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

```java
package com.example.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()));
```

```
55          b=b && (mUser.equals(n.getUser()));
56          return b;
57      }
58  }
59
```

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

```java
package com.example.cookbook;

import android.util.Log;

import java.util.Date;
import java.util.UUID;
import java.util.regex.Pattern;

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(UUID uuid){
        mId =uuid;
        // waiting for user database
        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() {
        return mFamily;
```

```
55      }
56
57      public void setFamily(String family) {
58          mFamily = family;
59      }
60
61      public String getName() {
62          return mName;
63      }
64
65      public void setName(String name) {
66          mName = name;
67      }
68
69      public boolean IsEqualUser(User u){
70          return (this==u);
71      }
72
73      public String getNameComplete() {
74          return mName+" de "+ mFamily;
75      }
76
77      public boolean IsEqual(User r){return (mId==r.getId());}
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.example.cookbook;

import android.util.Log;

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 RecipeDifficulty mDifficulty;
    private String[] mIngredients;
    private StatusRecipe mStatus;   // mettre à jour affichage
    private String mMessage;
    private User mIdFrom;
    private Boolean mTS_recipe;
    private Boolean mTS_photo;
    private Boolean mTS_comment;
    private Boolean mTS_note;

    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";



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

    public Recipe( UUID id){
        mId=id;
```

```
55          mLastUpdateRecipe =new Date();
56          Calendar c=Calendar.getInstance();
57          c.set(2000,0,1,0,0, 0);
58          mLastUpdateRecipe=c.getTime();
59          mLastUpdatePhoto=c.getTime();
60          mSteps = new String[NBSTEP_MAX];
61          mIngredients= new String[NBING_MAX];
62          for(int i=0;i<NBSTEP_MAX;i++){mSteps[i]="";}
63          for(int i=0;i<NBING_MAX;i++){mIngredients[i]="";}
64          mNbPers=4;
65          mSeason=RecipeSeason.ALLYEAR;
66          mDifficulty=RecipeDifficulty.UNDEFINED;
67          mStatus=StatusRecipe.Visible;
68          mComments=new ArrayList<Comment>();
69          mNotes=new ArrayList<Note>();
70          mSource="";
71          mIdFrom=new User(UUID.fromString( "00000000-0000-0000-0000-
    000000000000" ));
72          try {mSource_url=new URL(DEFAULT_URL);
73          } catch (MalformedURLException e) {}
74          mTS_recipe=false;
75          mTS_photo=false;
76          mTS_comment=false;
77          mTS_note=false;
78      }
79
80
81      public UUID getId() {
82          return mId;
83      }
84
85      public User getUserFrom() {
86          return mIdFrom;
87      }
88
89      public void setUserFrom(User idFrom) {
90          mIdFrom = idFrom;
91      }
92
93      public void setId(UUID id) {mId = id; }
94
95      public Date getDate() {
96          return mLastUpdateRecipe;
97      }
98
99      public Date getDatePhoto() {
100         return mLastUpdatePhoto;
101     }
102
103     public String getTitle() {
104         return mTitle;
105     }
106
107     public void setDate(Date date) {mLastUpdateRecipe = date;}
```

```
108
109     public void setDatePhoto(Date date) {mLastUpdatePhoto = date;}
110
111     public void setTitle(String title) {
112         mTitle = title;
113     }
114
115     public int getNbPers() {
116         return mNbPers;
117     }
118
119     public void setNbPers(int nbPers) {
120         mNbPers = nbPers;
121     }
122
123     public String getMessage() {
124         return mMessage;
125     }
126
127     public void setMessage(String message) {
128         mMessage=message;
129     }
130
131     public void updateTS(AsynCallFlag asyn, Boolean b){
132         switch(asyn){
133             case NEWRECIPE:{mTS_recipe=b;return;}
134             case NEWPHOTO:{mTS_photo=b;return;}
135             case NEWRATING:{mTS_note=b;return;}
136             case NEWCOMMENT:{mTS_comment=b;return;}
137         }
138         return;
139     }
140
141     public int getTS(AsynCallFlag asyn){
142         Boolean b=false;
143         switch(asyn){
144             case NEWRECIPE:{b=mTS_recipe; break;}
145             case NEWPHOTO:{b=mTS_photo; break;}
146             case NEWRATING:{b=mTS_note; break;}
147             case NEWCOMMENT:{b=mTS_comment; break;}
148         }
149         return (b ? 1:0);
150     }
151     public Boolean hasChanged(){
152         return (mTS_recipe || mTS_photo || mTS_note || mTS_comment);
153     }
154
155     public String getFlag(){
156         String s=mStatus.toString().substring(0,1);
157         s=s+(mTS_recipe ? "1":"0");
158         s=s+(mTS_photo ? "1":"0");
159         s=s+(mTS_comment ? "1":"0");
160         s=s+(mTS_note ? "1":"0");
161         return s;
```

```
162     }
163
164     // -----------------STEP-------------------
165     public void setStep(Integer i, String step) {
166         if ((i > 0) && (i <= NBSTEP_MAX)) {
167             mSteps[i - 1] = step;
168         }
169     }
170
171     public String getStep(Integer i){
172         if ((i>0)&&(i<=NBSTEP_MAX)) {return mSteps[i-1];}
173         else{ return "";}
174     }
175
176     public int getNbStep(){
177         int j=0;
178         for(int i=NBSTEP_MAX; i>0; i--){
179             if (!mSteps[i-1].isEmpty()) {j=i; break;}}
180         return j;
181     }
182
183     public int getNbStepMax(){return NBSTEP_MAX;}
184
185     //--------------Ingredients--------------------
186     public void setIngredient(Integer i, String ing) {
187         if ((i > 0) && (i <= NBING_MAX)) {
188             mIngredients[i-1] = ing;
189         }
190     }
191
192     public String getIngredient(Integer i){
193         if ((i>0)&&(i<=NBING_MAX)) {return mIngredients[i-1];}
194         else{ return "";}
195     }
196     public int getNbIng(){
197         int j=0;
198         for(int i=NBING_MAX; i>0; i--){
199             if (!mIngredients[i-1].isEmpty()) {j=i; break;}}
200         return j;
201     }
202
203     public int getNbIngMax(){return NBING_MAX;}
204
205     //------------------- Owner ----------------------------------
206     public User getOwner() {
207         return mOwner;
208     }
209
210     public String getOwnerIdString() {
211         return mOwner.getId().toString();
212     }
213
214     public void setOwner(User owner) {
215         mOwner = owner;
```

```
216     }
217
218
219     //------------------- Source ---------------------------------
220     public String getSource() {
221         if (mSource==null) return "";
222         return mSource;
223     }
224     public void setSource(String source) {
225         mSource = source;
226     }
227     public URL getSource_url() {
228         return mSource_url;
229     }
230
231     public String getSource_url_name() {
232         String ret=mSource_url.toString();
233         if (ret.equals("https:") || ret.equals("http:")){ret="";}
234         return ret;
235     }
236
237     public void setSource_url(URL source_url) {
238         mSource_url = source_url;
239     }
240
241     // ------------------------Enum -------------------------m
242
243     public RecipeSeason getSeason() {
244         return mSeason;
245     }
246
247     public void setSeason(RecipeSeason season) {
248         mSeason = season;
249     }
250
251     public boolean IsSummer(){
252         switch (mSeason) {
253             case SUMMER:
254                 return true;
255             case ALLYEAR:
256                 return true;
257             default:
258                 return false;
259         }
260     }
261     public boolean IsWinter(){
262         switch (mSeason) {
263             case WINTER:
264                 return true;
265             case ALLYEAR:
266                 return true;
267             default:
268                 return false;
269         }
```

```
270     }
271
272     public RecipeDifficulty getDifficulty() {
273         return mDifficulty;
274     }
275
276     public void setDifficulty(RecipeDifficulty difficulty) {
277         mDifficulty = difficulty;
278     }
279
280     public StatusRecipe getStatus() {return mStatus;}
281     public void setStatus(StatusRecipe s) {
282         mStatus = s;
283     }
284     public boolean IsVisible(){
285         if (mStatus==StatusRecipe.Visible) {return true;}
286         return false;
287     }
288     public boolean IsMessage(){
289         if (mStatus==StatusRecipe.Submitted) {return true;}
290         return false;
291     }
292     public boolean IsMarkedDeleted(){
293         if (mStatus==StatusRecipe.Deleted) {return true;}
294         return false;
295     }
296
297     //-------------------- photo filename---------------------
298
299     public String getPhotoFilename(){
300         return "IMG"+getId().toString()+".jpg";
301     }
302
303     //---------------------- Arraylist Comments et Notes
    ------------------------
304
305     public void addComment(Comment c){ mComments.add(c);}
306
307     public ArrayList<Comment> getComments() {return mComments;}
308
309     public Comment getComment(int i){
310         if (i<mComments.size()){
311             return mComments.get(i);
312         } else {return null;}
313
314     }
315
316     public int getNbComMax(){return NBCOM_MAX;} // nb max affiché
317
318     public void addNote(Note note){ mNotes.add(note);}
319     public ArrayList<Note> getNotes() {return mNotes;}
320     public Note getNote(int i){
321         if (mNotes==null) return null;
322         if (i<mNotes.size()){ return mNotes.get(i);}
```

```
323         else {return null;}
324     }
325
326     public double getNoteAvg() {
327         mNoteAvg=0;
328         if (mNotes==null) return mNoteAvg;
329         if (mNotes.size()!=0) {
330             for(Note n:mNotes){mNoteAvg+=n.getNote();}
331             mNoteAvg=mNoteAvg/mNotes.size();
332         } else {mNoteAvg=0;}
333         return mNoteAvg;
334     }
335
336     //---------------------- Serialisation ------------------------
337     public String getSerializedComments(){
338         Gson gson = new Gson();
339         return gson.toJson(mComments);
340     }
341
342     public void getCommentsDeserialised(String raw){
343         Gson gson=new Gson();
344         Type listOfNotesObject = new TypeToken<ArrayList<Comment>>()
    {}.getType();
345         mComments=gson.fromJson(raw, listOfNotesObject);
346     }
347     public String getSerializedOwner(){
348         Gson gson = new Gson();
349         return gson.toJson(mOwner);
350     }
351     public String getSerializedFrom(){
352         Gson gson = new Gson();
353         return gson.toJson(mIdFrom);
354     }
355     public void getOwnerDeserialized(String raw){
356         Gson gson=new Gson();
357         mOwner=gson.fromJson(raw, User.class);
358     }
359     public void getFromDeserialized(String raw){
360         Gson gson=new Gson();
361         mIdFrom=gson.fromJson(raw, User.class);
362     }
363
364     public String getSerializedNotes(){
365         Gson gson = new Gson();
366         return gson.toJson(mNotes);
367     }
368
369     public void getNotesDeserialised(String raw){
370         Gson gson=new Gson();
371         Type listOfNotesObject = new TypeToken<ArrayList<Note>>() {}.
    getType();
372         mNotes=gson.fromJson(raw, listOfNotesObject);
373     }
374
```

```
375     // ****************** tests **********************
376     public Boolean hasNotChangedSince(Recipe r){
377         if (!mId.toString().equals(r.getId().toString())) return
    false;
378         if (!mOwner.getId().toString().equals(r.getOwner().getId().
