s.toString();
175                 mMemberEntered=z;
176                 Integer d= getColorOnCondition(z,NEW_MEMBER);
177                 mEnterMember.setBackgroundColor(ContextCompat.
    getColor(getApplicationContext(), d));
178                 buttonEnabled((d==R.color.bg_enter_val));
179             }
180
181             @Override
182             public void afterTextChanged(Editable s) {}
183         });
184         mEnterPwd.addTextChangedListener(new TextWatcher() {
185             @Override
186             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) {}
187
188             @Override
189             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
190                 String z=s.toString();
191                 mPwdEntered=z;
192                 Integer d= getColorOnCondition(z,NEW_PWD);
193                 mEnterPwd.setBackgroundColor(ContextCompat.getColor(
    getApplicationContext(), d));
194                 buttonEnabled((d==R.color.bg_enter_val));
195             }
196
197             @Override
198             public void afterTextChanged(Editable s) {}
199         });
200         mNewSession.setOnClickListener(new View.OnClickListener() {
201             @Override
```

```
202             public void onClick(View v) {
203                 if(testConnectStatus()){
204                     mState=NEW_SESSION;
205                     updateDisplayOnAsync(true);
206                     goAsyncTasks();
207                 } else {
208                 tc.testGo();}
209             }
210         });
211         mNewMember.setOnClickListener(new View.OnClickListener() {
212             @Override
213             public void onClick(View v) {
214                 if(testConnectStatus()){
215                     mState=NEW_MEMBER;
216                     confirmNewUser();
217                 } else {
218                 tc.testGo();}
219             }
220         });
221         mNewFamily.setOnClickListener(new View.OnClickListener() {
222             @Override
223             public void onClick(View v) {
224                 if(testConnectStatus()){
225                     mState=NEW_FAMILY;
226                     confirmNewUser();
227                 } else {
228                 tc.testGo(); }
229             }
230         });
231     }
232
233     private void confirmNewUser(){
234         Context context=getApplicationContext();
235         String name=mMemberEntered+"@"+mFamilyEntered;
236         LinearLayout layout = new LinearLayout(context);
237         layout.setOrientation(LinearLayout.VERTICAL);
238         final TextView message = new TextView(context);
239         message.setText("   "+getString(mState==NEW_FAMILY ? R.string
    .P0COF:R.string.P0COM,name));
240         layout.addView(message);
241         mBuilder.setTitle(R.string.P0COT);
242         mBuilder.setView(layout);
243         mBuilder.setPositiveButton("OK", new DialogInterface.
    OnClickListener() {
244             @Override
245             public void onClick(DialogInterface dialog, int which) {
246                 updateDisplayOnAsync(true);
247                 goAsyncTasks();
248             }
249         });
250         mBuilder.setNegativeButton("Cancel", new DialogInterface.
    OnClickListener() {
251             @Override
252             public void onClick(DialogInterface dialog, int which) {
253                 dialog.cancel();
254             }
255         });
```

```
256         mBuilder.show();
257     }
258 
259     public void ShowHidePass(View view){
260         if(view.getId()==R.id.show_pass_btn){
261             if(mEnterPwd.getTransformationMethod().equals(
    PasswordTransformationMethod.getInstance())){
262                 ((ImageView)(view)).setImageResource(R.drawable.
    ic_pwd_no_vis);
263                 mEnterPwd.setTransformationMethod(
    HideReturnsTransformationMethod.getInstance());
264             }
265             else{
266                 ((ImageView)(view)).setImageResource(R.drawable.
    ic_pwd_vis);
267                 mEnterPwd.setTransformationMethod(
    PasswordTransformationMethod.getInstance());
268 
269             }
270         }
271     }
272 
273     private void updateDisplayOnAsync(Boolean b){
274         int indNet,indClick;
275         if (b){
276             indNet=View.VISIBLE;
277             indClick=View.INVISIBLE;
278             frameNetAnim.start();
279             mProgressBar.setProgress(1);
280         }
281         else {
282             indNet=View.INVISIBLE;
283             indClick=View.VISIBLE;
284             frameNetAnim.stop();
285             mProgressBar.setProgress(0);
286         }
287         mNetAnim.setVisibility(indNet);
288         mProgressBar.setVisibility(indNet);
289         mNewFamily.setVisibility(indClick);
290         mNewMember.setVisibility(indClick);
291         mNewSession.setVisibility(indClick);
292     }
293 
294     private Boolean IsLenOK(String s, int min, int max){
295         int l=s.length();
296         return ((l>min)&&(l<max));
297     }
298 
299     private Integer getColorOnCondition(String z, Integer state){
300         state=state-1;
301         int retDraw =0;
302         int TRUE=R.color.bg_enter_val;
303         int FALSE=R.color.light_red;
304         retDraw=((Pattern.matches(REGEX[state],z)&&(IsLenOK(z,MINMAX[
    state][0],MINMAX[state][1])))?
305                 TRUE : FALSE);
306         Integer err_mess[]={R.string.P0ERUF,R.string.P0RUM,R.string.
```

```
306 P0RUP};
307         if (retDraw==FALSE){
308         mEnterMessage.setText(getResources().getString(err_mess[state
    ],MINMAX[state][0],MINMAX[state][1]));}
309         else {mEnterMessage.setText("");}
310         return retDraw;
311     }
312 
313     private Boolean testConnectStatus(){
314         mEnterFamily.setBackgroundColor(ContextCompat.getColor(
    getApplicationContext(), R.color.bg_enter_val));
315         mEnterMember.setBackgroundColor(ContextCompat.getColor(
    getApplicationContext(), R.color.bg_enter_val));
316         mEnterPwd.setBackgroundColor(ContextCompat.getColor(
    getApplicationContext(), R.color.bg_enter_val));
317         Boolean ret=true;
318         //String message="";
319         if(!mSession.IsConnected()){
320             ret=false;
321             Toast.makeText(getApplicationContext(), getString(R.
    string.P0ER_nocon), Toast.LENGTH_LONG).show();
322         }
323         return ret;
324     }
325 
326     private void buttonEnabled(Boolean b){
327         mNewSession.setEnabled(b);
328         mNewMember.setEnabled(b);
329         mNewFamily.setEnabled(b);
330     }
331 
332 
    /*************************************************************************
    ***********************
333     *                                ASYNC
                                                                *
334     *************************************************************************
    ************************/
335 
336     private void goAsyncTasks() {
337 
338         class GoAsyncTasks extends AsyncTask<Void, Void, Boolean> {
339 
340             @Override
341             protected void onPreExecute() {
342                 super.onPreExecute();
343             }
344 
345             @Override
346             protected void onPostExecute(Boolean b) {
347                 super.onPostExecute(b);
348                 updateDisplayOnAsync(false);
349                 if (b) {
350                     CookBooksShort cbshort=CookBooksShort.get(
    getApplicationContext());
351                     cbshort.clearCompletely();
352                     mSession.setStoredUser(mUser);
```

```
353                     mSession.setListMask("");
354                     Intent intent=new Intent(getApplicationContext
    (), RecipeListActivity.class);
355                     startActivity(intent);
356                     finish();
357                 }
358                 return;
359             }
360 
361             @Override
362             protected Boolean doInBackground(Void... voids) {
363                 CookBook cb=CookBook.get(getApplicationContext());
364                 String s=getFamilyComp();
365                 mProgressBar.setProgress(2);
366                 if (s==null){
367                     mEnterMessage.setText(R.string.P0ER_com);
368                     return false; } else {
369                     memberFamily=parseJsonFamily(s); }
370                 mProgressBar.setProgress(5);
371                 Integer cas=goFamilyGet();
372                 if (cas==0) { return false;}
373                 deBugShow("Cas :"+cas);
374                 if ((cas==NEW_FAMILY)||(cas==NEW_MEMBER)){
375                     if (createMember(cas)){
376                         mProgressBar.setProgress(10);
377                         mEnterMessage.setText(getString(R.string.
    P0OKM));
378                     } else {
379                         mEnterMessage.setText(getString(R.string.
    P0ERM_new));
380                         return false;}
381                 }
382                 cb.clearCookBook();
383                 s=downloadCB();
384                 if (s==null){
385                     deBugShow( "download cookbook failed");
386                     return false;
387                 }
388                 mProgressBar.setProgress(50);
389                 mRecipes=parseJsonCB(s);
390                 if (!downloadNotesAndParse()){
391                     deBugShow("Failure in reading and parsing Notes"
    );
392                     return false;
393                 }
394                 mProgressBar.setProgress(70);
395                 if (!downloadCommentsAndParse()){
396                     deBugShow( "Failure in reading and parsing
    Comments");
397                     return false;
398                 }
399                 mProgressBar.setProgress(70);
400                 cb.fillCookBook(mRecipes);
401                 mProgressBar.setProgress(90);
402                 return true;
403             }
404         }
```

```
405         GoAsyncTasks getAsync = new GoAsyncTasks();
406         getAsync.execute();
407     }
408 
409     private String getFamilyComp(){
410         HashMap<String,String> data = new HashMap<>();
411         data.put("family", mFamilyEntered.trim());
412         String result = mNetUtils.sendPostRequestJson(mSession.
    getURLPath()+PHPREQFAMILYCOMP,data);
413         return result;
414     }
415 
416     private Boolean createMember(int flag){
417         String link=mSession.getURLPath();
418         if (flag==NEW_MEMBER) {
419             link=link+PHPREQNEWMEMBER;
420         } else {
421             if (flag==NEW_FAMILY) {
422                 link=link+PHPREQNEWFAMILY;
423             } else {
424                 deBugShow( "Function createmember flag bad value");
425                 return false;
426             }
427         }
428         deBugShow("URL to create member :"+link);
429         HashMap<String,String> data = new HashMap<>();
430         if ((mFamilyEntered==null)||(mMemberEntered==null)||
431                 (mPwdEntered==null)) {
432             deBugShow( "Anomalie, un element nul");
433             return false;
434         }
435         if ((mFamilyEntered.trim().equals(""))||(mMemberEntered.trim
    ().equals(""))||
436                 (mPwdEntered.trim().equals(""))) {
437             deBugShow( "Anomalie, un element vide");
438             return false;
439         }
440         data.put("family", mFamilyEntered.trim());
441         data.put("pwd", mPwdEntered.trim());
442         data.put("name", mMemberEntered.trim());
443         mUser=new User(mFamilyEntered.trim(),mMemberEntered.trim());
444         data.put("iduser", mUser.getId().toString().trim());
445         data.put("device", mSession.getDevice());
446         String result = mNetUtils.sendPostRequestJson(link,data);
447         if (result.trim().equals("1")) {
448             deBugShow("Function createmember succeed");
449             return true;
450         } else {
451             deBugShow("Function createmember failed");
452             return false;
453         }
454     }
455 
456     private String downloadCB(){
457         HashMap<String,String> data = new HashMap<>();
458         data.put("iduser", mUser.getId().toString().trim());
459         data.put("pwd", mPwdEntered.trim());
```

```
460         String result = mNetUtils.sendPostRequestJson(mSession.
    getURLPath()+PHPREQGETCB,data);
461         return result;
462     }
463 
464     private Boolean downloadNotesAndParse(){
465         HashMap<String,String> data = new HashMap<>();
466         data.put("iduser", mUser.getId().toString().trim());
467         String json = mNetUtils.sendPostRequestJson(mSession.
    getURLPath()+PHPREQGETNOTES,data);
468         UUID uuid;
469         Note n;
470         try {
471             JSONArray jarr1=new JSONArray(json);
472             for (int i=0; i<jarr1.length(); i++){
473                 JSONObject obj = jarr1.getJSONObject(i);
474                 uuid=UUID.fromString(obj.getString("id_recipe"));
475                 n=mNetUtils.parseObjectNote(obj);
476                 for(Recipe r:mRecipes){
477                     if (r.getId().equals(uuid)){
478                         r.addNote(n);
479                         break;
480                     }
481                 }
482             }
483         } catch (Exception e){
484                 deBugShow("Failure in parsing JSON Notes "+e);
485                 return false;
486             }
487         return true;
488     }
489 
490     private Boolean downloadCommentsAndParse(){
491         HashMap<String,String> data = new HashMap<>();
492         data.put("iduser", mUser.getId().toString().trim());
493         String json = mNetUtils.sendPostRequestJson(mSession.
    getURLPath()+PHPREQGETCOMMENTS,data);
494         UUID uuid;
495         Comment c;
496         try {
497             JSONArray jarr1=new JSONArray(json);
498             for (int i=0; i<jarr1.length(); i++){
499                 JSONObject obj = jarr1.getJSONObject(i);
500                 c=mNetUtils.parseObjectComment(obj);
501                 uuid=UUID.fromString(obj.getString("id_recipe"));
502                 if (c!=null) {
503                     for(Recipe r:mRecipes){
504                         if (r.getId().equals(uuid)){
505                             r.addComment(c);
506                             break;
507                         }
508                     }
509                 }
510             }
511         } catch (Exception e){
512             deBugShow("Failure in parsing JSON Array Comments "+e);
513             return false;
```

```
514         }
515         return true;
516     }
517 
518     public ArrayList<User> parseJsonFamily(String json){
519         ArrayList<User> ret=new ArrayList<>();
520         User u;
521         String name,dateString;
522         UUID uuid;
523         Date date;
524         mPwdRead="";
525         try {
526             JSONArray jarr1=new JSONArray(json);
527             for (int i=0; i<jarr1.length(); i++){
528                 JSONObject obj = jarr1.getJSONObject(i);
529                 name=obj.getString("name");
530                 uuid=UUID.fromString(obj.getString("id_user"));
531                 u=new User(mFamilyEntered, name );
532                 u.setId(uuid);
533                 dateString=obj.getString("last_sync");
534                 date=new SimpleDateFormat(MYSQLDATEFORMAT).parse(
    dateString);
535                 u.setDate(date);
536                 ret.add(u);
537                 mPwdRead=obj.getString("pass");
538             }
539         }
540         catch (Exception e){
541             deBugShow("Failure in parsing JSON FamilyGet" +e);
542         }
543         return ret;
544     }
545     public ArrayList<Recipe> parseJsonCB(String json){
546         ArrayList<Recipe> result=new ArrayList<>();
547         Recipe r;
548         try {
549             JSONArray jarr1 = new JSONArray(json);
550             for (int j = 0; j < jarr1.length(); j++) {
551                 JSONObject obj = jarr1.getJSONObject(j);
552                 r = new Recipe(UUID.fromString(obj.getString("
    id_recipe")));
553                 if (mNetUtils.parseObjectRecipe(r,obj, true, true))
    result.add(r);
554             }
555         } catch (Exception e) {
556             deBugShow( " Error parsing CB" + e);
557         }
558         return result;
559     }
560 
561     private Integer goFamilyGet(){
562         switch(mState){
563             case NEW_SESSION : {
564                 if (memberFamily.size()==0) {
565                     mEnterMessage.setText(R.string.P0ERM_nofam);
566                     mEnterFamily.setBackgroundColor(ContextCompat.
    getColor(getApplicationContext(), R.color.light_red)); }
```

```
567                 else {
568                     //mEnterMessage.setText(R.string.P0OKF);
569                     User u= new User("","");
570                     for(User user:memberFamily){
571                         if (user.getName().equals(mMemberEntered)){u=
    user;}
572                     }
573                     if (u.getName().equals("")) {
574                         mEnterMessage.setText(R.string.P0ERF_nomem);
575                         mEnterMember.setBackgroundColor(ContextCompat
    .getColor(getApplicationContext(), R.color.light_red));
576                     } else {
577                         if (!mPwdRead.equals(mPwdEntered.trim())){
578                             mEnterMessage.setText(R.string.
    P0ERP_wrong);
579                             mEnterPwd.setBackgroundColor(
    ContextCompat.getColor(getApplicationContext(), R.color.light_red));
580                         } else {
581                             mUser=u;
582                         mEnterMessage.setText(getString(R.string.
    P0OKP, u.getName()));
583                         return NEW_SESSION; }
584                     }
585                 }
586                 break;
587             }
588             case NEW_FAMILY : {
589                 if (memberFamily.size()==0) {
590                     mEnterMessage.setText(R.string.P0OK_done);
591                     return NEW_FAMILY;
592                 }
593                 else {
594                     mEnterMessage.setText(R.string.P0ERF_notnew);
595                     mEnterFamily.setBackgroundColor(ContextCompat.
    getColor(getApplicationContext(), R.color.light_red));
596                 }
597                 break;
598             }
599             case NEW_MEMBER : {
600                 if (memberFamily.size()==0) {
601                     mEnterMessage.setText(R.string.P0ERM_nofam);
602                     mEnterFamily.setBackgroundColor(ContextCompat.
    getColor(getApplicationContext(), R.color.light_red));
603                 }
604                 else {
605                     // family found
606                     User u= new User("","");
607                     for(User user:memberFamily){
608                         if (user.getName().equals(mMemberEntered)){u=
    user;}
609                     }
610                     if (u.getName().equals("")) {
611                         if (!mPwdRead.equals(mPwdEntered.trim())){
612                             mEnterMessage.setText(R.string.
    P0ERP_wrong);
613                             mEnterPwd.setBackgroundColor(
    ContextCompat.getColor(getApplicationContext(), R.color.light_red));
```

```
614                             } else {
615                                 mEnterMessage.setText(R.string.P0OKM);
616                                 return NEW_MEMBER;}
617                         } else {
618                             mEnterMessage.setText(R.string.P0ERM_notnew);
619                             mEnterMember.setBackgroundColor(ContextCompat
    .getColor(getApplicationContext(), R.color.light_red));
620                         }
621                     }
622                     break;
623                 }
624                 default : {
625                     deBugShow( "Switch case on mState anomaly : "+mState
    );
626                 }
627             }
628             return 0;
629         }
630         private void deBugShow(String s){
631             //Log.d(TAG, s);
632         }
633 }
634
```

File - D:\4. Softwares\AndroidStudioProjects\CookBook\app\src\main\java\com\fdx\cookbook\TestConnection.java

```java
package com.fdx.cookbook;

import android.content.Context;
import android.os.AsyncTask;

import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.URL;

public class TestConnection {
    private static String PHP204="return204.php";
    private static final String TAG = "DebugTestConnection";
    private SessionInfo mSession;

    public void TestConnection() {
        //
    }
    public void setTestConnexion(Context context) {
        mSession = SessionInfo.get(context);
    }

    public void testGo(){

        class testCon extends AsyncTask<Void, Void, Boolean> {

            @Override
            protected Boolean doInBackground(Void... voids) {
                try {
                    URL url = new URL(mSession.getURLPath()+PHP204);
                    HttpURLConnection conn = (HttpURLConnection) url.
openConnection();
                    conn.setConnectTimeout(mSession.getConnectTimeout
());
                    conn.setReadTimeout(mSession.getReadTimeout());
                    conn.setRequestMethod("HEAD");
                    InputStream in = conn.getInputStream();
                    int status = conn.getResponseCode();
                    in.close();
                    conn.disconnect();
                    if (status == HttpURLConnection.HTTP_NO_CONTENT) {
                        //Log.d(TAG, "Test 204 : true ");
                        return true;
                    } else {return false;}
                } catch (Exception e) {return false;}
            }

            @Override
            protected void onPostExecute(Boolean s) {
                super.onPostExecute(s);
                mSession.setConnection(s);
            }
        }

        testCon test = new testCon();
        test.execute();
    }
}
```

File - D:\4. Softwares\AndroidStudioProjects\CookBook\app\src\main\java\com\fdx\cookbook\RecipeBaseHelper.java

```java
package com.fdx.cookbook;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

public class RecipeBaseHelper extends SQLiteOpenHelper {
    private static final int VERSION=1;
    private static final String DATA_BASE_NAME="recipeBase.db";  // REINIT BASE
    public RecipeBaseHelper(Context context) {
        super(context, DATA_BASE_NAME, null, VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        String st="",si="";
        Recipe r=new Recipe();
        for(int i=0;i<r.getNbStepMax();i++){
            st=st+ RecipeDbSchema.RecipeTable.Cols.STEP[i];
            st=st+", ";
        }
        for(int i=0;i<r.getNbIngMax();i++){
            si=si+ RecipeDbSchema.RecipeTable.Cols.ING[i];
            si=si+", ";
        }
        db.execSQL("create table "+ RecipeDbSchema.RecipeTable.NAME+"("+
                " _id integer primary key autoincrement, "+
                RecipeDbSchema.RecipeTable.Cols.UUID+", "+
                RecipeDbSchema.RecipeTable.Cols.OWNER+", "+
                RecipeDbSchema.RecipeTable.Cols.TITLE+", "+
                RecipeDbSchema.RecipeTable.Cols.SOURCE+", "+
                RecipeDbSchema.RecipeTable.Cols.SOURCE_URL+", "+
                RecipeDbSchema.RecipeTable.Cols.DATE+", "+
                RecipeDbSchema.RecipeTable.Cols.DATE_PHOTO+", "+
                RecipeDbSchema.RecipeTable.Cols.NBPERS+", "+
                st +
                si +
                RecipeDbSchema.RecipeTable.Cols.SEASON+", "+
                RecipeDbSchema.RecipeTable.Cols.DIFFICULTY+","+
                RecipeDbSchema.RecipeTable.Cols.TYPE+","+
                RecipeDbSchema.RecipeTable.Cols.COMMENTS+", "+
                RecipeDbSchema.RecipeTable.Cols.STATUS+", "+
                RecipeDbSchema.RecipeTable.Cols.NOTES+", "+
                RecipeDbSchema.RecipeTable.Cols.MESSAGE+", "+
                RecipeDbSchema.RecipeTable.Cols.MESSAGE_FROM+", "+
                RecipeDbSchema.RecipeTable.Cols.TS_RECIPE+", "+
                RecipeDbSchema.RecipeTable.Cols.TS_PHOTO+", "+
                RecipeDbSchema.RecipeTable.Cols.TS_COMMENT+", "+
                RecipeDbSchema.RecipeTable.Cols.TS_NOTE+
                ")"
        );

    }

    @Override
```

```
56      public void onUpgrade(SQLiteDatabase db, int oldVersion, int
    newVersion) {
57
58      }
59  }
60
```

File - D:\4. Softwares\AndroidStudioProjects\CookBook\app\src\main\java\com\fdx\cookbook\RecipeDifficulty.java

```java
package com.fdx.cookbook;

public enum RecipeDifficulty {
    QUICK, EASY, ELABORATE, SOPHISTICATED, UNDEFINED;
    private static RecipeDifficulty[] list=RecipeDifficulty.values();

    public static RecipeDifficulty getDifficulty(int i){
        return list[i];
    }

    public static int getIndex(RecipeDifficulty r){
        for(int i=0;i<list.length;i++) {
            if (list[i].equals(r)){ return i;}
        }
        return 0;
    }
}

```

```java
package com.fdx.cookbook;

import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.support.v4.app.DialogFragment;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.Button;
import android.widget.RatingBar;

import java.util.UUID;

public class RatePickerFragment extends DialogFragment {
    public static final String EXTRA_RATE = "com.fdx.cookbook.rate";
    private static final String ARG_ID= "recipeId";
    private static final String TAG = "DebugRatePickerFragment";
    private int mRate;
    private UUID mUUID;
    public static RatePickerFragment newInstance(UUID uuid) {
        Bundle args = new Bundle();
        args.putSerializable(ARG_ID, uuid);
        RatePickerFragment fragment = new RatePickerFragment();
        fragment.setArguments(args);
        return fragment;
    }

    @Override
    public Dialog onCreateDialog(Bundle SavedInstanceState) {
        mUUID = (UUID) getArguments().getSerializable(ARG_ID);
        View v= LayoutInflater.from(getActivity())
                .inflate(R.layout.rank_dialog, null);
        RatingBar ratingBar = (RatingBar)v.findViewById(R.id.
dialog_ratingbar);
        ratingBar.setRating(4);
        AlertDialog.Builder dialog= new AlertDialog.Builder(
getActivity())
                .setView(v)
                .setTitle(R.string.DS_title)
                .setPositiveButton(android.R.string.ok,
                        new DialogInterface.OnClickListener() {
                            @Override
                            public void onClick(DialogInterface dialog
, int which) {
                                mRate=(int) ratingBar.getRating();
                                addNotetoRecipe();
                                sendResult(Activity.RESULT_OK, mRate);
                            }
                        });
        dialog.setNegativeButton("Cancel",
                new DialogInterface.OnClickListener() {

                    @Override
                    public void onClick(DialogInterface dialog, int
```

```
which) {
                        return ;
                    }
                });
        AlertDialog dia=dialog.show();
        Button b=dia.getButton(DialogInterface.BUTTON_POSITIVE);
        if (b != null) {
            b.setTextColor(getResources().getColor(R.color.fg_icon));
        }
        return dia;
    }

    private void sendResult(int resultCode, int rate) {
        if (getTargetFragment() == null) {
            return;
        }
        // renvoie le new rating (cela ne serta à rien, juste pour essayer)
        Intent intent = new Intent();
        intent.putExtra(EXTRA_RATE, rate);
        getTargetFragment()
                .onActivityResult(getTargetRequestCode(), resultCode, intent);
    }
     private void addNotetoRecipe(){
         SessionInfo loSession= SessionInfo.get(getActivity());
         CookBook locookbook=CookBook.get(getActivity());
         Recipe r=locookbook.getRecipe(mUUID);
         User lu=loSession.getUser();
         r.addNote(new Note(mRate, lu));
         r.updateTS(AsynCallFlag.NEWRATING, true);
         locookbook.updateRecipe(r);
     }
}

```

```java
package com.fdx.cookbook;

import android.app.Activity;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.content.pm.ResolveInfo;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.net.Uri;
import android.os.Bundle;
import android.provider.MediaStore;
import android.support.v4.app.Fragment;
import android.support.v4.content.FileProvider;
import android.support.v4.view.MenuCompat;
import android.text.Editable;
import android.text.TextWatcher;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.ScrollView;
import android.widget.Spinner;
import android.widget.TextView;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.Date;
import java.util.List;
import java.util.UUID;

public class RecipeEditFragment extends Fragment {
    // Constantes
    private static final String ARG_RECIPE_ID="recipe_id";
    private static final int PICK_IMAGE= 3;
    private static final String TAG = "CB_EditFrag";
    private static final String FPROVIDER="com.fdx.cookbook.
fileprovider";
    private static final String UUIDNULL="00000000-0000-0000-0000-
000000000000";
    private Recipe mRecipe;
    private Recipe mRecipeInit;
    private UUID mRecipeId;
    private File mPhotoFile;
    private EditText mTitleField;
    private EditText mSourceField;
```

```
 56     private EditText mSourceUrl;
 57     private EditText mNbPersField;
 58     private TextView[] mStepTextNum;
 59     private EditText[] mStepTextEdit;
 60     private ImageButton mStepInc;
 61     private int mStepNb;
 62     private TextView[] mIngTextNum;
 63     private EditText[] mIngTextEdit;
 64     private ImageButton mIngInc;
 65     private int mIngNb;
 66     private Spinner mSeasonSpinner;
 67     private Spinner mTypeSpinner;
 68     private Spinner mDifficultySpinner;
 69     private ScrollView mScroll;
 70     private ImageView mPhotoView;
 71     private Bitmap mBmp;
 72
 73     public static RecipeEditFragment newInstance(UUID recipeId){
 74         Bundle args=new Bundle();
 75         args.putSerializable(ARG_RECIPE_ID, recipeId);
 76         RecipeEditFragment fragment=new RecipeEditFragment();
 77         fragment.setArguments(args);
 78         return fragment;
 79     }
 80
 81     @Override
 82     public void onCreate(Bundle savedInstanceState){
 83         super.onCreate(savedInstanceState);
 84         mRecipe=new Recipe();
 85         mRecipeInit=new Recipe();
 86         mRecipeId=(UUID) getArguments().getSerializable(ARG_RECIPE_ID
    );
 87         setHasOptionsMenu(true);
 88         SessionInfo session=SessionInfo.get(getActivity());
 89         if (!IsRecipeNew(mRecipeId)) {
 90             mRecipe=CookBook.get(getActivity()).getRecipe(mRecipeId);
 91             mRecipeInit=CookBook.get(getActivity()).getRecipe(
    mRecipeId);
 92             mPhotoFile=CookBook.get(getActivity()).getPhotoFile(
    mRecipe);
 93         } else {
 94             mRecipe.setOwner(session.getUser());
 95             mRecipeInit.setOwner(session.getUser());
 96             mRecipeInit.setId(mRecipe.getId());
 97         }
 98         mStepNb=mRecipe.getNbStep();
 99         mIngNb=mRecipe.getNbIng();
100     }
101
102     @Override
103     public void onPause(){
104         super.onPause();
105     }
106
107     @Override
108     public void onCreateOptionsMenu(Menu menu, MenuInflater inflater
    ) {
```

```
109         super.onCreateOptionsMenu(menu, inflater);
110         inflater.inflate(R.menu.fragment_recipe, menu);
111 
112     }
113 
114     @Override
115     public boolean onOptionsItemSelected(MenuItem item) {
116         switch (item.getItemId()) {
117             case R.id.recipe_menu_done:
118                 if (mRecipe.getTitle().length()>0){
119                     if (!mRecipe.hasNotChangedSince(mRecipeInit)){
120                     mRecipe.updateTS(AsynCallFlag.NEWRECIPE, true);
121                     mRecipe.setDate(new Date());
122                     }
123                     if (mBmp!=null){
124                         NetworkUtils networkutils=new NetworkUtils(
    getContext());
125                         networkutils.saveBmpInRecipe(mBmp, mRecipe);
126                         mRecipe.setDatePhoto(new Date());
127                         mRecipe.updateTS(AsynCallFlag.NEWPHOTO,true);
128                     }
129                     if (IsRecipeNew(mRecipeId)){
130                         CookBook.get(getActivity()).addRecipe(mRecipe
    );
131                     } else {
132                         CookBook.get(getActivity()).updateRecipe(
    mRecipe);
133                     }
134 
135                 }
136                 getActivity().onBackPressed();
137                 return true;
138             case R.id.recipe_menu_delete:
139                 if (!IsRecipeNew(mRecipeId)){
140                     CookBook.get(getActivity()).markRecipeToDelete(
    mRecipe);
141                 }
142                 getActivity().onBackPressed();
143                 return true;
144             default:
145                 return super.onOptionsItemSelected(item);
146         }
147     }
148 
149     @Override
150     public View onCreateView(LayoutInflater inflater, ViewGroup
    container, Bundle savedInstanceState){
151         final View v=inflater.inflate(R.layout.fragment_recipe_edit,
    container, false);
152         mScroll=(ScrollView) v.findViewById(R.id.
    fragment_recipe_scroll);
153         mPhotoView=(ImageView) v.findViewById(R.id.recipe_photo);
154         updatePhotoView();
155         mPhotoView.setOnClickListener(new View.OnClickListener() {
156             @Override
157             public void onClick(View v) {
158                 Intent intent = new Intent();
```