    toString())) return false;
379         if (!mTitle.equals(r.getTitle())) return false;
380         if (!mSource.equals(r.getSource())) return false;
381         if (!mSource_url.equals(r.getSource_url())) return false;
382         if (mNbPers!=r.getNbPers()) return false;
383         for (int i = 0; i < r.getNbStepMax(); i++) {
384             if (!mSteps[i].equals(r.getStep(i+1))) return false;
385         }
386         for (int i = 0; i < r.getNbIngMax(); i++) {
387             if (!mIngredients[i].equals(r.getIngredient(i + 1)))
    return false;
388         }
389         if (!mSeason.toString().equals(r.getSeason().toString()))
    return false;
390         if (!mDifficulty.toString().equals(r.getDifficulty().toString
    ())) return false;
391         return true;
392     }
393
394 }
395
```

```java
package com.example.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;
```

```
55          if (getClass() != o.getClass()) return false;
56          Comment c=(Comment) o;
57          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      public String convert(){
64          String s1=">"+mTxt+"< (";
65          s1=s1 + mDate.toString()+") de ";
66          s1=s1+mUser.getNameComplete();
67          return s1;
68      }
69  }
70
```

```java
package com.example.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.net.MalformedURLException;
import java.net.URL;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.List;
import java.util.Random;
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();
        r.setDate(new Date());                          //last
modification
```

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

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

```
139
140     public Boolean deleteImage(Recipe r){
141         Boolean success=false;
142         File filesDir= mContext.getFilesDir();
143         File im=new File(filesDir, r.getPhotoFilename());
144         if (im.exists()){
145             im.delete();
146             success=true;
147         }
148         return success;
149     }
150
151     public void clearCookBook(){
152         List<Recipe> recipes=new ArrayList<>();
153         recipes=getRecipes();
154         for (Recipe r:recipes){
155             deleteImage(r);
156             removeRecipe(r);
157         }
158     }
159
160     public void fillCookBook(List<Recipe> recipes){
161         for(Recipe r:recipes){
162             addRecipe(r);
163         }
164     }
165
166     public Boolean isThereMail(){
167         List<Recipe> recipes=new ArrayList<>();
168         recipes=getRecipes();
169         for(Recipe r:recipes){
170             if (r.IsMessage())
171                 return true;
172         }
173         return false;
174     }
175 }
176
```

```java
package com.example.cookbook;

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

import java.util.UUID;
//************************************************************
// to invoke :
//      public SessionInfo mSession;
//      mSession= SessionInfo.get(getApplicationContext());
//           or   SessionInfo.get(getActivity());
//************************************************************
public class SessionInfo {
    private static SessionInfo ourInstance;
    private User mUser;
    private Context mContext;
    private Boolean mIsConnected;
    private Boolean mReqNewSession;
    private String CB_FAMILY="family";
    private String CB_NAME="name";
    private String CB_ID="iduser";
    private String NOT_FOUND="Not found";
    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;
        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));
        }
```

```
52      }
53
54
55      public User getUser() {
56          return mUser;
57      }
58
59      public void setStoredUser(User user){
60          PreferenceManager.getDefaultSharedPreferences(mContext)
61                  .edit()
62                  .putString(CB_FAMILY, user.getFamily())
63                  .putString(CB_NAME, user.getName())
64                  .putString(CB_ID, user.getId().toString())
65                  .apply();
66          mUser=user;
67      }
68      public void clearStoredUser(){
69          SharedPreferences settings = mContext.getSharedPreferences("
    PreferencesName", Context.MODE_PRIVATE);
70          settings.edit().remove("CB_FAMILY").commit();
71          settings.edit().remove("CB_NAME").commit();
72          settings.edit().remove("CB_ID").commit();
73
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
102 }
103
```

```
 1 package com.example.cookbook;
 2
 3 import android.content.Context;
 4 import android.graphics.Bitmap;
 5 import android.util.Log;
 6
 7 import org.json.JSONArray;
 8 import org.json.JSONObject;
 9
10 import java.io.BufferedReader;
11 import java.io.BufferedWriter;
12 import java.io.File;
13 import java.io.FileOutputStream;
14 import java.io.IOException;
15 import java.io.InputStreamReader;
16 import java.io.OutputStream;
17 import java.io.OutputStreamWriter;
18 import java.io.UnsupportedEncodingException;
19 import java.net.HttpURLConnection;
20 import java.net.MalformedURLException;
21 import java.net.URL;
22 import java.net.URLEncoder;
23 import java.text.DateFormat;
24 import java.text.DecimalFormat;
25 import java.text.SimpleDateFormat;
26 import java.util.ArrayList;
27 import java.util.Date;
28 import java.util.HashMap;
29 import java.util.List;
30 import java.util.Map;
31 import java.util.UUID;
32
33 public class NetworkUtils {
34     private SessionInfo mSession;
35     private static final String TAG = "CB_NetworkUtils";
36     private static final String MYSQLDATEFORMAT="yyyy-MM-dd HH:mm:ss";
37
38     public NetworkUtils(Context c){
39         mSession=SessionInfo.get(c);
40     }
41
42     public String sendGetRequest(String uri) {
43         try {
44             URL url = new URL(uri);
45             HttpURLConnection con = (HttpURLConnection) url.
   openConnection();
46             BufferedReader bufferedReader = new BufferedReader(new
   InputStreamReader(con.getInputStream()));
47             String result;
48             StringBuilder sb = new StringBuilder();
49             while((result = bufferedReader.readLine())!=null){
50                 sb.append(result);
51             }
52             return sb.toString();
```

```
53          } catch (Exception e) {
54              return null;
55          }
56      }
57
58      public String sendPostRequest(String requestURL,
59                                    HashMap<String, String>
    postDataParams) {
60          URL url;
61          String response = "";
62          try {
63              url = new URL(requestURL);
64
65              HttpURLConnection conn = (HttpURLConnection) url.
    openConnection();
66              conn.setReadTimeout(mSession.getReadTimeout());
67              conn.setConnectTimeout(mSession.getConnectTimeout());
68              conn.setRequestMethod("POST");
69              conn.setDoInput(true);
70              conn.setDoOutput(true);
71              OutputStream os = conn.getOutputStream();
72              BufferedWriter writer = new BufferedWriter(
73                      new OutputStreamWriter(os, "UTF-8"));
74              writer.write(getPostDataString(postDataParams));
75              writer.flush();
76              writer.close();
77              os.close();
78              int responseCode = conn.getResponseCode();
79              if (responseCode == HttpURLConnection.HTTP_OK) {
80                  BufferedReader br = new BufferedReader(new
    InputStreamReader(conn.getInputStream()));
81                  response = br.readLine();
82              } else {
83                  response = "Error Registering";
84              }
85          } catch (Exception e) {
86              e.printStackTrace();
87          }
88          return response;
89      }
90
91      public String sendPostRequestJson(String requestURL,
92                                    HashMap<String, String>
    postDataParams) {
93
94          URL url;
95          try {
96              url = new URL(requestURL);
97              HttpURLConnection conn = (HttpURLConnection) url.
    openConnection();
98              conn.setReadTimeout(mSession.getReadTimeout());
99              conn.setConnectTimeout(mSession.getConnectTimeout());
100             conn.setRequestMethod("POST");
101             conn.setDoInput(true);
```

```
102             conn.setDoOutput(true);
103             OutputStream os = conn.getOutputStream();
104             BufferedWriter writer = new BufferedWriter(
105                     new OutputStreamWriter(os, "UTF-8"));
106             writer.write(this.getPostDataString(postDataParams));
107             writer.flush();
108             writer.close();
109             os.close();
110             StringBuilder sb = new StringBuilder();
111             BufferedReader bufferedReader = new BufferedReader(new
    InputStreamReader(conn.getInputStream()));
112             String json;
113             while ((json = bufferedReader.readLine()) != null) {
114                 sb.append(json + "\n");
115             }
116             return sb.toString().trim();
117         } catch (Exception e) {
118             Log.d(TAG, ">"+e);
119             return null;
120         }
121     }
122     private String getPostDataString(HashMap<String, String> params)
    {
123         StringBuilder result = new StringBuilder();
124         boolean first = true;
125         for (Map.Entry<String, String> entry : params.entrySet()) {
126             if (first)
127                 first = false;
128             else
129                 result.append("&");
130             try {result.append(URLEncoder.encode(entry.getKey(), "UTF
    -8"));
131             result.append("=");
132             result.append(URLEncoder.encode(entry.getValue(), "UTF-8"
    ));