```
159                 intent.setType("image/*");
160                 intent.setAction(Intent.ACTION_GET_CONTENT);
161                 startActivityForResult(Intent.createChooser(intent, "
    Select Picture"), PICK_IMAGE);
162             }
163         });
164         mTitleField= (EditText) v.findViewById(R.id.recipe_title);
165         mTitleField.setText(mRecipe.getTitle());
166         mTitleField.addTextChangedListener(new TextWatcher() {
167             @Override
168             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) {
169             }
170 
171             @Override
172             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
173                 mRecipe.setTitle(s.toString());
174             }
175 
176             @Override
177             public void afterTextChanged(Editable s) {
178             }
179         });
180 
181         mSourceField= (EditText) v.findViewById(R.id.recipe_source);
182         mSourceField.setText(mRecipe.getSource());
183         mSourceField.addTextChangedListener(new TextWatcher() {
184             @Override
185             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) {
186             }
187 
188             @Override
189             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
190                 mRecipe.setSource(s.toString());
191             }
192 
193             @Override
194             public void afterTextChanged(Editable s) {
195             }
196         });
197 
198         mSourceUrl=(EditText) v.findViewById(R.id.recipe_source_url);
199         mSourceUrl.setText(mRecipe.getSource_url().toString());
200         mSourceUrl.addTextChangedListener(new TextWatcher() {
201             @Override
202             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) {
203 
204             }
205 
206             @Override
207             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
208                 if (s.toString().equals("")) {} else {
```

```
209                     try {
210                         URL url=new URL(s.toString());
211                         mRecipe.setSource_url(url);
212                     } catch (MalformedURLException e) {
213                         deBug( "onTextChanged de mSource_url >" +s.
    toString()+"< Failed >"+ e);
214                     }
215                 }
216             }
217 
218             @Override
219             public void afterTextChanged(Editable s) {
220             }
221         });
222 
223         mNbPersField= (EditText) v.findViewById(R.id.recipe_nbpers);
224         mNbPersField.setText(mRecipe.getNbPers()+"");
225         mNbPersField.addTextChangedListener(new TextWatcher() {
226             @Override
227             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) {
228 
229             }
230 
231             @Override
232             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
233                 if (s.toString().equals("")) {} else {
234                     int nb_entered = Integer.parseInt(s.toString());
235                     if ((nb_entered > 0) && (nb_entered < 13)) {
236                         mRecipe.setNbPers(nb_entered);
237                     }
238                 }
239             }
240 
241             @Override
242             public void afterTextChanged(Editable s) {
243 
244             }
245         });
246 
247         mSeasonSpinner= (Spinner) v.findViewById(R.id.recipe_season);
248         ArrayAdapter<CharSequence> adapterSeason = ArrayAdapter.
    createFromResource(getContext(),
249                 R.array.recipe_season_array, R.layout.
    edit_spinner_item);
250         mSeasonSpinner.setAdapter(adapterSeason);
251         mSeasonSpinner.setSelection(RecipeSeason.getIndex(mRecipe.
    getSeason()));
252         mSeasonSpinner.setOnItemSelectedListener(new AdapterView.
    OnItemSelectedListener() {
253             @Override
254             public void onItemSelected(AdapterView<?> parent, View
    view, int position, long id) {
255                 mRecipe.setSeason(RecipeSeason.getSeason(position));
256             }
257 
```

```
258             @Override
259             public void onNothingSelected(AdapterView<?> parent) {
260
261             }
262         });
263
264         mTypeSpinner= (Spinner) v.findViewById(R.id.recipe_type);
265         ArrayAdapter<CharSequence> adapterType = ArrayAdapter.
    createFromResource(getContext(),
266                 R.array.recipe_type_array, R.layout.edit_spinner_item
    );
267         mTypeSpinner.setAdapter(adapterType);
268         mTypeSpinner.setSelection(RecipeType.getIndex(mRecipe.getType
    ()));
269         mTypeSpinner.setOnItemSelectedListener(new AdapterView.
    OnItemSelectedListener() {
270             @Override
271             public void onItemSelected(AdapterView<?> parent, View
    view, int position, long id) {
272                 mRecipe.setType(RecipeType.getType(position));
273             }
274
275             @Override
276             public void onNothingSelected(AdapterView<?> parent) {
277
278             }
279         });
280
281         mDifficultySpinner= (Spinner) v.findViewById(R.id.
    recipe_difficulty);
282         ArrayAdapter<CharSequence> adapterDifficulty = ArrayAdapter.
    createFromResource(getContext(),
283                 R.array.recipe_difficulty_array, R.layout.
    edit_spinner_item);
284         mDifficultySpinner.setAdapter(adapterDifficulty);
285         mDifficultySpinner.setSelection(RecipeDifficulty.getIndex(
    mRecipe.getDifficulty()));
286         mDifficultySpinner.setOnItemSelectedListener(new AdapterView.
    OnItemSelectedListener() {
287             @Override
288             public void onItemSelected(AdapterView<?> parent, View
    view, int position, long id) {
289                 mRecipe.setDifficulty(RecipeDifficulty.getDifficulty(
    position));
290             }
291
292             @Override
293             public void onNothingSelected(AdapterView<?> parent) {
294
295             }
296         });
297         final int[] rID= {R.id.recipe_S1,R.id.recipe_S2,R.id.
    recipe_S3,R.id.recipe_S4,
298                 R.id.recipe_S5,R.id.recipe_S6,R.id.recipe_S7,R.id.
    recipe_S8,R.id.recipe_S9};
299         final int[] rID_in= {R.id.recipe_S1_in,R.id.recipe_S2_in,R.id
    .recipe_S3_in,R.id.recipe_S4_in,
```

```
300                 R.id.recipe_S5_in,R.id.recipe_S6_in,R.id.recipe_S7_in
    ,R.id.recipe_S8_in,R.id.recipe_S9_in};
301         mStepTextNum= new TextView[mRecipe.getNbStepMax()];
302         mStepTextEdit=new EditText[mRecipe.getNbStepMax()];
303         for(int i=0;i<mRecipe.getNbStepMax();i++) {
304             mStepTextNum[i] = (TextView) v.findViewById(rID[i]);
305             mStepTextEdit[i] = (EditText) v.findViewById(rID_in[i]);
306         }
307         mStepInc=(ImageButton) v.findViewById(R.id.step_add);
308         updateListStep(v);
309 
310         mStepTextEdit[0].addTextChangedListener(new TextWatcher() {
311             @Override
312             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) { }
313             @Override
314             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
315                 mRecipe.setStep(0+1, s.toString());
316             }
317             @Override
318             public void afterTextChanged(Editable s) {}
319         });
320         mStepTextEdit[1].addTextChangedListener(new TextWatcher() {
321             @Override
322             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) { }
323             @Override
324             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
325                 mRecipe.setStep(1+1, s.toString());
326             }
327             @Override
328             public void afterTextChanged(Editable s) {}
329         });
330         mStepTextEdit[2].addTextChangedListener(new TextWatcher() {
331             @Override
332             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) { }
333             @Override
334             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
335                 mRecipe.setStep(2+1, s.toString());
336             }
337             @Override
338             public void afterTextChanged(Editable s) {}
339         });
340         mStepTextEdit[3].addTextChangedListener(new TextWatcher() {
341             @Override
342             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) { }
343             @Override
344             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
345                 mRecipe.setStep(3+1, s.toString());
346             }
347             @Override
```

```
348             public void afterTextChanged(Editable s) {}
349         });
350         mStepTextEdit[4].addTextChangedListener(new TextWatcher() {
351             @Override
352             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) { }
353             @Override
354             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
355                 mRecipe.setStep(4+1, s.toString());
356             }
357             @Override
358             public void afterTextChanged(Editable s) {}
359         });
360         mStepTextEdit[5].addTextChangedListener(new TextWatcher() {
361             @Override
362             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) { }
363             @Override
364             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
365                 mRecipe.setStep(5+1, s.toString());
366             }
367             @Override
368             public void afterTextChanged(Editable s) {}
369         });
370         mStepTextEdit[6].addTextChangedListener(new TextWatcher() {
371             @Override
372             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) { }
373             @Override
374             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
375                 mRecipe.setStep(6+1, s.toString());
376             }
377             @Override
378             public void afterTextChanged(Editable s) {}
379         });
380         mStepTextEdit[7].addTextChangedListener(new TextWatcher() {
381             @Override
382             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) { }
383             @Override
384             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
385                 mRecipe.setStep(7+1, s.toString());
386             }
387             @Override
388             public void afterTextChanged(Editable s) {}
389         });
390         mStepTextEdit[8].addTextChangedListener(new TextWatcher() {
391             @Override
392             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) { }
393             @Override
394             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
```

```
395                 mRecipe.setStep(8+1, s.toString());
396             }
397             @Override
398             public void afterTextChanged(Editable s) {}
399         });
400 
401         mStepInc.setOnClickListener(new View.OnClickListener() {
402             @Override
403             public void onClick(View v) {
404                 updateListStep(v);
405             }
406         });
407 
408         final int[] rIngID= {R.id.recipe_I01,R.id.recipe_I02,R.id.
    recipe_I03,R.id.recipe_I04,
409                 R.id.recipe_I05,R.id.recipe_I06,R.id.recipe_I07,R.id.
    recipe_I08,
410                 R.id.recipe_I09,R.id.recipe_I10,R.id.recipe_I11,R.id.
    recipe_I12,
411                 R.id.recipe_I13,R.id.recipe_I14,R.id.recipe_I15};
412         final int[] rIngID_in= {R.id.recipe_I01_in,R.id.recipe_I02_in
    ,R.id.recipe_I03_in,R.id.recipe_I04_in,
413                 R.id.recipe_I05_in,R.id.recipe_I06_in,R.id.
    recipe_I07_in,R.id.recipe_I08_in,
414                 R.id.recipe_I09_in,R.id.recipe_I10_in,R.id.
    recipe_I11_in,R.id.recipe_I12_in,
415                 R.id.recipe_I13_in,R.id.recipe_I14_in,R.id.
    recipe_I15_in};
416         mIngTextNum= new TextView[mRecipe.getNbIngMax()];
417         mIngTextEdit=new EditText[mRecipe.getNbIngMax()];
418         for(int i=0;i<mRecipe.getNbIngMax();i++) {
419             mIngTextNum[i] = (TextView) v.findViewById(rIngID[i]);
420             mIngTextEdit[i] = (EditText) v.findViewById(rIngID_in[i
    ]);
421         }
422         mIngInc=(ImageButton) v.findViewById(R.id.ing_add);
423         mIngTextEdit[0].setText(mRecipe.getIngredient(0 + 1));
424         mIngTextEdit[1].setText(mRecipe.getIngredient(1 + 1));
425         mIngTextEdit[2].setText(mRecipe.getIngredient(2 + 1));
426         mIngInc=(ImageButton) v.findViewById(R.id.ing_add);
427         updateListIng(v);
428 
429         mIngTextEdit[0].addTextChangedListener(new TextWatcher() {
430             @Override
431             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) { }
432             @Override
433             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
434                 mRecipe.setIngredient(0+1, s.toString());
435             }
436             @Override
437             public void afterTextChanged(Editable s) {}
438         });
439         mIngTextEdit[1].addTextChangedListener(new TextWatcher() {
440             @Override
441             public void beforeTextChanged(CharSequence s, int start,
```

```
441 int count, int after) { }
442             @Override
443             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
444                 mRecipe.setIngredient(1+1, s.toString());
445             }
446             @Override
447             public void afterTextChanged(Editable s) {}
448         });
449         mIngTextEdit[2].addTextChangedListener(new TextWatcher() {
450             @Override
451             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) { }
452             @Override
453             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
454                 mRecipe.setIngredient(2+1, s.toString());
455             }
456             @Override
457             public void afterTextChanged(Editable s) {}
458         });
459         mIngTextEdit[3].addTextChangedListener(new TextWatcher() {
460             @Override
461             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) { }
462             @Override
463             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
464                 mRecipe.setIngredient(3+1, s.toString());
465             }
466             @Override
467             public void afterTextChanged(Editable s) {}
468         });
469         mIngTextEdit[4].addTextChangedListener(new TextWatcher() {
470             @Override
471             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) { }
472             @Override
473             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
474                 mRecipe.setIngredient(4+1, s.toString());
475             }
476             @Override
477             public void afterTextChanged(Editable s) {}
478         });
479         mIngTextEdit[5].addTextChangedListener(new TextWatcher() {
480             @Override
481             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) { }
482             @Override
483             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
484                 mRecipe.setIngredient(5+1, s.toString());
485             }
486             @Override
487             public void afterTextChanged(Editable s) {}
488         });
```

```
489         mIngTextEdit[6].addTextChangedListener(new TextWatcher() {
490             @Override
491             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) { }
492             @Override
493             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
494                 mRecipe.setIngredient(6+1, s.toString());
495             }
496             @Override
497             public void afterTextChanged(Editable s) {}
498         });
499         mIngTextEdit[7].addTextChangedListener(new TextWatcher() {
500             @Override
501             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) { }
502             @Override
503             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
504                 mRecipe.setIngredient(7+1, s.toString());
505             }
506             @Override
507             public void afterTextChanged(Editable s) {}
508         });
509         mIngTextEdit[8].addTextChangedListener(new TextWatcher() {
510             @Override
511             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) { }
512             @Override
513             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
514                 mRecipe.setIngredient(8+1, s.toString());
515             }
516             @Override
517             public void afterTextChanged(Editable s) {}
518         });
519         mIngTextEdit[9].addTextChangedListener(new TextWatcher() {
520             @Override
521             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) { }
522             @Override
523             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
524                 mRecipe.setIngredient(9+1, s.toString());
525             }
526             @Override
527             public void afterTextChanged(Editable s) {}
528         });
529         mIngTextEdit[10].addTextChangedListener(new TextWatcher() {
530             @Override
531             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) { }
532             @Override
533             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
534                 mRecipe.setIngredient(10+1, s.toString());
535             }
```

```
536             @Override
537             public void afterTextChanged(Editable s) {}
538         });
539         mIngTextEdit[11].addTextChangedListener(new TextWatcher() {
540             @Override
541             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) { }
542             @Override
543             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
544                 mRecipe.setIngredient(11+1, s.toString());
545             }
546             @Override
547             public void afterTextChanged(Editable s) {}
548         });
549         mIngTextEdit[12].addTextChangedListener(new TextWatcher() {
550             @Override
551             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) { }
552             @Override
553             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
554                 mRecipe.setIngredient(12+1, s.toString());
555             }
556             @Override
557             public void afterTextChanged(Editable s) {}
558         });
559         mIngTextEdit[13].addTextChangedListener(new TextWatcher() {
560             @Override
561             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) { }
562             @Override
563             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
564                 mRecipe.setIngredient(13+1, s.toString());
565             }
566             @Override
567             public void afterTextChanged(Editable s) {}
568         });
569         mIngTextEdit[14].addTextChangedListener(new TextWatcher() {
570             @Override
571             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) { }
572             @Override
573             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
574                 mRecipe.setIngredient(14+1, s.toString());
575             }
576             @Override
577             public void afterTextChanged(Editable s) {}
578         });
579         mIngInc.setOnClickListener(new View.OnClickListener() {
580             @Override
581             public void onClick(View v) {
582                 updateListIng(v);
583             }
584         });
```

```
585
586         return v;
587     }
588
589
590     @Override
591     public void onActivityResult(int requestCode, int resultCode,
    Intent data) {
592         if (resultCode != Activity.RESULT_OK) {
593             return;
594         }
595         if (requestCode==PICK_IMAGE){
596             if (data == null) return ;
597             try{
598             InputStream inputStream = getContext().getContentResolver
    ().openInputStream(data.getData());
599             PictureUtils picutil=new PictureUtils();
600             Bitmap bmp=BitmapFactory.decodeStream(inputStream);
601             mBmp=picutil.getScaledBitmap(bmp,600);
602             mPhotoView.setImageBitmap(mBmp);
603             } catch(Exception e) {
604                 deBug("recover onActivityResult error:"+e);
605             }
606         }
607     }
608
609     private void updatePhotoView(){
610         if (mPhotoFile==null || !mPhotoFile.exists()){
611             mPhotoView.setImageDrawable(getResources().getDrawable(R.
    drawable.ic_recipe_camera));
612         } else {
613             Bitmap bitmap=PictureUtils.getScaledBitmap(mPhotoFile.
    getPath(), getActivity());
614             mPhotoView.setImageBitmap(bitmap);
615         }
616     }
617
618     private void updateListStep(View v) {
619         String cas;
620         int i_cur=mRecipe.getNbStep();
621         int i_max=mRecipe.getNbStepMax();
622         for (int i=0;i<mRecipe.getNbStepMax();i++) {
623             cas = "NOTVISIBLE";
624             if (i == 0) cas = "FIRST";
625             if (i < i_cur) cas = "ACTIVE";
626             if ((i == i_cur)&&(i_cur<i_max)) cas = "EMPTY_READY";
627             if ((i == i_cur)&&(i_cur==i_max)) cas = "ACTIVE";
628             switch (cas) {
629                 case "FIRST":
630                     if (mRecipe.getNbStep() == 0) {
631                         mStepTextEdit[i].setText("");
632                         mStepTextEdit[i].setHint(R.string.P3S1H);}
633                     else mStepTextEdit[i].setText(mRecipe.getStep(i
     + 1));
634                     break;
635                 case "ACTIVE":
636                     mStepTextNum[i].setVisibility(View.VISIBLE);
```

```
637                     mStepTextEdit[i].setVisibility(View.VISIBLE);
638                     mStepTextEdit[i].setText(mRecipe.getStep(i + 1));
639                     mStepTextEdit[i].setHint("");
640                     break;
641                 case "EMPTY_READY":
642                     mStepTextNum[i].setVisibility(View.VISIBLE);
643                     mStepTextEdit[i].setVisibility(View.VISIBLE);
644                     mStepTextEdit[i].setText("");
645                     mStepTextEdit[i].setHint(R.string.P3S1H);
646                     break;
647                 case "NOTVISIBLE":
648                     mStepTextNum[i].setVisibility(View.GONE);
649                     mStepTextEdit[i].setVisibility(View.GONE);
650                     mStepTextEdit[i].setText("");
651                     mStepTextEdit[i].setHint("");
652             }
653         }
654     }
655
656     private void updateListIng(View v) {
657         String cas;
658         int i_cur=mRecipe.getNbIng();
659         int i_max=mRecipe.getNbIngMax();
660         for (int i=0;i<mRecipe.getNbIngMax();i++) {
661             cas = "NOTVISIBLE";
662             if (i == 0) cas = "FIRST";
663             if (i < i_cur) cas = "ACTIVE";
664             if ((i == i_cur)&&(i_cur<i_max)) cas = "EMPTY_READY";
665             if ((i == i_cur)&&(i_cur==i_max)) cas = "ACTIVE";
666             switch (cas) {
667                 case "FIRST":
668                     if (mRecipe.getNbIng() == 0) {
669                         mIngTextEdit[i].setText("");
670                         mIngTextEdit[i].setHint(R.string.P3IT);}
671                     else mIngTextEdit[i].setText(mRecipe.
    getIngredient(i + 1));
672                     break;
673                 case "ACTIVE":
674                     mIngTextNum[i].setVisibility(View.VISIBLE);
675                     mIngTextEdit[i].setVisibility(View.VISIBLE);
676                     mIngTextEdit[i].setText(mRecipe.getIngredient(i
     + 1));
677                     mIngTextEdit[i].setHint("");
678                     break;
679                 case "EMPTY_READY":
680                     mIngTextNum[i].setVisibility(View.VISIBLE);
681                     mIngTextEdit[i].setVisibility(View.VISIBLE);
682                     mIngTextEdit[i].setText("");
683                     mIngTextEdit[i].setHint(R.string.P3IT);
684                     break;
685                 case "NOTVISIBLE":
686                     mIngTextNum[i].setVisibility(View.GONE);
687                     mIngTextEdit[i].setVisibility(View.GONE);
688                     mIngTextEdit[i].setText("");
689                     mIngTextEdit[i].setHint("");
690             }
691         }
```

```
692     }
693
694     private Boolean IsRecipeNew(UUID uuid){
695         return uuid.toString().equals(UUIDNULL);
696     }
697
698     private void deBug(String s){
699         // Log.d(TAG, s);
700     }
701 }
702
```

```java
package com.fdx.cookbook;

import android.support.v4.app.Fragment;

public class RecipeListActivity extends SingleFragmentActivity {
    @Override
    protected Fragment createFragment(){
        return new RecipeListFragment();
    }
}

```

```java
package com.fdx.cookbook;

import android.app.Activity;
import android.content.Intent;
import android.graphics.Bitmap;
import android.net.Uri;
import android.os.AsyncTask;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v4.view.MenuCompat;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.support.v7.widget.SearchView;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.RatingBar;

import android.widget.TextView;
import android.widget.Toast;

import java.io.File;
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.UUID;

public class RecipeListFragment extends Fragment {
    private RecyclerView mRecipeRecyclerView;
    private RecipeAdapter mAdapter;
    private SessionInfo mSession;
    private Recipe mRecipeInit;
    private MenuItem mMessageItem;
    private static final String SAVED_SORT_STATUS="sort";
    private AsyncCallClass mInstanceAsync;
    private ListMask mListMask;
    private static final String TAG = "CB_ListFra";
    private static final String DIALOG_RATE = "DialogRate";
    private static final int REQUEST_RATE = 0;
    private static final String UUIDNULL="00000000-0000-0000-0000-000000000000";

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setHasOptionsMenu(true);
        mSession= SessionInfo.get(getActivity());
        mRecipeInit=new Recipe();
```

```
57          AppCompatActivity activity = (AppCompatActivity) getActivity
    ();
58          User u=mSession.getUser();
59          //String nameInSubtitle=getString(R.string.P2U_txt,u.getName
    (),u.getFamily());
60          String nameInSubtitle=u.getName();
61          activity.getSupportActionBar().setSubtitle(nameInSubtitle);
62          mInstanceAsync = new AsyncCallClass(getContext());
63          startSync();
64          mListMask=new ListMask(mSession.getUser());
65          String mask=mSession.getListMask();
66          if ((!mask.equals(""))&&(mask!=null)) mListMask.
    updateFromString(mask);
67      }
68
69      @Override
70      public void onActivityResult(int requestCode, int resultCode,
    Intent data) {
71          if (resultCode != Activity.RESULT_OK) return;
72          else {
73              startSync();
74              updateUI();}
75      }
76
77      @Override
78      public View onCreateView(LayoutInflater inflater, ViewGroup
    container,
79                               Bundle savedInstanceState) {
80          View view = inflater.inflate(R.layout.fragment_recipe_list,
    container, false);
81          mRecipeRecyclerView = (RecyclerView) view.findViewById(R.id.
    recipe_recycler_view);
82          mRecipeRecyclerView.setLayoutManager(new LinearLayoutManager(
    getActivity()));
83          if (savedInstanceState!=null){
84              mListMask.updateFromString(savedInstanceState.getString(
    SAVED_SORT_STATUS));
85          }
86          updateUI();
87          return view;
88      }
89
90      @Override
91      public void onResume(){
92          super.onResume();
93          startSync();
94          String mask=mSession.getListMask();
95          if ((!mask.equals(""))&&(mask!=null)) mListMask.
    updateFromString(mask);
96          updateUI();
97      }
98
99      @Override
100     public void onPause(){
101         super.onPause();
102         mSession.setListMask(mListMask.convertToString());
103     }
```

```
104
105     @Override
106     public void onSaveInstanceState(Bundle outState) {
107         super.onSaveInstanceState(outState);
108         outState.putString(SAVED_SORT_STATUS, mListMask.
    convertToString());
109     }
110
111     @Override
112     public void onCreateOptionsMenu(Menu menu, MenuInflater inflater
    ){
113         super.onCreateOptionsMenu(menu, inflater);
114         inflater.inflate(R.menu.fragment_recipe_list, menu);
115         mMessageItem = menu.findItem(R.id.new_mail);
116         CookBook cookbook=CookBook.get(getActivity());
117         Boolean b=((cookbook.isThereMail())||(mSession.
    IsRecipeRequest()));
118         mMessageItem.setVisible(b);
119         mMessageItem.setShowAsAction(b ? MenuItem.
    SHOW_AS_ACTION_ALWAYS:MenuItem.SHOW_AS_ACTION_NEVER);
120         //MenuCompat.setGroupDividerEnabled(menu, true);
121         MenuItem searchItem = menu.findItem(R.id.menu_item_search);
122         final SearchView searchView = (SearchView) searchItem.
    getActionView();
123         searchView.setOnQueryTextListener(new SearchView.
    OnQueryTextListener() {
124             @Override
125             public boolean onQueryTextSubmit(String s) {
126                 Toast.makeText(getContext(), getString(mListMask.
    toggleSearch(s)),
127                         Toast.LENGTH_SHORT ).show();
128                 updateUI();
129                 return true;
130             }
131             @Override
132             public boolean onQueryTextChange(String s) {
133                 if (s.equals("")){
134                     Toast.makeText(getContext(), getString(mListMask.
    toggleSearch(s)),
135                         Toast.LENGTH_SHORT ).show();
136                     updateUI();
137                 }
138                 return false;
139             }
140         });
141     }
142
143     @Override
144     public boolean onOptionsItemSelected(MenuItem item){
145         Intent intent;
146         switch (item.getItemId()){
147             case R.id.new_recipe:
148                 intent= RecipeActivity.newIntent(getActivity(), UUID.
    fromString( UUIDNULL));
149                 startActivity(intent);
150                 return true;
151             case R.id.list_logout:
```

```
152                 mSession.setReqNewSession(true);
153                 intent=new Intent(getActivity().getApplicationContext
    (), SplashActivity.class);
154                 startActivity(intent);
155                 return true;
156             case R.id.list_sync:
157                 startSync();
158                 return true;
159             case R.id.new_mail:
160                 CookBook cookbook=CookBook.get(getActivity());
161                 Boolean thereismail=(cookbook.isThereMail())||(
    mSession.IsRecipeRequest());
162                 if (!thereismail) {
163                     mMessageItem.setVisible(false);
164                     mMessageItem.setShowAsAction(MenuItem.
    SHOW_AS_ACTION_NEVER);
165                 } else {
166                     intent= RecipeMailDisplayActivity.newIntent(
    getActivity());
167                     startActivity(intent);
168                 }
169                 return true;
170             case R.id.list_filter:
171                 mListMask.reset();
172                 updateUI();
173                 return true;
174             case R.id.list_com:
175                 intent= RecipeCommunityActivity.newIntent(getActivity
    ());
176                 startActivity(intent);
177                 return true;
178             case R.id.feedback:
179                 intent = new Intent(Intent.ACTION_SENDTO);
180                 intent.putExtra(Intent.EXTRA_SUBJECT, "Feedback on
    application");
181                 intent.putExtra(Intent.EXTRA_TEXT, "Your comment");
182                 intent.setData(Uri.parse("mailto:cookbookfamily.
    founder@gmail.com"));
183                 intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
184                 startActivity(intent);
185                 return true;
186             default:
187                 return super.onOptionsItemSelected(item);
188         }
189     }
190
191     private void startSync(){
192         AsyncTask.Status as=mInstanceAsync.getStatus();
193         if (as== AsyncTask.Status.RUNNING){
194         }
195         if (as== AsyncTask.Status.PENDING){
196             mInstanceAsync.execute();
197         }
198         if (as== AsyncTask.Status.FINISHED){
199             mInstanceAsync=new AsyncCallClass(getContext());
200             mInstanceAsync.execute();
201         }
```

```
202     }
203
204     private void updateUI() {
205         CookBook cookbook=CookBook.get(getActivity());
206         List<Recipe> recipes_in=cookbook.getRecipes();
207         List<Recipe> recipes=new ArrayList<>();
208         recipes.addAll(recipes_in);
209         for(Recipe r:recipes_in){
210             if (!mListMask.isRecipeSelected(r)) recipes.remove(r);
211         }
212         if (recipes.isEmpty()){
213             Integer message;
214             if (cookbook.getRecipesVisibles().isEmpty())
215                 message = (cookbook.isThereMail() ? R.string.TEW : R.
    string.TEZ);
216             else message=R.string.TEY;
217             Toast.makeText(getContext(), getString(message), Toast.
    LENGTH_LONG).show();
218         } else {
219             if (mListMask.isTitleSorted()) {
220                 Collections.sort(recipes,
221                         (r1, r2) -> {
222                             return (r1.getTitle().compareTo(r2.
    getTitle()));
223                         });
224             } else {
225                 Collections.sort(recipes,
226                         (r1, r2) -> {
227                             return (r2.getDate().compareTo(r1.getDate
    ()));
228                         });
229             }
230             if (mListMask.isNoteSorted()) {
231                 Collections.sort(recipes,
232                         (r1, r2) -> {
233                             return (r2.getNbNotes()-r1.getNbNotes()+(
    int)(100* (r2.getNoteAvg() - r1.getNoteAvg())));
234                         });
235             }
236         }
237         if (mAdapter==null){
238             mAdapter=new RecipeAdapter(recipes);
239             mRecipeRecyclerView.setAdapter(mAdapter);
240         } else {
241             mAdapter.setRecipes(recipes);
242             mAdapter.notifyDataSetChanged();
243         }
244     }
245
246     private class RecipeHolder extends RecyclerView.ViewHolder
247                                     implements View.OnClickListener {
248         private TextView mTitleTextView;
249         private TextView mSourceTextView;
250         private TextView mNoteTextView;
251         private TextView mRatingTextView;
252         private TextView mDifficulty;
253         private TextView mType;
```