133             } catch (Exception e){
134                 Log.d(TAG, "Pb with >"+entry.getKey()+":"+entry.
    getValue()+"<");
135             }
136         }
137         return result.toString();
138     }
139     public List<Comment> parseCommentsOfRecipe(String json){
140         Comment c;
141         List<Comment> cs=new ArrayList<>();
142         if ((json==null)||(json.equals(""))) return null;
143         try {
144             JSONArray jarr1=new JSONArray(json);
145             for (int i=0; i<jarr1.length(); i++){
146                 JSONObject obj = jarr1.getJSONObject(i);
147                 c=parseObjectComment(obj);
148                 if (c!=null) cs.add(c);
149             }
150         } catch (Exception e){
```

```
151             Log.d(TAG, "Failure in parsing JSON Array Comments "+e);
152             return null;
153         }
154         return cs;
155     }
156 
157     public Comment parseObjectComment(JSONObject obj){
158         try {
159         UUID uuid=UUID.fromString(obj.getString("id_user"));
160         User u=new User(obj.getString("family"), obj.getString("name"
    ) );
161         u.setId(uuid);
162         String s=obj.getString("date_comment");
163         Date date=new SimpleDateFormat(MYSQLDATEFORMAT).parse(s);
164         return new Comment(obj.getString("comment"),u,date);}
165         catch (Exception e){
166             Log.d(TAG, "Failure in parsing JSONObject Comments "+e);
167             return null;
168         }
169     }
170     public List<Note> parseNotesOfRecipe(String json){
171         Note n;
172         List<Note> ns=new ArrayList<>();
173         if ((json==null)||(json.equals(""))) return null;
174         try {
175             JSONArray jarr1=new JSONArray(json);
176             for (int i=0; i<jarr1.length(); i++){
177                 JSONObject obj = jarr1.getJSONObject(i);
178                 n=parseObjectNote(obj);
179                 if (n!=null) ns.add(n);
180             }
181         } catch (Exception e){
182             Log.d(TAG, "Failure in parsing JSON Array Comments "+e);
183             return null;
184         }
185         return ns;
186     }
187 
188     public Note parseObjectNote(JSONObject obj){
189         try {
190             // uuid=UUID.fromString(obj.getString("id_recipe"));
191             UUID uuid=UUID.fromString(obj.getString("id_user"));
192             User u=new User(obj.getString("family"), obj.getString("
    name") );
193             u.setId(uuid);
194             String s=obj.getString("date_note");
195             Date date=new SimpleDateFormat(MYSQLDATEFORMAT).parse(s);
196             return new Note(obj.getInt("note"),u,date);}
197         catch (Exception e){
198             Log.d(TAG, "Failure in parsing JSONObject Note : "+e);
199             return null;
200         }
201     }
202     public Recipe parseObjectRecipeStamp(JSONObject obj){
```

```
203         try {
204             //---------------- uuid and users
205             UUID uuid = UUID.fromString(obj.getString("id_recipe"));
206             Recipe r = new Recipe(uuid);
207             uuid = UUID.fromString(obj.getString("id_owner"));
208             User u = new User(uuid);
209             r.setOwner(u);
210             //dates
211             Date date;
212             String s1 = obj.getString("lastupdate_recipe");
213             if (s1==null) return null;
214             if ((!s1.equals("null"))&&(s1.length()>5)) {
215                 date = new SimpleDateFormat(MYSQLDATEFORMAT).parse(s1
    );
216                 r.setDate(date);
217             } else {return null;}
218             s1 = obj.getString("lastupdate_photo");
219             if ((!s1.equals("null"))&&(s1.length()>5)) {
220                 date = new SimpleDateFormat(MYSQLDATEFORMAT).parse(s1
    );
221                 r.setDatePhoto(date);
222             }
223             s1=obj.getString("message");
224             if ((s1==null)||(s1.equals("null"))) s1="";
225             r.setMessage(s1);
226             r.setStatus(StatusRecipe.valueOf(obj.getString("status"))
    );
227             s1 = obj.getString("id_from");
228             if ((!s1.equals("null"))&&(s1.length()>5)) {
229                 uuid = UUID.fromString(s1);
230                 u = new User(uuid);
231                 r.setOwner(u);
232             }
233             return r;
234         }
235         catch (Exception e){
236             Log.d(TAG, "Failure in parsing JSONObject Recette Stamp
     : "+e);
237             return null;
238         }
239     }
240
241     public List<Recipe> parseStampsTiers(String json){
242         Recipe r;
243         List<Recipe> rs=new ArrayList<>();
244         if ((json==null)||(json.equals(""))) return null;
245         try {
246             JSONArray jarr1=new JSONArray(json);
247             for (int i=0; i<jarr1.length(); i++){
248                 JSONObject obj = jarr1.getJSONObject(i);
249                 r=parseObjectRecipeStamp(obj);
250                 if (r!=null) rs.add(r);
251             }
252         } catch (Exception e){
```

```
253             Log.d(TAG, "Failure in parsing JSON Array Comments "+e);
254             return null;
255         }
256         return rs;
257     }
258 
259     public boolean parseObjectRecipe(Recipe r, JSONObject obj,
    boolean withphoto, boolean full){
260         try {
261             UUID uuid;
262             String s1, s2;
263             User u;
264             if (full) {
265                 uuid = UUID.fromString(obj.getString("id_owner"));
266                 s1 = obj.getString("owner_family");
267                 s2 = obj.getString("owner_name");
268                 u = new User(s1, s2);
269                 u.setId(uuid);
270                 r.setOwner(u);
271                 s1=obj.getString("message");
272                 if ((s1==null)||(s1.equals("null"))) s1="";
273                 r.setMessage(s1);
274                 r.setStatus(StatusRecipe.valueOf(obj.getString("
    status")));
275                 s1 = obj.getString("id_from");
276                 if ((!s1.equals("null"))&&(s1.length()>5)&&(s1!=null)
    ) {
277                     uuid = UUID.fromString(s1);
278                     s1 = obj.getString("from_family");
279                     s2 = obj.getString("from_name");
280                     u = new User(s1, s2);
281                     u.setId(uuid);
282                     r.setUserFrom(u);
283                 }
284             }
285             //------------ dates
286             Date date;
287             s1 = obj.getString("lastupdate_recipe");
288             if ((!s1.equals("null"))&&(s1.length()>5)&&(s1!=null)) {
289                 date = new SimpleDateFormat(MYSQLDATEFORMAT).parse(s1
    );
290                 r.setDate(date);
291             } else {return false;}
292             s1 = obj.getString("lastupdate_photo");
293             if ((!s1.equals("null"))&&(s1.length()>5)&&(s1!=null)) {
294                 date = new SimpleDateFormat(MYSQLDATEFORMAT).parse(s1
    );
295                 r.setDatePhoto(date);
296             }
297             //----------------- titre, source x2, nbpers
298             s1 = obj.getString("title");
299             URL url;
300             if (s1==null) return false;
301             if (s1.equals("null")) return false;
```

```
302             r.setTitle(s1);
303             r.setSource(obj.getString("source"));
304             s1 = obj.getString("source_url");
305             if (!s1.equals("null")&&(s1.length()>5)&&(s1!=null)) {
306                 try {
307                     url = new URL(s1);
308                     r.setSource_url(url);
309                 } catch (MalformedURLException e) {
310                     Log.d(TAG, "getURL from cookbook download failed
    in parse recipe");
311                 }
312             }
313             int nbp=obj.getInt("nb_pers");
314             if (nbp==0) return false;
315             r.setNbPers(nbp);
316             //---------------- Etapes and ing
317             DecimalFormat formatter = new DecimalFormat("00");
318             for (int i = 0; i < r.getNbStepMax(); i++) {
319                 s1 = "etape" + formatter.format(i + 1);
320                 s2=obj.getString(s1);
321                 if (!s2.equals("null")) {
322                     r.setStep(i + 1, s2);}
323             }
324             for (int i = 0; i < r.getNbIngMax(); i++) {
325                 s1 = "ing" + formatter.format(i + 1);
326                 s2=obj.getString(s1);
327                 if (!s2.equals("null")) {
328                     r.setIngredient(i + 1, s2);}
329             }
330             //----------------- enum
331             r.setSeason(RecipeSeason.valueOf(obj.getString("season"))
    );
332             r.setDifficulty(RecipeDifficulty.valueOf(obj.getString("
    difficulty")));
333
334             // --------------- photo
335             CookBook cb = CookBook.get(mSession.getContext());
336             File f=cb.getPhotoFile(r);
337             if (withphoto) {
338                 s1 = obj.getString("photo");
339                 if ((!s1.equals("null"))&&(!s1.equals(""))&&(s1!=null
    )) {
340                     Bitmap bm=PictureUtils.getBitmapFromString(s1);
341                     if (f.exists()){
342                         Log.d(TAG, "file  " + f.toString()+" ecrasée
    dans parsing recipe !");
343                         f.delete();
344                     }
345                     try {
346                         if(!f.createNewFile()) {
347                             Log.d(TAG, "Error in creating file  "+f.
    toString());
348                         }
349                     } catch (IOException e) {
```

```
350                         Log.d(TAG, "Error in creating file "+f.
    toString()+":" +e);
351                     }
352                     try {
353                         FileOutputStream out = new FileOutputStream(f
    );
354                         bm.compress(Bitmap.CompressFormat.JPEG, 100,
    out);
355                         out.flush();
356                         out.close();
357                     } catch (Exception e) {
358                         Log.d(TAG, "Storing bitmap error  " + e);
359                     }
360
361                 }
362             }
363
364             return true;
365         }
366         catch (Exception e){
367             Log.d(TAG, "Failure in parsing JSONObject Recette : "+e);
368             return false;
369         }
370     }
371
372 }
373
```

```java
package com.example.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;
import java.io.File;

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 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);
        return getScaledBitmap(path, width, height);
```

```
51      }
52
53      public static String getStringImage(Bitmap bmp){
54          ByteArrayOutputStream baos = new ByteArrayOutputStream();
55          bmp.compress(Bitmap.CompressFormat.JPEG, 100, baos);
56          byte[] imageBytes = baos.toByteArray();
57          String encodedImage = Base64.encodeToString(imageBytes,
    Base64.DEFAULT);
58          return encodedImage;
59      }
60
61      public static Bitmap getBitmapFromString(String s){
62          byte[] decodedString = Base64.decode(s, Base64.DEFAULT);
63          Bitmap decodedByte = BitmapFactory.decodeByteArray(
    decodedString, 0, decodedString.length);
64          return decodedByte;
65      }
66
67  }
68
```

```java
package com.example.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.example.cookbook;

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

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> {
    // Constantes
    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 MYSQLDATEFORMAT="yyyy-MM-dd HH:mm:ss";
    // Variable
    private static Context mContext;
    private SessionInfo mSession;
```

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

```
 93             if (syncNotesRecipe(r)) {
 94                 Log.d(TAG, "Notes de " + r.getTitle()+" updaté");
 95                 isChanged=true;
 96                 r.updateTS(AsynCallFlag.NEWRATING,false);
 97                 mCookbook.updateRecipe(r);
 98             }
 99             if (r.IsMarkedDeleted()){
100                 if (deleteRecipeFromCB(r)) {
101                     Log.d(TAG, "Recette effacée : " + r.getTitle());
102                     mCookbook.removeRecipe(r);
103                     mCookbook.deleteImage(r);
104                     isChanged=true;
105                 } else{
106                     Log.d(TAG, "Recette " + r.getTitle()+" non
    effacée");
107                 }
108             }
109         }
110         return isChanged;
111     }
112
113     @Override
114     protected void onPostExecute(Boolean isChanged) {
115         super.onPostExecute(isChanged);
116         Log.d(TAG, "Has changed ? " + isChanged);
117     }
118
119     private Boolean test204() {
120         try {
121             URL url = new URL(mSession.getURLPath() + PHP204);
122             HttpURLConnection conn = (HttpURLConnection) url.
    openConnection();
123             conn.setConnectTimeout(mSession.getConnectTimeout());
124             conn.setReadTimeout(mSession.getReadTimeout());
125             conn.setRequestMethod("HEAD");
126             InputStream in = conn.getInputStream();
127             int status = conn.getResponseCode();
128             in.close();
129             conn.disconnect();
130             return (status == HttpURLConnection.HTTP_NO_CONTENT);
131         } catch (Exception e) {
132             Log.d(TAG, "Test 204 : " + e);
133             return false;
134         }
135     }
136
137     private Boolean uploadRecipe(Recipe r) {
138         HashMap<String, String> data = new HashMap<>();
139         String s1;
140         DateFormat dateFormat = new SimpleDateFormat(MYSQLDATEFORMAT)
    ;
141         data.put("idrecipe", r.getId().toString());
142         data.put("iduser", r.getOwner().getId().toString());
143         data.put("title", r.getTitle());
```

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

```
194         data.put("iduser", mSession.getUser().getId().toString());
195         String result = mNetUtils.sendPostRequestJson(mSession.
    getURLPath() + PHPDELETERECIPE, data);
196         if (result==null) return false;
197         if (!result.trim().equals("1")) {
198             Log.d(TAG, "Retour de "+ PHPDELETERECIPE+" = "+result);
199             return false;}
200         return true;
201     }
202 
203     private Boolean syncCommentsRecipe(Recipe r) {
204         boolean ret=false;
205         HashMap<String, String> data = new HashMap<>();
206         data.put("idrecipe", r.getId().toString());
207         String result = mNetUtils.sendPostRequestJson(mSession.
    getURLPath() + PHPGETCOMMENTSOFRECIPE, data);
208         List<Comment> downloadedComments=mNetUtils.
    parseCommentsOfRecipe(result);
209         List<Comment> recipeComments=r.getComments();
210         //----------- boucle local
211         List<Comment> cloc= new ArrayList<>();
212         Comment c,ci;
213         for (int i = 0; i < recipeComments.size(); i++){
214             ci=recipeComments.get(i);
215             c=new Comment(ci.getTxt(),ci.getUser(),ci.getDate());
216             if (downloadedComments.indexOf(c)==-1) cloc.add(c);
217         }
218         //----------- boucle serveur
219         List<Comment> cser= new ArrayList<>();
220         for (int i = 0; i < downloadedComments.size(); i++){
221             ci=downloadedComments.get(i);
222             c=new Comment(ci.getTxt(),ci.getUser(),ci.getDate());
223             if (recipeComments.indexOf(c)==-1) {
224                 cser.add(c);
225             }
226         }
227         for (int i = 0; i < cser.size(); i++){
228             r.addComment(cser.get(i));
229         }
230         if (cser.size()>0) {
231             mCookbook.updateRecipe(r);
232             ret=true;
233         }
234         if (cloc.size()>0) {
235             if (!uploadComments(r,cloc)) {ret=false;}
236             else {ret=true;}
237         }
238         return ret;
239     }
240 
241     private Boolean uploadComments(Recipe r, List<Comment> cs){
242         if (cs==null) return false;
243         if (cs.size()==0) return true;
244         HashMap<String, String> data = new HashMap<>();
```

```
245         DateFormat dateFormat = new SimpleDateFormat(MYSQLDATEFORMAT)
    ;
246         String s1;
247         boolean b=true;
248         for (Comment c:cs) {
249             data.clear();
250             data.put("idrecipe", r.getId().toString());
251             data.put("idfrom", c.getUser().getId().toString());
252             data.put("comment", c.getTxt());
253             s1 = dateFormat.format(c.getDate());
254             data.put("date",s1  );