```
254         private ImageView mEditIcon;
255         private ImageView mPhotoView;
256         private ImageView mSunIcon;
257         private ImageView mIceIcon;
258         private File mPhotoFile;
259         private Recipe mRecipe;
260         private RatingBar mRating;
261         public RecipeHolder(LayoutInflater inflater,ViewGroup parent
    ){
262             super(inflater.inflate(R.layout.list_item_recipe, parent
    , false));
263             itemView.setOnClickListener(this);
264             mTitleTextView= (TextView) itemView.findViewById(R.id.
    recipe_title);
265             mSourceTextView= (TextView) itemView.findViewById(R.id.
    recipe_source);
266             mNoteTextView= (TextView) itemView.findViewById(R.id.
    recipe_note);
267             mNoteTextView= (TextView) itemView.findViewById(R.id.
    recipe_note);
268             mEditIcon=(ImageView) itemView.findViewById(R.id.
    recipe_edit);
269             mSunIcon=(ImageView) itemView.findViewById(R.id.
    recipe_img_sun);
270             mIceIcon=(ImageView) itemView.findViewById(R.id.
    recipe_img_ice);
271             mDifficulty=(TextView) itemView.findViewById(R.id.
    recipe_difficulty);
272             mType=(TextView) itemView.findViewById(R.id.recipe_type);
273             mPhotoView=(ImageView) itemView.findViewById(R.id.
    recipe_photo);
274             mPhotoFile=CookBook.get(getActivity()).getPhotoFile(
    mRecipe);
275             mRating=(RatingBar) itemView.findViewById(R.id.
    recipe_list_ratingBar);
276             mRatingTextView= (TextView) itemView.findViewById(R.id.
    textEmpty);
277             mRatingTextView.setOnClickListener(new View.
    OnClickListener() {
278                 @Override
279                 public void onClick(View v) {
280                     FragmentManager manager = getFragmentManager();
281                     RatePickerFragment dialog = RatePickerFragment.
    newInstance(mRecipe.getId());
282                     dialog.setTargetFragment(RecipeListFragment.this
    , REQUEST_RATE);
283                     dialog.show(manager, DIALOG_RATE);
284                 }
285             });
286             mTitleTextView.setOnClickListener(new View.
    OnClickListener() {
287                 @Override
288                 public void onClick(View v) {
289                     Toast.makeText(getContext(),
290                             getString(mListMask.toggleTitle()),
291                             Toast.LENGTH_SHORT ).show();
292                     updateUI();
```

```
293                 }
294             });
295             mNoteTextView.setOnClickListener(new View.OnClickListener
    (){
296                 @Override
297                 public void onClick(View v){
298                     Toast.makeText(getContext(), getString(mListMask.
    toggleSortNote()),
299                             Toast.LENGTH_SHORT ).show();
300                     updateUI();
301                 }
302             });
303             mSourceTextView.setOnClickListener(new View.
    OnClickListener() {
304                 @Override
305                 public void onClick(View v) {
306                     Toast.makeText(getContext(), getString(mListMask.
    toggleUser(mRecipe.getOwner()),
307                             mRecipe.getOwner().getName()),Toast.
    LENGTH_SHORT ).show();
308                     updateUI();
309                 }
310             });
311             mSunIcon.setOnClickListener(new View.OnClickListener() {
312                 @Override
313                 public void onClick(View v) {
314                     Toast.makeText(getContext(), getString(mListMask.
    toggleSun()),Toast.LENGTH_SHORT ).show();
315                     updateUI();
316                 }
317             });
318             mIceIcon.setOnClickListener(new View.OnClickListener() {
319                 @Override
320                 public void onClick(View v) {
321                     Toast.makeText(getContext(), getString(mListMask.
    toggleWinter()),
322                             Toast.LENGTH_SHORT ).show();
323                     updateUI();
324                 }
325             });
326             mDifficulty.setOnClickListener(new View.OnClickListener
    () {
327                 @Override
328                 public void onClick(View v) {
329                     Toast.makeText(getContext(),
330                             getString(mListMask.toggleDifficulty(
    mRecipe.getDifficulty())),
331                             Toast.LENGTH_SHORT ).show();
332                     updateUI();
333                 }
334             });
335             mType.setOnClickListener(new View.OnClickListener() {
336                 @Override
337                 public void onClick(View v) {
338                     Toast.makeText(getContext(),
339                             getString(mListMask.toggleType(mRecipe.
    getType())),
```

```
340                                 Toast.LENGTH_SHORT ).show();
341                         updateUI();
342                     }
343                 });
344                 mEditIcon.setOnClickListener(new View.OnClickListener() {
345                     @Override
346                     public void onClick(View v) {
347                         Intent intent= RecipeActivity.newIntent(
    getActivity(),mRecipe.getId());
348                         startActivity(intent);
349                     }
350                 });
351                 mPhotoView.setOnClickListener(new View.OnClickListener
    () {
352                     @Override
353                     public void onClick(View v) {
354                         Intent intent=RecipeDisplayActivity.newIntent(
    getActivity(),mRecipe.getId());
355                         startActivity(intent);
356                     }
357                 });
358             }
359             public void bind(Recipe recipe){
360                 mRecipe=recipe;
361                 mTitleTextView.setText(mRecipe.getTitle());
362                 mSourceTextView.setText(getString(R.string.P1_de, mRecipe
    .getOwner().getName()));
363                 //mSourceTextView.setText(mRecipe.getFlag()); // for
    debug
364                 mRating.setRating((float) mRecipe.getNoteAvg());
365                 DecimalFormat df = new DecimalFormat("#.#");
366                 mNoteTextView.setText(df.format(mRecipe.getNoteAvg())+
    " ("+mRecipe.getNbNotes()+")");
367                 mPhotoFile=CookBook.get(getActivity()).getPhotoFile(
    mRecipe);
368                 int idff= RecipeDifficulty.getIndex(mRecipe.getDifficulty
    ());
369                 String[] stringArray = getResources().getStringArray(R.
    array.recipe_difficulty_array);
370                 mDifficulty.setText(stringArray[idff]);
371                 int ityp= RecipeType.getIndex(mRecipe.getType());
372                 String[] stringArrayTyp = getResources().getStringArray(R
    .array.recipe_type_array);
373                 mType.setText(stringArrayTyp[ityp]);
374                 if (mPhotoFile==null || !mPhotoFile.exists()){
375                     mPhotoView.setImageDrawable(getResources().
    getDrawable(R.drawable.ic_recipe_see));
376                 } else {
377                     Bitmap bitmap=PictureUtils.getScaledBitmap(mPhotoFile
    .getPath(), getActivity());
378                     mPhotoView.setImageBitmap(bitmap);
379                 }
380                 mIceIcon.setImageResource((mRecipe.IsWinter()) ? R.
    drawable.ic_recipe_ice : R.drawable.ic_recipe_ice_disabled);
381                 mSunIcon.setImageResource((mRecipe.IsSummer()) ? R.
    drawable.ic_recipe_sun : R.drawable.ic_recipe_sun_disabled);
382                 mEditIcon.setVisibility((mRecipe.getOwner().getId().
```

```
382 equals(mSession.getUser().getId())) ? View.VISIBLE : View.GONE);
383         }
384
385         @Override
386         public void onClick(View v){
387
388         }
389     }
390
391     private class RecipeAdapter extends RecyclerView.Adapter<
    RecipeHolder> {
392         private List<Recipe> mRecipes;
393         public RecipeAdapter(List<Recipe> recipes){
394             mRecipes=recipes;
395         }
396
397         @NonNull
398         @Override
399         public RecipeHolder onCreateViewHolder(@NonNull ViewGroup
    parent, int viewType) {
400             LayoutInflater layoutInflater=LayoutInflater.from(
    getActivity());
401             return new RecipeHolder(layoutInflater, parent);
402         }
403
404         @Override
405         public void onBindViewHolder(@NonNull RecipeHolder
    recipeHolder, int i) {
406             Recipe recipe=mRecipes.get(i);
407             recipeHolder.bind(recipe);
408         }
409
410         @Override
411         public int getItemCount() {
412             return mRecipes.size();
413         }
414
415         public void setRecipes(List<Recipe> recipes){
416             mRecipes=recipes;
417         }
418     }
419 }
420
```

```java
package com.fdx.cookbook;

import android.database.Cursor;
import android.database.CursorWrapper;
import android.util.Log;

import java.net.MalformedURLException;
import java.net.URL;
import java.util.Date;
import java.util.UUID;

import static com.fdx.cookbook.RecipeDbSchema.*;

public class RecipeCursorWrapper extends CursorWrapper {
    public RecipeCursorWrapper(Cursor cursor){
        super(cursor);
    }
    private static final String TAG = "DebugCursorWrapper";

    public Recipe getRecipe(){
        String uuidString = getString(getColumnIndex(RecipeTable.Cols.UUID));
        Recipe r=new Recipe(UUID.fromString(uuidString));
        String ownerString = getString(getColumnIndex(RecipeTable.Cols.OWNER));
        r.getOwnerDeserialized(ownerString);
        String title = getString(getColumnIndex(RecipeTable.Cols.TITLE));
        r.setTitle(title);
        String source = getString(getColumnIndex(RecipeTable.Cols.SOURCE));
        r.setSource(source);
        String sourceURL=getString(getColumnIndex(RecipeTable.Cols.SOURCE_URL));
        try {
            URL url = new URL(sourceURL);
            r.setSource_url(url);}
            catch (MalformedURLException e) {
                //fdx Log(TAG, "getURL from cursor failed");
            }
        long date = getLong(getColumnIndex(RecipeTable.Cols.DATE));
        r.setDate(new Date(date));
        date = getLong(getColumnIndex(RecipeTable.Cols.DATE_PHOTO));
        r.setDatePhoto(new Date(date));
        int nbpers=getInt(getColumnIndex(RecipeTable.Cols.NBPERS));
        r.setNbPers(nbpers);
        String[] step= new String[r.getNbStepMax()];
        for(int i=0;i<r.getNbStepMax();i++){
            step[i]=getString(getColumnIndex(RecipeTable.Cols.STEP[i]));
            r.setStep(i+1, step[i]);
        }
        String[] ing=new String[r.getNbIngMax()];
        for(int i=0;i<r.getNbIngMax(); i++){
            ing[i]=getString(getColumnIndex(RecipeTable.Cols.ING[i]));
            r.setIngredient(i+1, ing[i]);
        }
```

```
52          String season = getString(getColumnIndex(RecipeTable.Cols.
    SEASON));
53          r.setSeason(RecipeSeason.valueOf(season));
54          String difficulty = getString(getColumnIndex(RecipeTable.Cols
    .DIFFICULTY));
55          r.setDifficulty(RecipeDifficulty.valueOf(difficulty));
56          String type = getString(getColumnIndex(RecipeTable.Cols.TYPE
    ));
57          r.setType(RecipeType.valueOf(type));
58          String comments = getString(getColumnIndex(RecipeTable.Cols.
    COMMENTS));
59          r.getCommentsDeserialised(comments);
60          String status = getString(getColumnIndex(RecipeTable.Cols.
    STATUS));
61          r.setStatus(StatusRecipe.valueOf(status));
62          String notes = getString(getColumnIndex(RecipeTable.Cols.
    NOTES));
63          r.getNotesDeserialised(notes);
64          String message = getString(getColumnIndex(RecipeTable.Cols.
    MESSAGE));
65          r.setMessage(message);
66          String fromString = getString(getColumnIndex(RecipeTable.Cols
    .MESSAGE_FROM));
67          r.getFromDeserialized(fromString);
68          int tsrecipe = getInt(getColumnIndex(RecipeTable.Cols.
    TS_RECIPE));
69          r.updateTS(AsynCallFlag.NEWRECIPE, (tsrecipe==1));
70          int tsphoto = getInt(getColumnIndex(RecipeTable.Cols.TS_PHOTO
    ));
71          r.updateTS(AsynCallFlag.NEWPHOTO, (tsphoto==1));
72          int tscomment = getInt(getColumnIndex(RecipeTable.Cols.
    TS_COMMENT));
73          r.updateTS(AsynCallFlag.NEWCOMMENT, (tscomment==1));
74          int tsnote = getInt(getColumnIndex(RecipeTable.Cols.TS_NOTE
    ));
75          r.updateTS(AsynCallFlag.NEWRATING, (tsnote==1));
76          return r;
77      }
78  }
79
```

```java
package com.fdx.cookbook;

import android.content.Context;
import android.content.Intent;
import android.support.v4.app.Fragment;

import java.util.UUID;

public class RecipeDisplayActivity extends SingleFragmentActivity {
    private static final String EXTRA_RECIPE_ID="com.fdx.cookbook.recipe_id";

    public static Intent newIntent(Context packageContexte, UUID recipeId) {
        Intent intent=new Intent(packageContexte, RecipeDisplayActivity.class);
        intent.putExtra(EXTRA_RECIPE_ID, recipeId);
        return intent;
    }

    @Override
    protected Fragment createFragment() {
        UUID recipeId=(UUID) getIntent().getSerializableExtra(EXTRA_RECIPE_ID);
        return RecipeDisplayFragment.newInstance(recipeId);
    }
}

```

```java
package com.fdx.cookbook;

import android.content.DialogInterface;
import android.content.Intent;
import android.graphics.Bitmap;
import android.os.AsyncTask;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.support.v7.app.AlertDialog;
import android.text.Editable;
import android.text.TextWatcher;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.RatingBar;
import android.widget.ScrollView;
import android.widget.TextView;
import android.widget.Toast;

import java.io.File;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.URL;
import java.text.DecimalFormat;
import java.util.HashMap;
import java.util.UUID;
import java.util.regex.Pattern;

public class RecipeDisplayFragment extends Fragment {
    private static final String ARG_RECIPE_ID="recipe_id";
    private static final String TAG = "CB_RecipeDisplayFrag";
    private Recipe mRecipe;
    private File mPhotoFile;
    private int mStepNb;
    private int mIngNb;
    private String newcomment;
    private String DEFAULT_URL="https://wwww.cookbookfamily.com";
    private SessionInfo mSession;
    private ImageView mDPhotoView;
    private TextView mDTitleText;
    private RatingBar mDRatingBar;
    private TextView mDRatingBarText;
    private TextView mDAuthorText;
    private TextView mDDifficulty;
    private TextView mDType;
    private ImageView mSunIcon;
    private ImageView mIceIcon;
    private TextView mDSourceText;
    private TextView mDSourceUrl;
    private TextView mDIngTitle;
    private TextView mDStepTitle;
```

```
 58      private TextView mDSourceTitle;
 59      private TextView mDComTitle;
 60      private String mToFamily;
 61      private String mToMember;
 62      private String mToMessage;
 63      private TextView[] mDStepText;
 64      private TextView[] mDIngText;
 65      private TextView[] mDComText;
 66      private EditText mDNexComment;
 67      private ImageView mDEnterComment;
 68      private ScrollView mScroll;
 69      private final static Integer MINMAX[][]={{8,45},{1,25},{3,25}};
    // min max pour family, member, pwd strings
 70      private static final String REGEX_FAMILY="[-_!?\\w\\p{
    javaLowerCase}\\p{javaUpperCase}()\\p{Space}]*";
 71      private static final String REGEX_MEMBER="[-_\\w\\p{javaLowerCase
    }\\p{javaUpperCase}]*";
 72
 73      public static RecipeDisplayFragment newInstance(UUID recipeId) {
 74          Bundle args=new Bundle();
 75          args.putSerializable(ARG_RECIPE_ID, recipeId);
 76          RecipeDisplayFragment fragment=new RecipeDisplayFragment();
 77          fragment.setArguments(args);
 78          return fragment;
 79      }
 80
 81      @Override
 82      public void onCreate(Bundle savedInstanceState) {
 83          super.onCreate(savedInstanceState);
 84          mRecipe=new Recipe();
 85          UUID recipeId=(UUID) getArguments().getSerializable(
    ARG_RECIPE_ID);
 86          mRecipe=CookBook.get(getActivity()).getRecipe(recipeId);
 87          mPhotoFile=CookBook.get(getActivity()).getPhotoFile(mRecipe);
 88          mSession= SessionInfo.get(getActivity());
 89          setHasOptionsMenu(true);
 90          mStepNb=mRecipe.getNbStep();
 91          mIngNb=mRecipe.getNbIng();
 92          mToFamily="";
 93          mToMember="";
 94      }
 95      @Override
 96      public void onPause(){
 97          super.onPause();
 98          //todo pourquoi update recipe ?
 99          CookBook.get(getActivity()).updateRecipe(mRecipe);
100      }
101
102      @Override
103      public void onCreateOptionsMenu(Menu menu, MenuInflater inflater
    ) {
104          super.onCreateOptionsMenu(menu, inflater);
105          inflater.inflate(R.menu.fragment_recipe_display, menu);
106          MenuItem menuItem = menu.findItem(R.id.recipe_menu_edit);
107          if(!mRecipe.getOwner().getId().equals(mSession.getUser().
    getId())){
108              menuItem.setVisible(false);
```