255             String result = mNetUtils.sendPostRequestJson(mSession.
    getURLPath() + PHPADDCOMMENTTORECIPE, data);
256             if (result==null) b=false;
257             if (!result.equals("1")) {
258                 Log.d(TAG, "Retour de "+ PHPADDCOMMENTTORECIPE+" = "+
    result);
259                 b=false;}
260         }
261         return b;
262     }
263     private Boolean syncNotesRecipe(Recipe r) {
264         boolean ret=false;
265         HashMap<String, String> data = new HashMap<>();
266         data.put("idrecipe", r.getId().toString());
267         String result = mNetUtils.sendPostRequestJson(mSession.
    getURLPath() + PHPGETNOTESOFRECIPE, data);
268         List<Note> downloadedNotes=mNetUtils.parseNotesOfRecipe(
    result);
269         List<Note> recipeNotes=r.getNotes();
270         //----------- boucle local
271         List<Note> cloc= new ArrayList<>();
272         Note c,ci;
273         for (int i = 0; i < recipeNotes.size(); i++){
274             ci=recipeNotes.get(i);
275             c=new Note(ci.getNote(),ci.getUser(),ci.getDate());
276             if (downloadedNotes.indexOf(c)==-1) cloc.add(c);
277         }
278         //----------- boucle serveur
279         List<Note> cser= new ArrayList<>();
280         for (int i = 0; i < downloadedNotes.size(); i++){
281             ci=downloadedNotes.get(i);
282             c=new Note(ci.getNote(),ci.getUser(),ci.getDate());
283             if (recipeNotes.indexOf(c)==-1) {
284                 cser.add(c);
285             }
286         }
287         for (int i = 0; i < cser.size(); i++){
288             r.addNote(cser.get(i));
289         }
290         if (cser.size()>0) {
291             mCookbook.updateRecipe(r);
292             ret=true;
293         }
```

```
294         if (cloc.size()>0) {
295             if (!uploadNotes(r,cloc)) {ret=false;}
296             else {ret=true;}
297         }
298         return ret;
299     }
300     private Boolean uploadNotes(Recipe r, List<Note> cs){
301         if (cs==null) return false;
302         if (cs.size()==0) return true;
303         HashMap<String, String> data = new HashMap<>();
304         DateFormat dateFormat = new SimpleDateFormat(MYSQLDATEFORMAT)
    ;
305         DecimalFormat formatter = new DecimalFormat("00");
306         String s1;
307         boolean b=true;
308         for (Note c:cs) {
309             data.clear();
310             data.put("idrecipe", r.getId().toString());
311             data.put("idfrom", c.getUser().getId().toString());
312             data.put("note", formatter.format(c.getNote()));
313             s1 = dateFormat.format(c.getDate());
314             data.put("date",s1  );
315             String result = mNetUtils.sendPostRequestJson(mSession.
    getURLPath() + PHPADDNOTETORECIPE, data);
316             if (result==null) b=false;
317             if (!result.equals("1")) {
318                 Log.d(TAG, "Retour de "+ PHPADDNOTETORECIPE+" = "+
    result);
319                 b=false;}
320         }
321         return b;
322     }
323     private Boolean syncTiersRecipe(User user) {
324         boolean ret=false;
325         HashMap<String, String> data = new HashMap<>();
326         data.put("iduser", user.getId().toString());
327         String result = mNetUtils.sendPostRequestJson(mSession.
    getURLPath() + PHPGETSTAMPSTIERS, data);
328         if (result==null)return ret;
329         List<Recipe> tiersRecipe=mNetUtils.parseStampsTiers(result);
330         if (tiersRecipe==null) return ret;
331         boolean withphoto=false,update=false;
332         Recipe rloc, rnew;
333         for(Recipe r:tiersRecipe) {
334             rloc=mCookbook.getRecipe(r.getId());
335             if (rloc==null) {
336                 rnew=downloadRecipe(r);
337                 mCookbook.addRecipe(rnew);
338                 Log.d(TAG, "Recette "+ rnew.getTitle()+" downloadée")
    ;
339                 ret=true;
340             } else {
341                 if (rloc.IsVisible()){
342                     withphoto=IsAfterAndNonNull(r.getDatePhoto(),
```

```
342 rloc.getDatePhoto());
343                     update=IsAfterAndNonNull(r.getDate(), rloc.
    getDate());
344                     if(update || withphoto){
345                         if (updateRecipe(rloc, withphoto)){
346                             mCookbook.updateRecipe(rloc);
347                             Log.d(TAG, "Recette "+ rloc.getTitle()+"
    updatée (with photo :"+withphoto+")");
348                             ret=true;
349                         }
350                   } else continue;
351                 }
352             }
353         }
354         return ret;
355     }
356     private boolean IsAfterAndNonNull(Date ds, Date dl){
357         Calendar c=Calendar.getInstance();
358         c.set(2000,0,1,0,0, 0);
359         Date d0=c.getTime();
360         boolean morerecent=(ds.compareTo(dl)==1);
361         boolean isnotnull=(ds.compareTo(d0)==1);
362         return morerecent && isnotnull;
363     }
364
365     private Recipe downloadRecipe(Recipe ref){
366         if (ref==null) return null;
367         HashMap<String,String> data = new HashMap<>();
368         data.put("idrecipe", ref.getId().toString().trim());
369         data.put("iduser", mSession.getUser().getId().toString().trim
    ());
370         String result = mNetUtils.sendPostRequestJson(mSession.
    getURLPath()+PHPGETRECIPEFROMCBWITHPHOTO,data);
371         Recipe r=new Recipe();
372         try {
373             JSONArray jarr1 = new JSONArray(result);
374             if (jarr1.length()!=1){
375                 Log.d(TAG, " Number of json objects from
    downloadRecipes =" + jarr1.length()+" !");
376                 return null;
377             } else {
378                 JSONObject obj = jarr1.getJSONObject(0);
379                 r = new Recipe(UUID.fromString(obj.getString("
    id_recipe")));
380                 if (!mNetUtils.parseObjectRecipe(r,obj, true, true))
    return null;
381             }
382         } catch (Exception e) {
383             Log.d(TAG, " Error parsing CB in downloadRecipe" + e);
384         }
385         return r;
386     }
387     private Boolean updateRecipe(Recipe ref, boolean withphoto){
388         if (ref==null) return null;
```

```
389         HashMap<String,String> data = new HashMap<>();
390         data.put("idrecipe", ref.getId().toString().trim());
391         data.put("withphoto", (withphoto)?"1":"0");
392         String result = mNetUtils.sendPostRequestJson(mSession.
    getURLPath()+ PHPGETRECIPEFROMCB,data);
393         Recipe r=new Recipe();
394         try {
395             JSONArray jarr1 = new JSONArray(result);
396             if (jarr1.length()!=1){
397                 Log.d(TAG, " Number of json objects from
    downloadRecipes =" + jarr1.length()+" !");
398                 return null;
399             } else {
400                 JSONObject obj = jarr1.getJSONObject(0);
401                 if (!mNetUtils.parseObjectRecipe(ref,obj, withphoto,
    false)) {
402                     Log.d(TAG, " Null recipe from JSOn object" +
    result);
403                     return false;
404                 }
405             }
406         } catch (Exception e) {
407             Log.d(TAG, " Error parsing CB in downloadRecipe" + e);
408             return false;
409         }
410         return true;
411     }
412 }
413
```

```java
package com.example.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.example.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);
    }

}

```

```java
package com.example.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 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.example.cookbook;

import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.drawable.AnimationDrawable;
import android.os.AsyncTask;
import android.os.Bundle;
import android.support.v4.content.ContextCompat;
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.ProgressBar;
import android.widget.TextView;

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

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.net.MalformedURLException;
import java.net.URL;
import java.text.DateFormat;
import java.text.DecimalFormat;
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;
```

```
55      private Button mNewFamily;
56      private ProgressBar mProgressBar;
57      private ImageView mNetAnim;
58      private AnimationDrawable frameNetAnim;
59      private ArrayList<User> memberFamily;
60      private ArrayList<Recipe> mRecipes;
61      private SessionInfo mSession;
62      private NetworkUtils mNetUtils;
63      private User mUser;
64      private int mState;
65      private static final String TAG = "CB_SplashActivity";
66      private final static int NEW_FAMILY=1;
67      private final static int NEW_MEMBER=2;
68      private final static int NEW_SESSION=3;
69      private final static int NEW_PWD=3;
70      private final static Integer MINMAX[][]={{8,45},{1,25},{4,25}};
    // min max pour family, member, pwd strings
71      private static final String REGEX_FAMILY="[-_!?\\w\\p{
    javaLowerCase}\\p{javaUpperCase}()\\p{Space}]*";
72      private static final String REGEX_MEMBER="[-_\\w\\p{javaLowerCase
    }\\p{javaUpperCase}]*";
73      private static final String REGEX_PWD="[-_!?\\w\\p{javaLowerCase}
    \\p{javaUpperCase}()]*";
74      private static final String REGEX[]={REGEX_FAMILY, REGEX_MEMBER,
    REGEX_PWD};
75      private static final String PHPREQFAMILYCOMP="
    getfamilycomposition.php";
76      private static final String PHPREQNEWMEMBER="createnewmember.php"
    ;
77      private static final String PHPREQNEWFAMILY="createnewfamily.php"
    ;
78      private static final String PHPREQUPLOADPHOTO="
    uploadphotointorecipe.php";
79      private static final String PHPREQGETCB="getcookbook.php";
80      private static final String PHPREQGETNOTES="getnotes.php";
81      private static final String PHPREQGETCOMMENTS="getcomments.php";
82      private static final String MYSQLDATEFORMAT="yyyy-MM-dd HH:mm:ss"
    ;
83
84      @Override
85      protected void onCreate(Bundle savedInstanceState) {
86          super.onCreate(savedInstanceState);
87
88          mSession= SessionInfo.get(getApplicationContext());
89          if ((!mSession.IsEmpty())&&(!mSession.IsReqNewSession())){
90              // ******************  SYNC ******************
91              Intent intent=new Intent(getApplicationContext(),
    RecipeListActivity.class);
92              startActivity(intent);
93              return;
94          }
95          mNetUtils=new NetworkUtils(getApplicationContext());
96          ArrayList<User> memberFamily=new ArrayList<>();
97          ArrayList<Recipe> mRecipes=new ArrayList<>();
```

```
 98          TestConnection t;
 99          t = new TestConnection();
100          t.setTestConnexion(getApplicationContext());
101          t.testGo();
102
103          getWindow().setFlags(
104                  WindowManager.LayoutParams.FLAG_FULLSCREEN,
105                  WindowManager.LayoutParams.FLAG_FULLSCREEN);
106          setContentView(R.layout.activity_splash);
107          mMoto=(TextView) findViewById(R.id.splash_moto);
108          mMoto.setText(R.string.splash_moto_txt);
109          if (mSession.IsReqNewSession()){
110              mFamilyEntered=mSession.getUser().getFamily();
111              mMemberEntered=mSession.getUser().getName();
112              mPwdEntered="";
113          } else {
114              mFamilyEntered=getString(R.string.splash_edit_family_hint
    );
115              mMemberEntered=getString(R.string.splash_edit_member_hint
    );
116              mPwdEntered=getString(R.string.splash_edit_pwd_hint);
117          }
118          mProgressBar=(ProgressBar) findViewById(R.id.splash_progBar);
119          mProgressBar.setProgress(0);
120          mEnterMessage=(TextView) findViewById(R.id.
    splash_edit_message);
121          mEnterMessage.setText("");
122          mEnterFamilyLbl=(TextView) findViewById(R.id.
    splash_edit_family_lbl);
123          mEnterFamily=(EditText) findViewById(R.id.splash_edit_family)
    ;
124          mEnterMemberLbl=(TextView) findViewById(R.id.
    splash_edit_member_lbl);
125          mEnterMember=(EditText) findViewById(R.id.splash_edit_member)
    ;
126          mEnterPwdLbl=(TextView) findViewById(R.id.splash_edit_pwd_lbl
    );
127          mEnterPwd=(EditText) findViewById(R.id.splash_edit_pwd);
128          mNewSession=(Button) findViewById(R.id.splash_button_session)
    ;
129          mNewMember=(Button) findViewById(R.id.splash_button_member);
130          mNewFamily=(Button) findViewById(R.id.splash_button_family);
131          updateTop();
132          mNetAnim=(ImageView) findViewById(R.id.net_anim);
133          mNetAnim.setBackgroundResource(R.drawable.network_animation);
134          mNetAnim.setVisibility(View.INVISIBLE);
135          frameNetAnim = (AnimationDrawable) mNetAnim.getBackground();
136          updateDisplayOnAsync(false);
137
138          mEnterFamily.addTextChangedListener(new TextWatcher() {
139              @Override
140              public void beforeTextChanged(CharSequence s, int start,
    int count, int after) {}
141
```

```
142             @Override
143             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
144                 String z=s.toString();
145                 mFamilyEntered=z;
146                 Integer d= getColorOnCondition(z,NEW_FAMILY);
147                 mEnterFamily.setBackgroundColor(ContextCompat.
    getColor(getApplicationContext(), d));
148                 buttonEnabled((d==R.color.bg_enter_val));
149             }
150
151             @Override
152             public void afterTextChanged(Editable s) {}
153         });
154         mEnterMember.addTextChangedListener(new TextWatcher() {
155             @Override
156             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) {}
157
158             @Override
159             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
160                 String z=s.toString();
161                 mMemberEntered=z;
162                 Integer d= getColorOnCondition(z,NEW_MEMBER);
163                 mEnterMember.setBackgroundColor(ContextCompat.
    getColor(getApplicationContext(), d));
164                 buttonEnabled((d==R.color.bg_enter_val));
165             }
166
167             @Override
168             public void afterTextChanged(Editable s) {}
169         });
170         mEnterPwd.addTextChangedListener(new TextWatcher() {
171             @Override
172             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) {}
173
174             @Override
175             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
176                 String z=s.toString();
177                 mPwdEntered=z;
178                 Integer d= getColorOnCondition(z,NEW_PWD);
179                 mEnterPwd.setBackgroundColor(ContextCompat.getColor(
    getApplicationContext(), d));
180                 buttonEnabled((d==R.color.bg_enter_val));
181             }
182
183             @Override
184             public void afterTextChanged(Editable s) {}
185         });
186         mNewSession.setOnClickListener(new View.OnClickListener() {
187             @Override
```

```
188             public void onClick(View v) {
189                 if(testConnectStatus()){
190                     mState=NEW_SESSION;
191                     updateDisplayOnAsync(true);
192                     goAsyncTasks();
193                 }
194                 t.testGo();
195             }
196         });
197         mNewMember.setOnClickListener(new View.OnClickListener() {
198             @Override
199             public void onClick(View v) {
200                 if(testConnectStatus()){
201                     mState=NEW_MEMBER;
202                     updateDisplayOnAsync(true);
203                     goAsyncTasks();
204                 }
205                 t.testGo();
206             }
207         });
208         mNewFamily.setOnClickListener(new View.OnClickListener() {
209             @Override
210             public void onClick(View v) {
211                 if(testConnectStatus()){
212                     mState=NEW_FAMILY;
213                     updateDisplayOnAsync(true);
214                     goAsyncTasks();
215                 }
216                 t.testGo();
217             }
218         });
219     }
220
221     public void ShowHidePass(View view){
222         if(view.getId()==R.id.show_pass_btn){
223             if(mEnterPwd.getTransformationMethod().equals(
    PasswordTransformationMethod.getInstance())){
224                 ((ImageView)(view)).setImageResource(R.drawable.
    ic_pwd_no_vis);
225                 mEnterPwd.setTransformationMethod(
    HideReturnsTransformationMethod.getInstance());
226             }
227             else{
228                 ((ImageView)(view)).setImageResource(R.drawable.
    ic_pwd_vis);
229                 mEnterPwd.setTransformationMethod(
    PasswordTransformationMethod.getInstance());
230
231             }
232         }
233     }
234     private void updateTop(){
235         mEnterFamilyLbl.setText(R.string.splash_edit_family_label);
236         mEnterFamily.setText(mFamilyEntered);
```

```
237         mEnterMemberLbl.setText(R.string.splash_edit_member_label);
238         mEnterMember.setText(mMemberEntered);
239         mEnterPwdLbl.setText(R.string.splash_edit_pwd_label);
240         mEnterPwd.setText(mPwdEntered);
241         mEnterMessage.setText("");
242         mNewSession.setText(R.string.splash_button_session_txt);
243         mNewMember.setText(R.string.splash_button_member_txt);
244         mNewFamily.setText(R.string.splash_button_family_txt);
245     }
246 
247     private void updateDisplayOnAsync(Boolean b){
248         int indNet,indClick;
249         if (b){
250             indNet=View.VISIBLE;
251             indClick=View.INVISIBLE;
252             frameNetAnim.start();
253             mProgressBar.setProgress(1);
254         }
255         else {
256             indNet=View.INVISIBLE;
257             indClick=View.VISIBLE;
258             frameNetAnim.stop();
259             mProgressBar.setProgress(0);
260         }
261         mNetAnim.setVisibility(indNet);
262         mProgressBar.setVisibility(indNet);
263         mNewFamily.setVisibility(indClick);
264         mNewMember.setVisibility(indClick);
265         mNewSession.setVisibility(indClick);
266     }
267 
268     private Boolean IsLenOK(String s, int min, int max){
269         int l=s.length();
270         return ((l>min)&&(l<max));
271     }
272 
273     private Integer getColorOnCondition(String z, Integer state){
274         state=state-1;
275         int retDraw =0;
276         int TRUE=R.color.bg_enter_val;
277         int FALSE=R.color.light_red;
278         retDraw=((Pattern.matches(REGEX[state],z)&&(IsLenOK(z,MINMAX[
    state][0],MINMAX[state][1])))?
279                 TRUE : FALSE);
280         Integer err_mess[]={R.string.enter_err_family,R.string.
    enter_err_member,R.string.enter_err_pwd};
281         if (retDraw==FALSE){
282         mEnterMessage.setText(getResources().getString(err_mess[state
    ],MINMAX[state][0],MINMAX[state][1]));}
283         else {mEnterMessage.setText("");}
284         return retDraw;
285     }
286 
287     private Boolean testConnectStatus(){
```

```
288         mEnterFamily.setBackgroundColor(ContextCompat.getColor(
    getApplicationContext(), R.color.bg_enter_val));
289         mEnterMember.setBackgroundColor(ContextCompat.getColor(
    getApplicationContext(), R.color.bg_enter_val));
290         mEnterPwd.setBackgroundColor(ContextCompat.getColor(
    getApplicationContext(), R.color.bg_enter_val));
291         Boolean ret=true;
292         String message="";
293         if(!mSession.IsConnected()){
294             ret=false;
295             message += getResources().getString(R.string.
    network_err_no_connection)+"\n";
296         }
297         mEnterMessage.setText(message);
298         return ret;
299     }
300 
301     private void buttonEnabled(Boolean b){
302         mNewSession.setEnabled(b);
303         mNewMember.setEnabled(b);
304         mNewFamily.setEnabled(b);
305     }
306 
307     /**
    ************************************************************************
    ******************
308     *                                ASYNC                                 *
309 
    ************************************************************************
    **********************/
310 
311     private void goAsyncTasks() {
312 
313         class GoAsyncTasks extends AsyncTask<Void, Void, Boolean> {
314 
315             @Override
316             protected void onPreExecute() {
317                 super.onPreExecute();
318             }
319 
320             @Override
321             protected void onPostExecute(Boolean b) {
322                 super.onPostExecute(b);
323                 updateDisplayOnAsync(false);
324                 if (b) {
325                     mSession.setStoredUser(mUser);
326                     Intent intent=new Intent(getApplicationContext(),
     RecipeListActivity.class);
327                     startActivity(intent);
328                 }
329                 return;
330             }
331 
```

```
332             @Override
333             protected Boolean doInBackground(Void... voids) {
334                 CookBook cb=CookBook.get(getApplicationContext());
335                 String s=getFamilyComp();
336                 mProgressBar.setProgress(2);
337                 if (s==null){
338                     mEnterMessage.setText(R.string.enter_err_com);
339                     return false; } else {
340                     memberFamily=parseJsonFamily(s); }
341                 mProgressBar.setProgress(5);
342                 Integer cas=goFamilyGet();
343                 if (cas==0) { return false;}
344                 if ((cas==NEW_FAMILY)||(cas==NEW_MEMBER)){
345                     if (createMember(cas)){
346                         mProgressBar.setProgress(10);
347                         mEnterMessage.setText(getString(R.string.
    enter_noerr_new_member));
348                     } else {
349                         mEnterMessage.setText(getString(R.string.