```
109         }
110     }
111
112     @Override
113     public boolean onOptionsItemSelected(MenuItem item) {
114         switch (item.getItemId()) {
115             case R.id.recipe_menu_report:
116                 Intent i=new Intent(Intent.ACTION_SEND);
117                 i.setType("text/plain");
118                 i.putExtra(Intent.EXTRA_TEXT, getRecipeReport());
119                 i.putExtra(Intent.EXTRA_SUBJECT,getString(R.string.
    P2MU_send));
120                 startActivity(i);
121                 return true;
122             case R.id.recipe_menu_edit:
123                 Intent intent= RecipeActivity.newIntent(getActivity
    (),mRecipe.getId());
124                 startActivity(intent);
125                 return true;
126             case R.id.recipe_mail:
127                 LinearLayout layout = new LinearLayout(getContext());
128                 layout.setOrientation(LinearLayout.VERTICAL);
129                 final EditText familyBox = new EditText(getContext
    ());
130                 familyBox.setHint(getString(R.string.P0HF));
131                 layout.addView(familyBox);
132                 final EditText memberBox = new EditText(getContext
    ());
133                 memberBox.setHint(getString(R.string.P0HM));
134                 layout.addView(memberBox);
135                 final EditText messageBox = new EditText(getContext
    ());
136                 messageBox.setHint(getString(R.string.P4M_mes));
137                 layout.addView(messageBox);
138                 AlertDialog.Builder builder = new AlertDialog.Builder
    (getContext());
139                 builder.setTitle(getString(R.string.P4M_title));
140                 builder.setView(layout);
141                 builder.setPositiveButton("OK", new DialogInterface.
    OnClickListener() {
142                     @Override
143                     public void onClick(DialogInterface dialog, int
    which) {
144                         mToFamily = familyBox.getText().toString();
145                         mToMember = memberBox.getText().toString();
146                         mToMessage = messageBox.getText().toString();
147                         Boolean b=(Pattern.matches(REGEX_FAMILY,
    mToFamily));
148                         b=b&&(Pattern.matches(REGEX_MEMBER,mToMember
    ));
149                         b=b&&(IsLenOK(mToFamily,MINMAX[0][0],MINMAX[0
    ][1]));
150                         b=b&&(IsLenOK(mToMember,MINMAX[1][0],MINMAX[1
    ][1]));
151                         b=b&&(Pattern.matches(REGEX_FAMILY,mToMessage
    ));
152                         if (b) {
```

```
153                         sendMailAsync sendmail = new sendMailAsync();
154                         sendmail.execute();
155                         } else {
156                             Toast.makeText(getActivity(),getString(R.
    string.P4M_err), Toast.LENGTH_SHORT).show();
157                         }
158                     }
159                 });
160                 builder.setNegativeButton("Cancel", new
    DialogInterface.OnClickListener() {
161                     @Override
162                     public void onClick(DialogInterface dialog, int
    which) {
163                         dialog.cancel();
164                     }
165                 });
166                 builder.show();
167                 return true;
168             case R.id.recipe_menu_delete:
169                 CookBook.get(getActivity()).markRecipeToDelete(
    mRecipe);
170                 getActivity().onBackPressed();
171             default:
172                 return super.onOptionsItemSelected(item);
173         }
174     }
175
176     @Override
177     public View onCreateView(LayoutInflater inflater, ViewGroup
    container, Bundle savedInstanceState){
178         final View v=inflater.inflate(R.layout.
    fragment_display_recipe, container, false);
179         mScroll=(ScrollView) v.findViewById(R.id.
    fragment_recipe_scroll);
180         mDTitleText =(TextView) v.findViewById(R.id.
    recipe_display_title);
181         mDPhotoView=(ImageView) v.findViewById(R.id.
    recipe_display_photo);
182         updatePhotoView();
183         mDRatingBar=(RatingBar) v.findViewById(R.id.
    recipe_display_ratingBar);
184         mDRatingBarText=(TextView) v.findViewById(R.id.
    recipe_display_ratingBar_txt);
185         mDDifficulty=(TextView) v.findViewById(R.id.
    recipe_display_difficulty);
186         mDType=(TextView) v.findViewById(R.id.recipe_display_type);
187         mSunIcon=(ImageView) v.findViewById(R.id.recipe_img_sun) ;
188         mIceIcon=(ImageView) v.findViewById(R.id.recipe_img_ice) ;
189         mDAuthorText=(TextView) v.findViewById(R.id.
    recipe_display_author);
190         mDSourceText=(TextView) v.findViewById(R.id.
    recipe_display_source);
191         mDSourceUrl=(TextView) v.findViewById(R.id.
    recipe_display_source_url);
192         mDIngTitle=(TextView) v.findViewById(R.id.
    recipe_display_title_ing);
193         mDStepTitle=(TextView) v.findViewById(R.id.
```

```
193 recipe_display_title_step);
194         mDSourceTitle=(TextView) v.findViewById(R.id.
    recipe_display_source_title);
195         mDComTitle=(TextView)v.findViewById(R.id.
    recipe_display_comment_title);
196         final int[] rIngID= {R.id.recipe_display_I01,R.id.
    recipe_display_I02,R.id.recipe_display_I03,
197                 R.id.recipe_display_I04,R.id.recipe_display_I05,R.id.
    recipe_display_I06,
198                 R.id.recipe_display_I07,R.id.recipe_display_I08,R.id.
    recipe_display_I09,
199                 R.id.recipe_display_I10,R.id.recipe_display_I11,R.id.
    recipe_display_I12,
200                 R.id.recipe_display_I13,R.id.recipe_display_I14,R.id.
    recipe_display_I15};
201         mDIngText= new TextView[mRecipe.getNbIngMax()];
202         for(int i=0;i<mRecipe.getNbIngMax();i++) {
203             mDIngText[i] = (TextView) v.findViewById(rIngID[i]);
204         }
205         final int[] rID= {R.id.recipe_display_S1,R.id.
    recipe_display_S2,R.id.recipe_display_S3,
206                 R.id.recipe_display_S4,R.id.recipe_display_S5,R.id.
    recipe_display_S6,
207                 R.id.recipe_display_S7,R.id.recipe_display_S8,R.id.
    recipe_display_S9};
208         mDStepText=new TextView[mRecipe.getNbStepMax()];
209         for(int i=0;i<mRecipe.getNbStepMax();i++) {
210             mDStepText[i] = (TextView) v.findViewById(rID[i]);
211         }
212         final int[] rComID= {R.id.recipe_display_C01,R.id.
    recipe_display_C02,R.id.recipe_display_C03,
213                 R.id.recipe_display_C04,R.id.recipe_display_C05,R.id.
    recipe_display_C06,
214                 R.id.recipe_display_C07,R.id.recipe_display_C08,R.id.
    recipe_display_C09,
215                 R.id.recipe_display_C10,R.id.recipe_display_C11,R.id.
    recipe_display_C12,
216                 R.id.recipe_display_C13,R.id.recipe_display_C14,R.id.
    recipe_display_C15,
217                 R.id.recipe_display_C16,R.id.recipe_display_C17,R.id.
    recipe_display_C18,
218                 R.id.recipe_display_C19,R.id.recipe_display_C20};
219         mDComText=new TextView[mRecipe.getNbComMax()];
220         for(int i=0;i<mRecipe.getNbComMax();i++) {
221             mDComText[i] = (TextView) v.findViewById(rComID[i]);
222         }
223         updateUI();
224         mDNexComment=(EditText) v.findViewById(R.id.
    recipe_display_enter_comment);
225         mDNexComment.setText("");
226         mDNexComment.setHint(getString(R.string.P2C_hint));
227         //todo test comment versus pattern
228         mDNexComment.addTextChangedListener(new TextWatcher() {
229             @Override
230             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) {
231
```

```
232             }
233
234             @Override
235             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
236                 newcomment=s.toString();
237             }
238
239             @Override
240             public void afterTextChanged(Editable s) {
241
242             }
243         });
244         mDEnterComment=(ImageView) v.findViewById(R.id.
    recipe_img_enter);
245         mDEnterComment.setOnClickListener(new View.OnClickListener
    () {
246             @Override
247             public void onClick(View v) {
248                 mRecipe.addComment(new Comment(newcomment, mSession.
    getUser()));
249                 mRecipe.updateTS(AsynCallFlag.NEWCOMMENT,true);
250                 mDNexComment.setText("...");
251                 updateUI();
252             }
253         });
254         return v;
255     }
256     private void updatePhotoView(){
257         if (mPhotoFile==null || !mPhotoFile.exists()){
258             mDPhotoView.setImageDrawable(getResources().getDrawable(R
    .drawable.ic_recipe_camera));
259         } else {
260             Bitmap bitmap=PictureUtils.getScaledBitmap(mPhotoFile.
    getPath(), getActivity());
261             mDPhotoView.setImageBitmap(bitmap);
262         }
263     }
264     private void updateUI(){
265         String s,s2;
266         User u=mRecipe.getOwner();
267         int ingMax=mRecipe.getNbIngMax();
268         int stepMax=mRecipe.getNbStepMax();
269         int comMax=mRecipe.getNbComMax();
270         int NbCom=mRecipe.getComments().size();
271         int gone=View.GONE, visible=View.VISIBLE;
272         DecimalFormat df = new DecimalFormat("#.#");
273         mDTitleText.setText(mRecipe.getTitle());
274         mDRatingBar.setRating((float) mRecipe.getNoteAvg());
275         mDRatingBarText.setText(df.format(mRecipe.getNoteAvg())+"  ("
    + mRecipe.getNotes().size()+")");
276         s=getString(R.string.P2U_txt,u.getName(),u.getFamily());
277         int idiff= RecipeDifficulty.getIndex(mRecipe.getDifficulty
    ());
278         String[] stringArrayDiff = getResources().getStringArray(R.
    array.recipe_difficulty_array);
279         mDDifficulty.setText(stringArrayDiff[idiff]);
```

```
280         int ityp= RecipeType.getIndex(mRecipe.getType());
281         String[] stringArrayTyp = getResources().getStringArray(R.
    array.recipe_type_array);
282         mDType.setText(stringArrayTyp[ityp]);
283         mSunIcon.setImageResource((mRecipe.IsSummer()) ? R.drawable.
    ic_recipe_sun : R.drawable.ic_recipe_sun_disabled);
284         mIceIcon.setImageResource((mRecipe.IsWinter()) ? R.drawable.
    ic_recipe_ice : R.drawable.ic_recipe_ice_disabled);
285         mDAuthorText.setText(s);
286         // source
287         s=mRecipe.getSource();
288         s2=mRecipe.getSource_url_name();
289         if (s.equals("")) mDSourceText.setVisibility(gone);
290         else mDSourceText.setText(s);
291         if ((s2.equals(DEFAULT_URL)) || (s2.equals(""))) {
292             mDSourceUrl.setVisibility(gone);
293             if (s.equals(""))  mDSourceTitle.setVisibility(gone);
294         }
295         else mDSourceUrl.setText(s2);
296         // ingredients
297         s=getString(R.string.P2IT, ""+mRecipe.getNbPers());
298         if (mRecipe.getNbIng()>0) mDIngTitle.setText(s); else
    mDIngTitle.setVisibility(gone);
299         for(int i=0;i<ingMax;i++){
300             if (mIngNb>0){mDIngText[i].setText("- "+mRecipe.
    getIngredient(i+1));}
301             if (i>=0){mDIngText[i].setVisibility((mIngNb>i)? visible:
    gone);}
302         }
303         // steps
304         if (mRecipe.getNbStep()==0)  mDStepTitle.setVisibility(gone);
305         for(int i=0;i<stepMax;i++){
306             if (mStepNb>0){mDStepText[i].setText((i+1)+". "+mRecipe.
    getStep(i+1));}
307             if (i>=0){mDStepText[i].setVisibility((mStepNb>i)?
    visible:gone);}
308         }
309         // comments
310         s=getString(R.string.P2CT, ""+mRecipe.getComments().size());
311         mDComTitle.setText(s);
312         for(int i=0;i<comMax;i++){
313             if (NbCom>0){
314                 if (i<NbCom) {mDComText[i].setText("- "+mRecipe.
    getComment(NbCom-i-1).toTxt());}
315                 else {mDComText[i].setText("");}
316             }
317             if (i>=0){mDComText[i].setVisibility((NbCom>i)? visible:
    gone);}
318         }
319     }
320     private Boolean IsLenOK(String s, int min, int max){
321         int l=s.length();
322         return ((l>min)&&(l<max));
323     }
324     private String getRecipeReport(){
325         String report;
326         Integer iplus;
```

```
327         report =getString(R.string.P2RE_title, mRecipe.getTitle())+"\
    n";
328         report +=getString(R.string.P2RE_user,mRecipe.getOwner().
    getNameComplete())+"\n";
329         if(!mRecipe.getSource().equals("")){
330             report +=  getString(R.string.P2RE_src, mRecipe.getSource
    ())+"\n";
331         }
332         if(!mRecipe.getSource_url_name().equals("")){
333             report +=  getString(R.string.P2RE_url, mRecipe.
    getSource_url_name())+"\n";
334         }
335         if (mRecipe.getNbIng()>0){
336             report +=getString(R.string.P1R_ing)+"\n";
337             for(int i=0;i<mRecipe.getNbIng();i++){
338                 report += getString(R.string.P2RE_ing, mRecipe.
    getIngredient(i+1))+"\n";
339             }
340         }
341         if (mRecipe.getNbStep()>0) {
342             report +=getString(R.string.P1R_step)+"\n";
343             for (int i = 0; i < mRecipe.getNbStep(); i++) {
344                 iplus = i + 1;
345                 report += getString(R.string.P2RE_step, iplus + "",
346                         mRecipe.getStep(i + 1)) + "\n";
347             }
348         }
349         report +=getString(R.string.P2RE_end);
350         return report;
351     }
352
353
    /*************************************************************************
    ***********************
354      *                                ASYNC
                                                                *
355      *************************************************************************
    ***************************/
356
357
358     class sendMailAsync extends AsyncTask<Void, Void, Boolean> {
359         private static final String PHP204 = "return204.php";
360         private static final String MYSQLDATEFORMAT="yyyy-MM-dd HH:mm
    :ss";
361         private static final String PHPSENDMAIL = "sendmail.php";
362         @Override
363         protected void onPreExecute() {
364             super.onPreExecute();
365             Toast.makeText(getActivity(),getString(R.string.P4_start
    ), Toast.LENGTH_SHORT).show();
366         }
367
368         @Override
369         protected void onPostExecute(Boolean b) {
370             super.onPostExecute(b);
371             if (b) {
372                 Toast.makeText(getActivity(), getString(R.string.
```