    enter_err_new_member));
350                         return false;}
351                 }
352                 cb.clearCookBook();
353                 s=downloadCB();
354                 if (s==null){
355                     Log.d(TAG, "download cookbook failed");
356                     return false;
357                 }
358                 mProgressBar.setProgress(50);
359                 mRecipes=parseJsonCB(s);
360                 if (!downloadNotesAndParse()){
361                     Log.d(TAG, "Failure in reading and parsing Notes"
    );
362                     return false;
363                 }
364                 mProgressBar.setProgress(70);
365                 if (!downloadCommentsAndParse()){
366                     Log.d(TAG, "Failure in reading and parsing
    Comments");
367                     return false;
368                 }
369                 mProgressBar.setProgress(70);
370                 cb.fillCookBook(mRecipes);
371                 mEnterMessage.setText("Number of recipes : "+mRecipes
    .size());
372                 mProgressBar.setProgress(90);
373                 return true;
374             }
375         }
376         GoAsyncTasks getAsync = new GoAsyncTasks();
377         getAsync.execute();
378     }
379
380     private String getFamilyComp(){
```

```
381         HashMap<String,String> data = new HashMap<>();
382         data.put("family", mFamilyEntered.trim());
383         String result = mNetUtils.sendPostRequestJson(mSession.
    getURLPath()+PHPREQFAMILYCOMP,data);
384         return result;
385     }
386 
387     private Boolean createMember(int flag){
388         String link=mSession.getURLPath();
389         if (flag==NEW_MEMBER) {
390             link=link+PHPREQNEWMEMBER;
391         } else {
392             if (flag==NEW_FAMILY) {
393                 link=link+PHPREQNEWFAMILY;
394             } else {
395                 Log.d(TAG, "Function createmember flag bad value");
396                 return false;
397             }
398         }
399         HashMap<String,String> data = new HashMap<>();
400         data.put("family", mFamilyEntered.trim());
401         data.put("pwd", mPwdEntered.trim());
402         data.put("name", mMemberEntered.trim());
403         mUser=new User(mFamilyEntered.trim(),mMemberEntered.trim());
404         data.put("iduser", mUser.getId().toString().trim());
405         String result = mNetUtils.sendPostRequestJson(link,data);
406         if (result.trim().equals("1")) {
407             Log.d(TAG, "Function createmember succeed");
408             return true;
409         } else {
410             Log.d(TAG, "Function createmember failed");
411             return false;
412         }
413     }
414 
415     private String downloadCB(){
416         HashMap<String,String> data = new HashMap<>();
417         data.put("iduser", mUser.getId().toString().trim());
418         data.put("pwd", mPwdEntered.trim());
419         String result = mNetUtils.sendPostRequestJson(mSession.
    getURLPath()+PHPREQGETCB,data);
420         return result;
421     }
422 
423     private Boolean downloadNotesAndParse(){
424         HashMap<String,String> data = new HashMap<>();
425         data.put("iduser", mUser.getId().toString().trim());
426         String json = mNetUtils.sendPostRequestJson(mSession.
    getURLPath()+PHPREQGETNOTES,data);
427         //User u;
428         //String s;
429         UUID uuid;
430         //Date date;
431         Note n;
```

```
        try {
            JSONArray jarr1=new JSONArray(json);
            for (int i=0; i<jarr1.length(); i++){
                JSONObject obj = jarr1.getJSONObject(i);
                uuid=UUID.fromString(obj.getString("id_recipe"));
                n=mNetUtils.parseObjectNote(obj);
                for(Recipe r:mRecipes){
                    if (r.getId().equals(uuid)){
                        r.addNote(n);
                        break;
                    }
                }
            }
        } catch (Exception e){
                Log.d(TAG, "Failure in parsing JSON Notes "+e);
                return false;
            }
        return true;
    }

    private Boolean downloadCommentsAndParse(){
        HashMap<String,String> data = new HashMap<>();
        data.put("iduser", mUser.getId().toString().trim());
        String json = mNetUtils.sendPostRequestJson(mSession.
getURLPath()+PHPREQGETCOMMENTS,data);
        UUID uuid;
        Comment c;
        try {
            JSONArray jarr1=new JSONArray(json);
            for (int i=0; i<jarr1.length(); i++){
                JSONObject obj = jarr1.getJSONObject(i);
                c=mNetUtils.parseObjectComment(obj);
                uuid=UUID.fromString(obj.getString("id_recipe"));
                if (c!=null) {
                    for(Recipe r:mRecipes){
                        if (r.getId().equals(uuid)){
                            r.addComment(c);
                            break;
                        }
                    }
                }
            }
        } catch (Exception e){
            Log.d(TAG, "Failure in parsing JSON Array Comments "+e);
            return false;
        }
        return true;
    }

    public ArrayList<User> parseJsonFamily(String json){
        ArrayList<User> ret=new ArrayList<>();
        User u;
        String name,dateString;
        UUID uuid;
```

```
485         Date date;
486         mPwdRead="";
487         try {
488             JSONArray jarr1=new JSONArray(json);
489             for (int i=0; i<jarr1.length(); i++){
490                 JSONObject obj = jarr1.getJSONObject(i);
491                 name=obj.getString("name");
492                 uuid=UUID.fromString(obj.getString("id_user"));
493                 u=new User(mFamilyEntered, name );
494                 u.setId(uuid);
495                 dateString=obj.getString("last_sync");
496                 date=new SimpleDateFormat(MYSQLDATEFORMAT).parse(
    dateString);
497                 u.setDate(date);
498                 ret.add(u);
499                 mPwdRead=obj.getString("pass");
500             }
501         }
502         catch (Exception e){
503             Log.d(TAG, "Failure in parsing JSON FamilyGet" +e);
504         }
505         return ret;
506     }
507     public ArrayList<Recipe> parseJsonCB(String json){
508         ArrayList<Recipe> result=new ArrayList<>();
509         Recipe r;
510         try {
511             JSONArray jarr1 = new JSONArray(json);
512             for (int j = 0; j < jarr1.length(); j++) {
513                 JSONObject obj = jarr1.getJSONObject(j);
514                 r = new Recipe(UUID.fromString(obj.getString("
    id_recipe")));
515                 if (mNetUtils.parseObjectRecipe(r,obj, true, true))
    result.add(r);
516             }
517         } catch (Exception e) {
518             Log.d(TAG, " Error parsing CB" + e);
519         }
520         return result;
521     }
522
523     private Integer goFamilyGet(){
524         switch(mState){
525             case NEW_SESSION : {
526                 if (memberFamily.size()==0) {
527                     mEnterMessage.setText(R.string.
    enter_err_member_wo_family);
528                     mEnterFamily.setBackgroundColor(ContextCompat.
    getColor(getApplicationContext(), R.color.light_red)); }
529                 else {
530                     mEnterMessage.setText(R.string.
    enter_noerr_fam_found);
531                     User u= new User("","");
532                     for(User user:memberFamily){
```

```
533                         if (user.getName().equals(mMemberEntered)){u=
    user;}
534                     }
535                     if (u.getName().equals("")) {
536                         mEnterMessage.setText(R.string.
    enter_err_member_in_family);
537                         mEnterMember.setBackgroundColor(ContextCompat
    .getColor(getApplicationContext(), R.color.light_red));
538                     } else {
539                         if (!mPwdRead.equals(mPwdEntered.trim())){
540                             mEnterMessage.setText(R.string.
    enter_err_wrong_pass);
541                             mEnterPwd.setBackgroundColor(
    ContextCompat.getColor(getApplicationContext(), R.color.light_red));
542                         } else {
543                             mUser=u;
544                         mEnterMessage.setText(getString(R.string.
    enter_noerr_new_session, u.getName()));
545                         return NEW_SESSION; }
546                     }
547                 }
548                 break;
549             }
550             case NEW_FAMILY : {
551                 if (memberFamily.size()==0) {
552                     mEnterMessage.setText(R.string.
    enter_noerr_welcome);
553                     return NEW_FAMILY;
554                 }
555                 else {
556                     mEnterMessage.setText(R.string.
    enter_err_family_already);
557                     mEnterFamily.setBackgroundColor(ContextCompat.
    getColor(getApplicationContext(), R.color.light_red));
558                 }
559                 break;
560             }
561             case NEW_MEMBER : {
562                 if (memberFamily.size()==0) {
563                     mEnterMessage.setText(R.string.
    enter_err_member_wo_family);
564                     mEnterFamily.setBackgroundColor(ContextCompat.
    getColor(getApplicationContext(), R.color.light_red));
565                 }
566                 else {
567                     // family found
568                     User u= new User("","");
569                     for(User user:memberFamily){
570                         if (user.getName().equals(mMemberEntered)){u=
    user;}
571                     }
572                     if (u.getName().equals("")) {
573                         if (!mPwdRead.equals(mPwdEntered.trim())){
574                             mEnterMessage.setText(R.string.
```

```
574 enter_err_wrong_pass);
575                             mEnterPwd.setBackgroundColor(
    ContextCompat.getColor(getApplicationContext(), R.color.light_red));
576                         } else {
577                         mEnterMessage.setText(R.string.
    enter_noerr_new_member);
578                         return NEW_MEMBER;}
579                     } else {
580                         mEnterMessage.setText(R.string.
    enter_err_member_notin_family);
581                         mEnterMember.setBackgroundColor(ContextCompat
    .getColor(getApplicationContext(), R.color.light_red));
582                     }
583                 }
584                 break;
585             }
586             default : {
587                 Log.d(TAG, "Switch case on mState anomaly : "+mState)
    ;
588             }
589         }
590         return 0;
591     }
592 }
593
```

```java
package com.example.cookbook;

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

import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.URL;
// ********************************************************************
*
//  to use :
//        TestConnection t;
//        t = new TestConnection();
//        t.setTestConnexion(getApplicationContext());
//        t.testGo(); start async => result in Boolean  mSession.
IsConnected()
//************************************************************

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;}
```

```
51              }
52
53              @Override
54              protected void onPostExecute(Boolean s) {
55                  super.onPostExecute(s);
56                  mSession.setConnection(s);
57              }
58          }
59
60          testCon test = new testCon();
61          test.execute();
62      }
63  }
64
```

```java
package com.example.cookbook;

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

import com.example.cookbook.RecipeDbSchema.RecipeTable;

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

```

```
54      }
55
56      @Override
57      public void onUpgrade(SQLiteDatabase db, int oldVersion, int
    newVersion) {
58
59      }
60  }
61
```

```java
package com.example.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.example.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.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.Button;
import android.widget.RatingBar;

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

public class RatePickerFragment extends DialogFragment {
    public static final String EXTRA_RATE = "com.example.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.dialog_star_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);
                            }
```

```
51                          });
52          dialog.setNegativeButton("Cancel",
53                  new DialogInterface.OnClickListener() {
54
55                      @Override
56                      public void onClick(DialogInterface dialog, int
    which) {
57                          return ;
58                      }
59                  });
60          AlertDialog dia=dialog.show();
61          Button b=dia.getButton(DialogInterface.BUTTON_POSITIVE);
62          if (b != null) {
63              b.setTextColor(getResources().getColor(R.color.fg_icon));
64          }
65          return dia;
66      }
67
68      private void sendResult(int resultCode, int rate) {
69          if (getTargetFragment() == null) {
70              return;
71          }
72          // renvoie le new rating (cela ne serta à rien, juste pour
    essayer)
73          Intent intent = new Intent();
74          intent.putExtra(EXTRA_RATE, rate);
75          getTargetFragment()
76                  .onActivityResult(getTargetRequestCode(), resultCode,
     intent);
77      }
78       private void addNotetoRecipe(){
79           SessionInfo loSession= SessionInfo.get(getActivity());
80           CookBook locookbook=CookBook.get(getActivity());
81           Recipe r=locookbook.getRecipe(mUUID);
82           User lu=loSession.getUser();
83           r.addNote(new Note(mRate, lu));
84           r.updateTS(AsynCallFlag.NEWRATING, true);
85           locookbook.updateRecipe(r);
86       }
87  }
88
```

```java
package com.example.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.net.Uri;
import android.os.Bundle;
import android.provider.MediaStore;
import android.support.v4.app.Fragment;
import android.support.v4.content.FileProvider;
import android.text.Editable;
import android.text.TextWatcher;
import android.text.format.DateFormat;
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.ByteArrayOutputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
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 String DIALOG_DATE="DialogDate";
    private static final int REQUEST_DATE=0;
    private static final int REQUEST_PHOTO= 2;
    private static final String TAG = "CB_RecipeEditFragment";
    // Variables internes
    private Recipe mRecipe;
    private Recipe mRecipeInit;
    private File mPhotoFile;
    private EditText mTitleField;
    private EditText mSourceField;
```

```
55      private EditText mSourceUrl;
56      private EditText mNbPersField;
57      //    private Button mDateButton;
58      private TextView[] mSTextView;
59      private EditText mNewStepField;
60      private ImageButton mNewStepEnter;
61      private ImageButton mNewStepBack;
62      private int mStepNb;
63      private TextView[] mSTextIngView;
64      private EditText mNewIngField;
65      private ImageButton mNewIngEnter;
66      private ImageButton mNewIngBack;
67      private int mIngNb;
68      private Spinner mSeasonSpinner;
69      private Spinner mDifficultySpinner;
70      private ScrollView mScroll;
71      private ImageView mPhotoView;
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          UUID recipeId=(UUID) getArguments().getSerializable(
    ARG_RECIPE_ID);
87          mRecipe=CookBook.get(getActivity()).getRecipe(recipeId);
88          mRecipeInit=CookBook.get(getActivity()).getRecipe(recipeId);
89          setHasOptionsMenu(true);
90          mPhotoFile=CookBook.get(getActivity()).getPhotoFile(mRecipe);
91          mStepNb=mRecipe.getNbStep();
92          mIngNb=mRecipe.getNbIng();
93      }
94
95      @Override
96      public void onPause(){
97          super.onPause();
98          mRecipe.updateTS(AsynCallFlag.NEWRECIPE, !mRecipe.