```
372 P4_OK,mToMember,mToFamily), Toast.LENGTH_LONG).show();
373             }
374             else {
375                 Toast.makeText(getActivity(), getString(R.string.
    P4_fail), Toast.LENGTH_LONG).show();
376             }
377          }
378
379         @Override
380         protected Boolean doInBackground(Void... voids) {
381             NetworkUtils mNetUtils = new NetworkUtils(getContext());
382             if (!test204()) {
383                 return false;
384             }
385             HashMap<String, String> data = new HashMap<>();
386             data.put("idrecipe", mRecipe.getId().toString());
387             if ((mToFamily==null)||(mToFamily.length()<5)) return
    false;
388             data.put("family",mToFamily.trim());
389             if ((mToMember==null)||(mToMember.length()<5)) return
    false;
390             data.put("membre", mToMember.trim());
391             data.put("idfrom", mSession.getUser().getId().toString
    ());
392             if (mToMessage==null) return false;
393             data.put("message", mToMessage.trim());
394             String result = mNetUtils.sendPostRequestJson(mSession.
    getURLPath() + PHPSENDMAIL, data);
395             if (result==null) return false;
396             if (!result.trim().equals("1")) {
397                 return false;}
398             return true;
399         }
400
401         private Boolean test204() {
402             try {
403                 URL url = new URL(mSession.getURLPath() + PHP204);
404                 HttpURLConnection conn = (HttpURLConnection) url.
    openConnection();
405                 conn.setConnectTimeout(mSession.getConnectTimeout());
406                 conn.setReadTimeout(mSession.getReadTimeout());
407                 conn.setRequestMethod("HEAD");
408                 InputStream in = conn.getInputStream();
409                 int status = conn.getResponseCode();
410                 in.close();
411                 conn.disconnect();
412                 return (status == HttpURLConnection.HTTP_NO_CONTENT);
413             } catch (Exception e) {
414                 return false;
415             }
416         }
417
418     }
419
420 }
```

```java
package com.fdx.cookbook;

import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v7.app.AppCompatActivity;

import com.fdx.cookbook.R;

public abstract class SingleFragmentActivity extends AppCompatActivity {
    protected abstract Fragment createFragment();
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_fragment);
        FragmentManager fm = getSupportFragmentManager();
        Fragment fragment = fm.findFragmentById(R.id.fragment_container);
        if (fragment == null) {
            fragment = createFragment();
            fm.beginTransaction()
                    .add(R.id.fragment_container, fragment)
                    .commit();
        }
    }
}

```

```java
package com.fdx.cookbook;

import android.content.Context;
import android.content.Intent;
import android.support.v4.app.Fragment;

public class RecipeCommunityActivity extends SingleFragmentActivity  {
    public static Intent newIntent(Context packageContexte) {
        Intent intent=new Intent(packageContexte,
RecipeCommunityActivity.class);
        return intent;
    }
    @Override
    protected Fragment createFragment() {
        return RecipeCommunityFragment.newInstance();
    }
}

```

```java
package com.fdx.cookbook;

import android.content.DialogInterface;
import android.graphics.Bitmap;
import android.os.AsyncTask;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.v4.app.Fragment;
import android.support.v7.app.AlertDialog;
import android.support.v7.widget.GridLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.util.DisplayMetrics;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;

import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.URL;
import java.text.DecimalFormat;
import java.util.HashMap;
import java.util.List;
import java.util.regex.Pattern;

public class RecipeCommunityFragment extends Fragment {
    private RecyclerView mRecipeRecyclerView;
    private RecipeAdapter mAdapter;
    private SessionInfo mSession;
    private static final Integer RECENT=1;
    private static final Integer POPULAR=2;
    private static final Integer BESTNOTE=3;
    private Integer mSelectType;
    private MenuItem mMenuR;
    private MenuItem mMenuP;
    private MenuItem mMenuN;
    private CookBooksShort mCookbookShort;
    private getShortRecipesFromCommunity getAsyncShorties;
    private static final String TAG = "CB_ComFrag";
    private static final String REGEX_FAMILY="[\\p{Punct}\\w\\p{
javaLowerCase}\\p{javaUpperCase}()\\p{Space}]*";

    public static RecipeCommunityFragment newInstance() {
        RecipeCommunityFragment fragment=new RecipeCommunityFragment
();
        return fragment;
    }

    @Override
```

```
 56     public void onCreate(Bundle savedInstanceState) {
 57         super.onCreate(savedInstanceState);
 58         setHasOptionsMenu(true);
 59         mSession= SessionInfo.get(getActivity());
 60         mCookbookShort=CookBooksShort.get(getActivity());
 61         mSelectType=mCookbookShort.getSelectType();
 62         if (mCookbookShort.isNew()){
 63             getAsyncShorties=new getShortRecipesFromCommunity();
 64             getAsyncShorties.execute(20);
 65         }
 66     }
 67 
 68     @Override
 69     public void onCreateOptionsMenu(Menu menu, MenuInflater inflater
    ) {
 70         super.onCreateOptionsMenu(menu, inflater);
 71         inflater.inflate(R.menu.fragment_menu_com, menu);
 72         mMenuR = menu.findItem(R.id.com_recent);
 73         mMenuN = menu.findItem(R.id.com_bestnote);
 74         mMenuP = menu.findItem(R.id.com_popular);
 75         mMenuR.setEnabled(mSelectType!=RECENT);
 76         mMenuN.setEnabled(mSelectType!=BESTNOTE);
 77         mMenuP.setEnabled(mSelectType!=POPULAR);
 78     }
 79 
 80     public boolean onOptionsItemSelected(MenuItem item) {
 81         if (mCookbookShort.isDownloading()) return true;
 82         switch (item.getItemId()) {
 83             case R.id.com_recent:
 84                 mCookbookShort.setSelectType(RECENT);
 85                 break;
 86             case R.id.com_popular:
 87                 mCookbookShort.setSelectType(POPULAR);
 88                 break;
 89             case R.id.com_bestnote:
 90                 mCookbookShort.setSelectType(BESTNOTE);
 91                 break;
 92             default:
 93                 return super.onOptionsItemSelected(item);
 94         }
 95         mSelectType=mCookbookShort.getSelectType();
 96         mMenuR.setEnabled(mSelectType!=RECENT);
 97         mMenuN.setEnabled(mSelectType!=BESTNOTE);
 98         mMenuP.setEnabled(mSelectType!=POPULAR);
 99         getAsyncShorties=new getShortRecipesFromCommunity();
100         getAsyncShorties.execute(20);
101         return true;
102     }
103 
104     @Override
105     public View onCreateView(LayoutInflater inflater, ViewGroup
    container,
106                              Bundle savedInstanceState) {
107         View view = inflater.inflate(R.layout.
    fragment_recipe_community, container, false);
108         mRecipeRecyclerView = (RecyclerView) view.findViewById(R.id.
    recipe_recycler_view);
```

```
109         DisplayMetrics displayMetrics = new DisplayMetrics();
110         ((RecipeCommunityActivity) getContext()).getWindowManager().
    getDefaultDisplay().getMetrics(displayMetrics);
111         int ncol = (Integer) displayMetrics.widthPixels/300;
112         if (ncol<3) ncol=3;
113         mRecipeRecyclerView.setLayoutManager(new GridLayoutManager(
    getActivity(),ncol));
114         updateUI();
115         return view;
116     }
117     @Override
118     public void onPause(){
119         super.onPause();
120     }
121
122     private void updateUI() {
123         List<Recipe> recipes=mCookbookShort.getRecipes();
124         if (mAdapter==null){
125             mAdapter=new RecipeCommunityFragment.RecipeAdapter(
    recipes);
126             mRecipeRecyclerView.setAdapter(mAdapter);
127         } else {
128             mAdapter.setRecipes(recipes);
129             mAdapter.notifyDataSetChanged();
130         }
131     }
132     // ----------------------------------RECIPE HOLDER
     ------------------------
133     private class RecipeHolder extends RecyclerView.ViewHolder
134             implements View.OnClickListener {
135         private TextView mTitleTextView;
136         private TextView mName;
137         private TextView mFamily;
138         private ImageView mPhotoView;
139         private Recipe mRecipe;
140
141         public RecipeHolder(LayoutInflater inflater,ViewGroup parent
    ){
142             super(inflater.inflate(R.layout.list_item_grid, parent,
    false));
143             itemView.setOnClickListener(this);
144             mTitleTextView= (TextView) itemView.findViewById(R.id.
    com_title);
145             mName=(TextView) itemView.findViewById(R.id.com_name);
146             mFamily=(TextView) itemView.findViewById(R.id.com_family
    );
147             mPhotoView=(ImageView) itemView.findViewById(R.id.
    com_photo);
148             mPhotoView.setOnClickListener(new View.OnClickListener
    () {
149                 @Override
150                 public void onClick(View v) {
151                     LinearLayout layout = new LinearLayout(getContext
    ());
152                     layout.setOrientation(LinearLayout.VERTICAL);
153                     final TextView requestRecipeBox = new TextView(
    getContext());
```

```
154                     requestRecipeBox.setText("   "+getString(R.string
    .P5DT, mRecipe.getTitle()));
155                     layout.addView(requestRecipeBox);
156                     final TextView toRecipeBox = new TextView(
    getContext());
157                     toRecipeBox.setText("   "+getString(R.string.P5DN
    , mRecipe.getOwner().getNameComplete()));
158                     layout.addView(toRecipeBox);
159                     final EditText messageBox = new EditText(
    getContext());
160                     messageBox.setHint(getString(R.string.P5DM));
161                     layout.addView(messageBox);
162                     AlertDialog.Builder builder = new AlertDialog.
    Builder(getContext());
163                     builder.setTitle(getString(R.string.P5DTT));
164                     builder.setView(layout);
165                     builder.setPositiveButton("OK", new
    DialogInterface.OnClickListener() {
166                         @Override
167                         public void onClick(DialogInterface dialog,
    int which) {
168                             String id=mRecipe.getId().toString();
169                             String mToMessage = messageBox.getText().
    toString();
170                             Boolean b=(Pattern.matches(REGEX_FAMILY,
    mToMessage));
171                             if (b) {
172                                 sendRequest sendrequest = new
    sendRequest();
173                                 sendrequest.execute(id,mToMessage);
174                                 deBugShow("Launch request :"+
    mToMessage+" / "+id);
175                             } else {
176                                 Toast.makeText(getActivity(),
    getString(R.string.P5DMErr), Toast.LENGTH_SHORT).show();
177                             }
178                         }
179                     });
180                     builder.setNegativeButton("Cancel", new
    DialogInterface.OnClickListener() {
181                         @Override
182                         public void onClick(DialogInterface dialog,
    int which) {
183                             dialog.cancel();
184                         }
185                     });
186                     builder.show();
187                 }
188             });
189
190         }
191         public void bind(Recipe recipe){
192             mRecipe=recipe;
193             mTitleTextView.setText(mRecipe.getTitle());
194             mName.setText(mRecipe.getOwner().getName());
195             mFamily.setText("@"+mRecipe.getOwner().getFamily());
196             Bitmap bm=mRecipe.getImage();
```

```
197             if (bm!=null) mPhotoView.setImageBitmap(bm);
198             else mPhotoView.setImageDrawable(getResources().
    getDrawable(R.drawable.ic_recipe_see));
199         }
200 
201         @Override
202         public void onClick(View v){    }
203     }
204     // ----------------------------------ADAPTER
    ------------------------
205     private class RecipeAdapter extends RecyclerView.Adapter<
    RecipeCommunityFragment.RecipeHolder> {
206         private List<Recipe> mRecipes;
207         public RecipeAdapter(List<Recipe> recipes){
208             mRecipes=recipes;
209         }
210 
211         @NonNull
212         @Override
213         public RecipeCommunityFragment.RecipeHolder
    onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
214             LayoutInflater layoutInflater=LayoutInflater.from(
    getActivity());
215             return new RecipeCommunityFragment.RecipeHolder(
    layoutInflater, parent);
216         }
217 
218         @Override
219         public void onBindViewHolder(@NonNull RecipeCommunityFragment
    .RecipeHolder recipeHolder, int i) {
220             Recipe recipe=mRecipes.get(i);
221             recipeHolder.bind(recipe);
222         }
223 
224         @Override
225         public int getItemCount() {
226             return mRecipes.size();
227         }
228 
229         public void setRecipes(List<Recipe> recipes){
230             mRecipes=recipes;
231         }
232     }
233 
234 
    /**********************************************************************
    **********************
235      *                                ASYNC
                                                                *
236      **********************************************************************
    **************************/
237 
238     class getShortRecipesFromCommunity extends AsyncTask<Integer,
    Integer, Boolean> {
239         private static final String PHP204 = "return204.php";
240         private static final String PHPGETSHORTRECIPES = "
    getcommunitynews.php";
```

```
241
242         @Override
243         protected void onPreExecute() {
244             super.onPreExecute();
245             mCookbookShort.getDLStarted();
246             mSelectType=mCookbookShort.getSelectType();
247         }
248
249         @Override
250         protected void onPostExecute(Boolean b) {
251             super.onPostExecute(b);
252             mCookbookShort.setDLCompleted();
253         }
254
255         @Override
256         protected Boolean doInBackground(Integer ... integers) {
257             Integer ntoDownload=integers[0];
258             if (!test204()) {deBugShow("204!"); return false;}
259             // download texts
260             String s=getStringRecipesFromServer(0,ntoDownload,
    mSelectType, false);
261             if (s.equals("")) {deBugShow("Echec query 1"); return
    false;}
262             if (!fillInitialTable(s)) {deBugShow("Echec parse >"+s+
    "<"); return false;}
263             publishProgress(1);
264             // download photo i
265             for(Integer i=0;i<mCookbookShort.getsize();i++) {
266                 if(isCancelled()) {deBugShow("Interrupted)"); return
    false;}
267                 s = getStringRecipesFromServer(i, 1, mSelectType,
    true);
268                 if (s.equals("")) {
269                     deBugShow("Echec query "+i);
270                     break;
271                 }
272                 if (updatePhotoOfRecipe(s)) publishProgress(i+2);
273             }
274             return true;
275         }
276
277         @Override
278         protected void onProgressUpdate(Integer ... values){
279             updateUI();
280         }
281
282         private String getStringRecipesFromServer(Integer start,
    Integer count, Integer type, Boolean withphoto){
283             HashMap<String, String> data = new HashMap<>();
284             data.put("iduser", mSession.getUser().getId().toString
    ());
285             String comtype="";
286             if (type==RECENT) comtype="RECENT";
287             if (type==POPULAR) comtype="POPULAR";
288             if (type==BESTNOTE) comtype="BESTNOTE";
289             data.put("comtype", comtype);
290             DecimalFormat formatter = new DecimalFormat("00");
```

```
291             data.put("start", formatter.format( start));
292             data.put("count", formatter.format(count));
293             data.put("comparam", (withphoto ? "WP":"WOP"));
294             NetworkUtils mNetUtils = new NetworkUtils(getContext());
295             String result = mNetUtils.sendPostRequestJson(mSession.
    getURLPath() + PHPGETSHORTRECIPES, data);
296             if (result==null) return "";
297             return result;
298         }
299 
300         private boolean fillInitialTable(String s){
301             NetworkUtils mNetUtils = new NetworkUtils(getContext());
302             List<Recipe> recipes=mNetUtils.parseRecipesOfCommunity(s,
    false);
303             if ((recipes!=null)&&(!recipes.isEmpty())){
304                 mCookbookShort.clear();
305                 mCookbookShort.fill(recipes);
306                 return true;
307             } else return false;
308         }
309 
310         private boolean updatePhotoOfRecipe(String s){
311             NetworkUtils mNetUtils = new NetworkUtils(getContext());
312             List<Recipe> recipes=mNetUtils.parseRecipesOfCommunity(s,
    true);
313             if ((recipes!=null)&&(!recipes.isEmpty())){
314                 mCookbookShort.updatePhoto(recipes.get(0));
315                 return true;
316             } return false;
317         }
318 
319 
320         private Boolean test204() {
321             try {
322                 URL url = new URL(mSession.getURLPath() + PHP204);
323                 HttpURLConnection conn = (HttpURLConnection) url.
    openConnection();
324                 conn.setConnectTimeout(mSession.getConnectTimeout());
325                 conn.setReadTimeout(mSession.getReadTimeout());
326                 conn.setRequestMethod("HEAD");
327                 InputStream in = conn.getInputStream();
328                 int status = conn.getResponseCode();
329                 in.close();
330                 conn.disconnect();
331                 return (status == HttpURLConnection.HTTP_NO_CONTENT);
332             } catch (Exception e) {
333                 deBugShow("Test 204 : " + e);
334                 return false;
335             }
336         }
337 
338     }
339 
340 
    /*************************************************************************
    **********************
341      *                                ASYNC
```

```
341                                                                     *
342      ************************************************************************
    ***************************/
343
344
345      class sendRequest extends AsyncTask<String, Void, Boolean> {
346          private static final String PHP204 = "return204.php";
347          private static final String PHPREQ = "newrequest.php";
348          private String mTo;
349          @Override
350          protected void onPreExecute() {
351              super.onPreExecute();
352          }
353
354          @Override
355          protected void onPostExecute(Boolean b) {
356              super.onPostExecute(b);
357              if (b) {
358                  Toast.makeText(getActivity(), getString(R.string.
    P5DMOk), Toast.LENGTH_SHORT).show();
359              }
360              else {
361                  Toast.makeText(getActivity(), getString(R.string.
    P4_fail), Toast.LENGTH_LONG).show();
362              }
363          }
364
365          @Override
366          protected Boolean doInBackground(String... strings) {
367              NetworkUtils mNetUtils = new NetworkUtils(getContext());
368              if (!test204()) {
369                  return false;
370              }
371              String message= strings[1];
372              String recipeid=strings[0];
373              String idfrom=mSession.getUser().getId().toString();
374              HashMap<String, String> data = new HashMap<>();
375              data.put("idrecipe", recipeid);
376              data.put("idfrom", idfrom);
377              data.put("message", message);
378              String result = mNetUtils.sendPostRequestJson(mSession.
    getURLPath() + PHPREQ, data);
379              if (result.equals("1")) return true;
380              return false;
381          }
382
383          private Boolean test204() {
384              try {
385                  URL url = new URL(mSession.getURLPath() + PHP204);
386                  HttpURLConnection conn = (HttpURLConnection) url.
    openConnection();
387                  conn.setConnectTimeout(mSession.getConnectTimeout());
388                  conn.setReadTimeout(mSession.getReadTimeout());
389                  conn.setRequestMethod("HEAD");
390                  InputStream in = conn.getInputStream();
391                  int status = conn.getResponseCode();
392                  in.close();
```

```
393                 conn.disconnect();
394                 return (status == HttpURLConnection.HTTP_NO_CONTENT);
395             } catch (Exception e) {
396                 deBugShow("Test 204 : " + e);
397                 return false;
398             }
399         }
400 
401     }
402 
403     private void deBugShow(String s){
404         //Log.d(TAG, s);
405     }
406 
407 }
408 
```

```java
package com.fdx.cookbook;

import android.content.Context;
import android.content.Intent;
import android.support.v4.app.Fragment;

public class RecipeMailDisplayActivity extends SingleFragmentActivity {
    //private static final String EXTRA_RECIPE_ID="com.fdx.cookbook.recipe_id";
    public static Intent newIntent(Context packageContexte) {
        Intent intent=new Intent(packageContexte, RecipeMailDisplayActivity.class);
        return intent;
    }

    @Override
    protected Fragment createFragment() {
        return RecipeMailDisplayFragment.newInstance();
    }
}

```

```java
package com.fdx.cookbook;

import android.graphics.Bitmap;
import android.os.AsyncTask;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.v4.app.Fragment;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import org.json.JSONArray;
import org.json.JSONObject;

import java.io.File;
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

public class RecipeMailDisplayFragment extends Fragment {
    private RecyclerView mRecipeRecyclerView;
    private RecipeAdapter mAdapter;
    private SessionInfo mSession;
    private ArrayList<MailCard> mMailCards;
    private static final String TAG = "CB_R.MailDisplayFrag";

    public static RecipeMailDisplayFragment newInstance() {
        RecipeMailDisplayFragment fragment=new
RecipeMailDisplayFragment();
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setHasOptionsMenu(true);
        mSession= SessionInfo.get(getActivity());
        mMailCards=new ArrayList<>();
        List<Recipe> recipes =CookBook.get(getActivity()).getRecipes
();
        for (Recipe r:recipes){
            if (r.IsMessage()) mMailCards.add(new MailCard(r));
        }
        getRequests gR=new getRequests();
        gR.execute(0);
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
container,
```

```
 55                                  Bundle savedInstanceState) {
 56         View view = inflater.inflate(R.layout.fragment_recipe_list,
    container, false);
 57         mRecipeRecyclerView = (RecyclerView) view.findViewById(R.id.
    recipe_recycler_view);
 58         mRecipeRecyclerView.setLayoutManager(new LinearLayoutManager(
    getActivity()));
 59         updateUI();
 60         return view;
 61     }
 62
 63     @Override
 64     public void onResume(){
 65         super.onResume();
 66         updateUI();
 67     }
 68
 69     private void updateUI() {
 70         List<MailCard> mailcards=new ArrayList<>();
 71         for(MailCard mc:mMailCards){
 72             if (mc.isSubmitted()) mailcards.add(mc);
 73         }
 74         if (mailcards.isEmpty()){
 75             mSession.setIsRecipeRequest(false);
 76             //Toast.makeText(getContext(), getString(R.string.P4M0),
    Toast.LENGTH_SHORT ).show();
 77         }
 78         if (mAdapter==null){
 79             mAdapter=new RecipeMailDisplayFragment.RecipeAdapter(
    mailcards);
 80             mRecipeRecyclerView.setAdapter(mAdapter);
 81         } else {
 82             mAdapter.setRecipes(mailcards);
 83             mAdapter.notifyDataSetChanged();
 84         }
 85     }
 86 // ----------------------------------RECIPE HOLDER
     ------------------------
 87     private class RecipeHolder extends RecyclerView.ViewHolder
 88             implements View.OnClickListener {
 89         private TextView mTitleTextView;
 90         private TextView mMessage;
 91         private TextView mFrom;
 92         private ImageView mPhotoView;
 93         private ImageView mDelete;
 94         private ImageView mAdd;
 95         private File mPhotoFile;
 96         private Recipe mRecipe;
 97         private MailCard mMailCard;
 98
 99         public RecipeHolder(LayoutInflater inflater,ViewGroup parent
    ){
100             super(inflater.inflate(R.layout.list_item_mail_display,
    parent, false));
101             itemView.setOnClickListener(this);
102             mTitleTextView= (TextView) itemView.findViewById(R.id.
    recipe_MD_title);
```

```
103             mMessage=(TextView) itemView.findViewById(R.id.
    mail_display_message);
104             mFrom=(TextView) itemView.findViewById(R.id.
    mail_display_author);
105             mDelete=(ImageView) itemView.findViewById(R.id.
    mail_display_delete);
106             mAdd=(ImageView) itemView.findViewById(R.id.
    mail_display_add);
107             mPhotoView=(ImageView) itemView.findViewById(R.id.
    recipe_MD_photo);
108             mPhotoFile=CookBook.get(getActivity()).getPhotoFile(
    mRecipe);
109
110             mDelete.setOnClickListener(new View.OnClickListener() {
111                 @Override
112                 public void onClick(View v) {
113                     if (mMailCard.isReceived()){
114                         deBugShow("Onclick del received"+mRecipe.
    getTitle());
115                         mRecipe.setStatus(StatusRecipe.Deleted);
116                         CookBook cookbook=CookBook.get(getActivity
    ());
117                         cookbook.updateRecipe(mRecipe);
118                         mMailCard.setRefused();
119                         updateUI();
120                     }
121                     if (mMailCard.isRequest()){
122                         deBugShow("Onclick del request :"+mMailCard.
    getRequestId());
123                         mMailCard.setMessage(getString(R.string.
    P5MESNO));
124                         updateRequest uR=new updateRequest();
125                         uR.execute(mMailCard.getRequestId(),1);
126                     }
127                 }
128             });
129             mAdd.setOnClickListener(new View.OnClickListener() {
130                 @Override
131                 public void onClick(View v) {
132                     if (mMailCard.isReceived()){
133                         deBugShow("Onclick add received"+mRecipe.
    getTitle());
134                         mRecipe.setStatus(StatusRecipe.Visible);
135                         CookBook cookbook=CookBook.get(getActivity
    ());
136                         cookbook.updateRecipe(mRecipe);
137                         mMailCard.setAccepted();
138                         mSession.setIsRecipeRequest(false);
139                         updateUI();
140                     }
141                     if (mMailCard.isRequest()){
142                         deBugShow("Onclick add request : "+mMailCard.
    getRequestId());
143                         mMailCard.setMessage(getString(R.string.P5MES
    ));
144                         mMailCard.setAccepted();
145                         updateRequest uR=new updateRequest();
```

```
146                         uR.execute(mMailCard.getRequestId(),0);
147                     }
148                 }
149             });
150 
151         }
152 
153         public void bind(MailCard mc){
154             mMailCard=mc;
155             CookBook cookbook=CookBook.get(getActivity());
156             mTitleTextView.setText(mc.getTitle());
157             mMessage.setText(mc.getMessage());
158             if (mc.isReceived()) mFrom.setText(getString(R.string.
    P4MES_IN,mc.getUser().getNameComplete()));
159             if (mc.isRequest()) mFrom.setText(getString(R.string.
    P4MES_OUT,mc.getUser().getNameComplete()));
160             mRecipe=cookbook.getRecipe(mc.getRecipeId());
161             if (mRecipe!=null) mPhotoFile=CookBook.get(getActivity
    ()).getPhotoFile(mRecipe);
162             if (mPhotoFile==null || !mPhotoFile.exists()){
163                 mPhotoView.setImageDrawable(getResources().
    getDrawable(R.drawable.ic_recipe_see));
164             } else {
165                 Bitmap bitmap=PictureUtils.getScaledBitmap(mPhotoFile
    .getPath(), getActivity());
166                 mPhotoView.setImageBitmap(bitmap);
167             }
168         }
169     @Override
170     public void onClick(View v){    }
171     }
172     // ----------------------------------ADAPTER
    ------------------------
173     private class RecipeAdapter extends RecyclerView.Adapter<
    RecipeMailDisplayFragment.RecipeHolder> {
174         private List<MailCard> maMailCards;
175         public RecipeAdapter(List<MailCard> mailcards){
176             maMailCards=mailcards;
177         }
178 
179         @NonNull
180         @Override
181         public RecipeMailDisplayFragment.RecipeHolder
    onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
182             LayoutInflater layoutInflater=LayoutInflater.from(
    getActivity());
183             return new RecipeMailDisplayFragment.RecipeHolder(
    layoutInflater, parent);
184         }
185 
186         @Override
187         public void onBindViewHolder(@NonNull
    RecipeMailDisplayFragment.RecipeHolder recipeHolder, int i) {
188             MailCard mc=maMailCards.get(i);
189             recipeHolder.bind(mc);
190         }
191 
```

```
192         @Override
193         public int getItemCount() {
194             return maMailCards.size();
195         }
196 
197         public void setRecipes(List<MailCard> mailcards){
198             maMailCards=mailcards;
199         }
200     }
201 
202 
    /*************************************************************************
    **********************
203      *                                ASYNC
                                                                *
204      *************************************************************************
    ***************************/
205 
206     class getRequests extends AsyncTask<Integer, Integer, Boolean> {
207         private static final String PHPGETREQUESTS = "getrequests.php
    ";
208 
209         @Override
210         protected void onPreExecute() {
211             super.onPreExecute();
212         }
213 
214         @Override
215         protected void onPostExecute(Boolean b) {
216             super.onPostExecute(b);
217             if (b) {
218                 updateUI();
219             }
220         }
221 
222         @Override
223         protected Boolean doInBackground(Integer ... integers) {
224             MailCard mc;
225             NetworkUtils netutil=new NetworkUtils(getContext());
226             if (!netutil.test204()) {deBugShow("204!"); return false
    ;}
227             HashMap<String, String> data = new HashMap<>();
228             data.put("iduser", mSession.getUser().getId().toString
    ());
229             String json = netutil.sendPostRequestJson(mSession.
    getURLPath() + PHPGETREQUESTS, data);
230             if (json==null) return false;
231             if (json.equals("")) return false;
232             try {
233                 JSONArray jarr1=new JSONArray(json);
234                 for (int i=0; i<jarr1.length(); i++){
235                     JSONObject obj = jarr1.getJSONObject(i);
236                     mc=netutil.parseObjectRequest(obj);
237                     if (mc==null) {
238                         deBugShow("Error in parsing request");
239                         return false;}
240                     mMailCards.add(mc);
```

```
241                 }
242             } catch (Exception e){
243                 deBugShow("Failure in parsing JSON Array Requests "+e
    );
244                 return false;
245             }
246             return true;
247         }
248     }
249 
250 
    /**************************************************************************
    **********************
251      *                                ASYNC
                                                                *
252      **************************************************************************
    ***************************/
253 
254     class updateRequest extends AsyncTask<Integer, Integer, Integer
    []> {
255         private static final String PHPAORD = "acceptorrefuserequest.
    php";
256 
257         @Override
258         protected void onPreExecute() {
259             super.onPreExecute();
260         }
261 
262         @Override
263         protected void onPostExecute(Integer[] result) {
264             super.onPostExecute(result);
265             String status=(result[1]==0 ? "ACCEPTED":"DENIED");
266             if (result[2]==1) {
267                 deBugShow("Succes mailcard  "+ result[0]+" is "+
    status);
268                 updateUI();
269             }
270             else {
271                 deBugShow("Echec mailcard  "+ result[0]+" could not
    be "+status);
272             }
273         }
274 
275         @Override
276         protected Integer[] doInBackground(Integer ... integers) {
277             Integer pknum=integers[0];
278             Integer j=integers[1];
279             String status=(j==0 ? "ACCEPTED":"DENIED");
280             Integer[] result={pknum,j,0};
281             MailCard mc=findMailCard(pknum);
282             if (mc==null) {deBugShow("Request "+pknum+" not found");
    return result;}
283             NetworkUtils netutil=new NetworkUtils(getContext());
284             if (!netutil.test204()) {deBugShow(" request => 204!");
    return result;}
285             HashMap<String, String> data = new HashMap<>();
286             DecimalFormat formatter = new DecimalFormat("######");
```

```
287             data.put("pknum", formatter.format( pknum));
288             data.put("status", status);
289             data.put("iduser", mSession.getUser().getId().toString
    ());
290             data.put("message", mc.getMessage());
291             String json = netutil.sendPostRequestJson(mSession.
    getURLPath() + PHPAORD, data);
292             if (json.equals("1")){
293                 if (mMailCards.indexOf(mc)==-1){
294                     deBugShow("Can't find the card to be deleted");
295                     return result;
296                 }
297                 mMailCards.remove(mMailCards.indexOf(mc));
298                 result[2]=1;
299             } else {
300                 deBugShow("Request "+pknum+" to accept or deny did
    not work");
301                 return result;}
302             return result;
303         }
304     }
305
306     private MailCard findMailCard(Integer i){
307         for(MailCard m:mMailCards){
308             if (m.getRequestId()==i) return m;
309         }
310         return null;
311     }
312
313     private void deBugShow(String s){
314         //Log.d(TAG, s);
315     }
316
317 }
318
```