    hasNotChangedSince(mRecipeInit));
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, menu);
```

```
106     }
107
108     @Override
109     public boolean onOptionsItemSelected(MenuItem item) {
110         switch (item.getItemId()) {
111             case R.id.recipe_menu_report:
112                 Intent i=new Intent(Intent.ACTION_SEND);
113                 i.setType("text/plain");
114                 i.putExtra(Intent.EXTRA_TEXT, getRecipeReport());
115                 i.putExtra(Intent.EXTRA_SUBJECT,getString(R.string.
    recipe_report_subject));
116                 startActivity(i);
117                 return true;
118             case R.id.recipe_menu_delete:
119                 CookBook.get(getActivity()).markRecipeToDelete(
    mRecipe);
120                 getActivity().onBackPressed();
121             default:
122                 return super.onOptionsItemSelected(item);
123         }
124     }
125
126     @Override
127     public View onCreateView(LayoutInflater inflater, ViewGroup
    container, Bundle savedInstanceState){
128         final View v=inflater.inflate(R.layout.fragment_recipe_edit,
    container, false);
129         mScroll=(ScrollView) v.findViewById(R.id.
    fragment_recipe_scroll);
130         PackageManager packageManager=getActivity().getPackageManager
    ();
131             final Intent captureImage=new Intent(MediaStore.
    ACTION_IMAGE_CAPTURE);
132             boolean canTakePhoto=(mPhotoFile != null) &&
133                     (captureImage.resolveActivity(packageManager)!=
    null);
134         mPhotoView=(ImageView) v.findViewById(R.id.recipe_photo);
135         mPhotoView.setOnClickListener(new View.OnClickListener() {
136             @Override
137             public void onClick(View v) {
138                 Uri uri= FileProvider.getUriForFile(getActivity(),
139                         "com.example.cookbook.fileprovider",
    mPhotoFile);
140                 captureImage.putExtra(MediaStore.EXTRA_OUTPUT, uri);
141                 List<ResolveInfo> cameraActivities=getActivity()
142                         .getPackageManager().queryIntentActivities(
    captureImage,
143                                 PackageManager.MATCH_DEFAULT_ONLY);
144                 for(ResolveInfo activity : cameraActivities){
145                     getActivity().grantUriPermission(activity.
    activityInfo.packageName,
146                             uri, Intent.
    FLAG_GRANT_WRITE_URI_PERMISSION);
147                 }
```

```
148                 startActivityForResult(captureImage, REQUEST_PHOTO);
149             }
150         });
151         updatePhotoView();
152         mTitleField= (EditText) v.findViewById(R.id.recipe_title);
153         mTitleField.setText(mRecipe.getTitle());
154         mTitleField.addTextChangedListener(new TextWatcher() {
155             @Override
156             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) {
157                 // blank
158             }
159 
160             @Override
161             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
162                 mRecipe.setTitle(s.toString());
163             }
164 
165             @Override
166             public void afterTextChanged(Editable s) {
167                 //blank
168             }
169         });
170 
171         mSourceField= (EditText) v.findViewById(R.id.recipe_source);
172         mSourceField.setText(mRecipe.getSource());
173         mSourceField.addTextChangedListener(new TextWatcher() {
174             @Override
175             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) {
176                 // blank
177             }
178 
179             @Override
180             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
181                 mRecipe.setSource(s.toString());
182             }
183 
184             @Override
185             public void afterTextChanged(Editable s) {
186 
187             }
188         });
189 
190         mSourceUrl=(EditText) v.findViewById(R.id.recipe_source_url);
191         mSourceUrl.setText(mRecipe.getSource_url().toString());
192         mSourceUrl.addTextChangedListener(new TextWatcher() {
193             @Override
194             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) {
195 
196             }
```

```
197
198             @Override
199             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
200                 if (s.toString().equals("")) {} else {
201                     try {
202                         URL url=new URL(s.toString());
203                         mRecipe.setSource_url(url);
204                     } catch (MalformedURLException e) {
205                         Log.d(TAG, "onTextChanged de mSource_url >" +
    s.toString()+"< Failed >"+ e);
206                     }
207                     }
208             }
209
210             @Override
211             public void afterTextChanged(Editable s) {
212
213             }
214         });
215
216         mNbPersField= (EditText) v.findViewById(R.id.recipe_nbpers);
217         mNbPersField.setText(mRecipe.getNbPers()+"");
218         mNbPersField.addTextChangedListener(new TextWatcher() {
219             @Override
220             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) {
221
222             }
223
224             @Override
225             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
226                 if (s.toString().equals("")) {} else {
227                     int nb_entered = Integer.parseInt(s.toString());
228                     if ((nb_entered > 0) && (nb_entered < 13)) {
229                         mRecipe.setNbPers(nb_entered);
230                     }
231                 }
232             }
233
234             @Override
235             public void afterTextChanged(Editable s) {
236
237             }
238         });
239
240
241
242         mSeasonSpinner= (Spinner) v.findViewById(R.id.recipe_season);
243         ArrayAdapter<CharSequence> adapterSeason = ArrayAdapter.
    createFromResource(getContext(),
244                 R.array.recipe_season_array, android.R.layout.
    simple_spinner_item);
```

```
245         mSeasonSpinner.setAdapter(adapterSeason);
246         mSeasonSpinner.setSelection(RecipeSeason.getIndex(mRecipe.
    getSeason()));
247         mSeasonSpinner.setOnItemSelectedListener(new AdapterView.
    OnItemSelectedListener() {
248             @Override
249             public void onItemSelected(AdapterView<?> parent, View
    view, int position, long id) {
250                 mRecipe.setSeason(RecipeSeason.getSeason(position));
251             }
252 
253             @Override
254             public void onNothingSelected(AdapterView<?> parent) {
255 
256             }
257         });
258 
259         mDifficultySpinner= (Spinner) v.findViewById(R.id.
    recipe_difficulty);
260         ArrayAdapter<CharSequence> adapterDifficulty = ArrayAdapter.
    createFromResource(getContext(),
261                 R.array.recipe_difficulty_array, android.R.layout.
    simple_spinner_item);
262         mDifficultySpinner.setAdapter(adapterDifficulty);
263         mDifficultySpinner.setSelection(RecipeDifficulty.getIndex(
    mRecipe.getDifficulty()));
264         mDifficultySpinner.setOnItemSelectedListener(new AdapterView.
    OnItemSelectedListener() {
265             @Override
266             public void onItemSelected(AdapterView<?> parent, View
    view, int position, long id) {
267                 mRecipe.setDifficulty(RecipeDifficulty.getDifficulty(
    position));
268             }
269 
270             @Override
271             public void onNothingSelected(AdapterView<?> parent) {
272 
273             }
274         });
275 
276         final int[] rID= {R.id.recipe_S1,R.id.recipe_S2,R.id.
    recipe_S3,R.id.recipe_S4,
277                 R.id.recipe_S5,R.id.recipe_S6,R.id.recipe_S7,R.id.
    recipe_S8,R.id.recipe_S9};
278         mSTextView= new TextView[mRecipe.getNbStepMax()];
279         for(int i=0;i<mRecipe.getNbStepMax();i++) {
280             mSTextView[i] = (TextView) v.findViewById(rID[i]);
281         }
282         mNewStepEnter=(ImageButton) v.findViewById(R.id.
    recipe_step_enter);
283         mNewStepBack=(ImageButton) v.findViewById(R.id.
    recipe_step_back);
284         updateListStep(v);
```

```
285
286         mNewStepField= (EditText) v.findViewById(R.id.recipe_new_step
    );
287         mNewStepField.addTextChangedListener(new TextWatcher() {
288             @Override
289             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) {
290                 // blank
291             }
292
293             @Override
294             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
295                 mRecipe.setStep(mStepNb+1, s.toString());
296             }
297             @Override
298             public void afterTextChanged(Editable s) {
299
300             }
301         });
302
303         mNewStepEnter.setOnClickListener(new View.OnClickListener() {
304             @Override
305             public void onClick(View v) {
306 //                Toast.makeText(getActivity(), mRecipe.getStep(
    mStepNb+1)+"/"+mStepNb,
307 //                        Toast.LENGTH_SHORT).show();
308                 mNewStepField.setVisibility(updateListStep(v));
309                 mNewStepField.getText().clear();
310             }
311         });
312
313         mNewStepBack.setOnClickListener(new View.OnClickListener() {
314             @Override
315             public void onClick(View v) {
316                 mRecipe.setStep(mStepNb, "");
317                 mRecipe.setStep(mStepNb+1, "");
318                 mNewStepField.setVisibility(updateListStep(v));
319                 mNewStepField.getText().clear();
320             }
321         });
322
323         final int[] rIngID= {R.id.recipe_I01,R.id.recipe_I02,R.id.
    recipe_I03,R.id.recipe_I04,
324                 R.id.recipe_I05,R.id.recipe_I06,R.id.recipe_I07,R.id.
    recipe_I08,
325                 R.id.recipe_I09,R.id.recipe_I10,R.id.recipe_I11,R.id.
    recipe_I12,
326                 R.id.recipe_I13,R.id.recipe_I14,R.id.recipe_I15};
327         mSTextIngView= new TextView[mRecipe.getNbIngMax()];
328         for(int i=0;i<mRecipe.getNbIngMax();i++) {
329             mSTextIngView[i] = (TextView) v.findViewById(rIngID[i]);
330         }
331         mNewIngEnter=(ImageButton) v.findViewById(R.id.
```

```
recipe_ing_enter);
        mNewIngBack=(ImageButton) v.findViewById(R.id.recipe_ing_back
);
        updateListIng(v);

        mNewIngField= (EditText) v.findViewById(R.id.recipe_new_ing);
        mNewIngField.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int start,
int count, int after) {

            }

            @Override
            public void onTextChanged(CharSequence s, int start, int
before, int count) {
                mRecipe.setIngredient(mIngNb+1, s.toString());
            }

            @Override
            public void afterTextChanged(Editable s) {

            }
        });
        mNewIngEnter.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                mNewIngField.setVisibility(updateListIng(v));
                mNewIngField.getText().clear();
            }
        });
        mNewIngBack.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                mRecipe.setIngredient(mIngNb, "");
                mRecipe.setIngredient(mIngNb+1, "");
                mNewIngField.setVisibility(updateListIng(v));
                mNewIngField.getText().clear();
            }
        });

       return v;
    }


    @Override
    public void onActivityResult(int requestCode, int resultCode,
Intent data) {
        if (resultCode != Activity.RESULT_OK) {
            return;
        }

        if (requestCode==REQUEST_PHOTO){
            Uri uri=FileProvider.getUriForFile(getActivity(),
```

```
381                     "com.example.cookbook.fileprovider", mPhotoFile);
382             getActivity().revokeUriPermission(uri, Intent.
    FLAG_GRANT_WRITE_URI_PERMISSION);
383             if (!ResizePhoto(mRecipe)){
384                 // pb
385             }
386             mRecipe.setDatePhoto(new Date());
387             mRecipe.updateTS(AsynCallFlag.NEWPHOTO,true);
388             updatePhotoView();
389         }
390     }
391
392     private String getRecipeReport(){
393         String report="";
394         Integer iplus=0;
395         report =getString(R.string.recipe_report_title, mRecipe.
    getTitle())+"\n";
396         report +=getString(R.string.recipe_report_owner,mRecipe.
    getOwner().getNameComplete())+"\n";
397
398          if(!mRecipe.getSource_url_name().equals("")){
399             report += getString(R.string.recipe_report_url, mRecipe.
    getSource_url_name())+"\n";
400         }
401         for(int i=0;i<mRecipe.getNbIng();i++){
402             report += getString(R.string.recipe_report_ing, mRecipe.
    getIngredient(i+1))+"\n";
403         }
404         for(int i=0;i<mRecipe.getNbStep();i++){
405             iplus=i+1;
406             report += getString(R.string.recipe_report_step, iplus+""
    ,
407                     mRecipe.getStep(i+1))+"\n";
408         }
409         report +=getString(R.string.recipe_report_final);
410         return report;
411     }
412
413     private void updatePhotoView(){
414         if (mPhotoFile==null || !mPhotoFile.exists()){
415             mPhotoView.setImageDrawable(getResources().getDrawable(R.
    drawable.ic_recipe_camera));
416         } else {
417             Bitmap bitmap=PictureUtils.getScaledBitmap(mPhotoFile.
    getPath(), getActivity());
418             mPhotoView.setImageBitmap(bitmap);
419         }
420     }
421
422     private int updateListStep(View v) {
423         int imax=mRecipe.getNbStepMax(), iplus;
424         String[] display=new String[imax];
425         mStepNb = mRecipe.getNbStep();
426         int gone=View.GONE;
```

```
427         int visible=View.VISIBLE;
428         mSTextView[0].setText("1...");
429         mSTextView[0].setVisibility(visible);
430         for(int i=0;i<imax;i++){
431             iplus=i+1; display[i]=iplus+". "+mRecipe.getStep(i+1);
432             if (mStepNb>0){mSTextView[i].setText(display[i]);}
433             if (i>=0){mSTextView[i].setVisibility((mStepNb>i)?
    visible:gone);}
434         }
435         mNewStepBack.setVisibility((mStepNb==0)? gone:visible);
436         mNewStepEnter.setVisibility((mStepNb==imax)? gone:visible);
437         return ((mStepNb==imax)?gone:visible);
438     }
439
440     private int updateListIng(View v) {
441         int imax=mRecipe.getNbIngMax(), iplus;
442         String[] display=new String[imax];
443         mIngNb = mRecipe.getNbIng();
444         int gone=View.GONE;
445         int visible=View.VISIBLE;
446         mSTextIngView[0].setText("-...");
447         mSTextIngView[0].setVisibility(visible);
448         for(int i=0;i<imax;i++){
449             iplus=i+1; display[i]="-"+mRecipe.getIngredient(i+1);
450             if (mIngNb>0){mSTextIngView[i].setText(display[i]);}
451             if (i>=0){mSTextIngView[i].setVisibility((mIngNb>i)?
    visible:gone);}
452         }
453         mNewIngBack.setVisibility((mIngNb==0)? gone:visible);
454         mNewIngEnter.setVisibility((mIngNb==imax)? gone:visible);
455         return ((mIngNb==imax)?gone:visible);
456     }
457
458
459     private Boolean ResizePhoto(Recipe r) {
460         File f = CookBook.get(getActivity()).getPhotoFile(r);
461         if (f==null || !f.exists()){
462             Log.d(TAG, "No file from Cookbook for this recipe :" + r.
    getTitle()+" Error CB004");
463             return false;
464         } else {
465             Bitmap bitmap=PictureUtils.getScaledBitmap(mPhotoFile.
    getPath(), getActivity(), 600);
466             f.delete();
467             try {
468                 if(!f.createNewFile()) {
469                     Log.d(TAG, "File not deleted ! Error CB001");
470                     return false;
471                 }
472             } catch (IOException e) {
473                 Log.d(TAG, "Error in creating new file : "+e+" Error
    CB002");
474                 return false;
475             }
```

```
476             try {
477
478                 FileOutputStream fOut = new FileOutputStream(f);
479                 bitmap.compress(Bitmap.CompressFormat.JPEG, 100, fOut
    );
480                 fOut.flush();
481                 fOut.close();
482                 return true;
483             } catch (IOException e) {
484                 Log.d(TAG, "Error in reducing and saving file "+"
    Error CB003");
485                 return false;
486             }
487         }
488     }
489 }
490
```

```java
package com.example.cookbook;

import android.support.v4.app.Fragment;

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

```

```java
package com.example.cookbook;

import android.app.Activity;
import android.app.Dialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.graphics.Bitmap;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
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.Button;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.RatingBar;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.Toolbar;

import java.io.File;
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Iterator;
import java.util.List;
import java.util.stream.Collectors;

//-----------------------------------------------------------------------------
// print 16/07/2021
// Saved GitHUb V2.0
//-----------------------------------------------------------------------------
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 int mSortOption;
    private int finalRate;
    private static final int maskSortTitle=1 <<3;
```

```
53      private static final int maskSortSource=1;
54      private static final int maskSortNote=1 <<2;
55      private static final int maskSortSeason=1 <<4;
56      private static final int maskSortDifficulty=1 <<5;
57      private static final String TAG = "CB_RecipeListFragment";
58      private static final String DIALOG_RATE = "DialogRate";
59      private static final int REQUEST_RATE = 0;
60
61      @Override
62      public void onCreate(Bundle savedInstanceState) {
63          super.onCreate(savedInstanceState);
64          setHasOptionsMenu(true);
65          mSession= SessionInfo.get(getActivity());
66          mSortOption=0;
67          mRecipeInit=new Recipe();
68          AppCompatActivity activity = (AppCompatActivity) getActivity(
    );
69          User u=mSession.getUser();
70          activity.getSupportActionBar().setSubtitle(getString(R.string
    .recipe_display_author,u.getName(),u.getFamily()));
71          AsyncCallClass instanceAsync = new AsyncCallClass(getContext(
    ));
72          instanceAsync.execute();
73      }
74
75      @Override
76      public void onActivityResult(int requestCode, int resultCode,
    Intent data) {
77          if (resultCode != Activity.RESULT_OK) {
78              return;
79          }
80          //todo P3 enlever request rate reliquat
81          if (requestCode == REQUEST_RATE) {
82              Integer rate = (Integer) data
83                      .getSerializableExtra(RatePickerFragment.
    EXTRA_RATE);
84              updateUI();
85          }
86      }
87
88      @Override
89      public View onCreateView(LayoutInflater inflater, ViewGroup
    container,
90                               Bundle savedInstanceState) {
91          View view = inflater.inflate(R.layout.fragment_recipe_list,
    container, false);
92          mRecipeRecyclerView = (RecyclerView) view.findViewById(R.id.
    recipe_recycler_view);
93          mRecipeRecyclerView.setLayoutManager(new LinearLayoutManager(
    getActivity()));
94          if (savedInstanceState!=null){
95              mSortOption=savedInstanceState.getInt(SAVED_SORT_STATUS);
96          }
97          updateUI();
```

```
 98          return view;
 99      }
100
101      @Override
102      public void onResume(){
103          super.onResume();
104          updateUI();
105      }
106
107      @Override
108      public void onSaveInstanceState(Bundle outState) {
109          super.onSaveInstanceState(outState);
110          outState.putInt(SAVED_SORT_STATUS, mSortOption);
111      }
112
113      @Override
114      public void onCreateOptionsMenu(Menu menu, MenuInflater inflater)
     {
115          super.onCreateOptionsMenu(menu, inflater);
116          inflater.inflate(R.menu.fragment_recipe_list, menu);
117          mMessageItem = menu.findItem(R.id.new_mail);
118          CookBook cookbook=CookBook.get(getActivity());
119          mMessageItem.setVisible(cookbook.isThereMail());
120      }
121
122      @Override
123      public boolean onOptionsItemSelected(MenuItem item){
124          switch (item.getItemId()){
125              case R.id.new_recipe:
126                  Recipe recipe=new Recipe();
127                  SessionInfo session=SessionInfo.get(getActivity());
128                  recipe.setOwner(session.getUser());
129                  recipe.updateTS(AsynCallFlag.NEWRECIPE, true);
130                  CookBook.get(getActivity()).addRecipe(recipe);
131                  Intent intent= RecipeActivity.newIntent(getActivity()
     ,recipe.getId());
132                  startActivity(intent);
133                  return true;
134              case R.id.list_logout:
135                  mSession.setReqNewSession(true);
136                  Intent intent2=new Intent(getActivity().
     getApplicationContext(), SplashActivity.class);
137                  startActivity(intent2);
138                  return true;
139              case R.id.list_sync:
140                  AsyncCallClass instanceAsync = new AsyncCallClass(
     getContext());
141                  instanceAsync.execute();
142                  return true;
143              case R.id.new_mail:
144                  Intent intent3= RecipeMailDisplayActivity.newIntent(
     getActivity());
145                  startActivity(intent3);
146                  return true;
```

```
147             default:
148                 return super.onOptionsItemSelected(item);
149         }
150     }
151
152     private void updateUI() {
153         CookBook cookbook=CookBook.get(getActivity());
154         List<Recipe> recipes_in=cookbook.getRecipes();
155         List<Recipe> recipes=new ArrayList<>();
156         recipes.addAll(recipes_in);
157         for(Recipe r:recipes_in){
158             if (!r.IsVisible()){
159                 recipes.remove(r);
160             }
161         }
162         if ((mSortOption & maskSortTitle) == maskSortTitle) {
163             Collections.sort(recipes,
164                     (r1, r2)->{return(r1.getTitle().compareTo(r2.
    getTitle()));});}
165         if ((mSortOption & maskSortNote) == maskSortNote) {
166             Collections.sort(recipes,
167                     (r1, r2)->{return((int)(r2.getNoteAvg()-r1.
    getNoteAvg()));});}
168         if ((mSortOption & maskSortSource) == maskSortSource) {
169             Collections.sort(recipes,
170                     (r1, r2)->{return(r1.getOwner().getName().
    compareTo(r2.getOwner().getName()));});}
171         if ((mSortOption & maskSortSeason) == maskSortSeason) {
172             Collections.sort(recipes,
173                     (r1, r2)->{return(RecipeSeason.getIndex(r1.
    getSeason())
174                             -RecipeSeason.getIndex(r2.getSeason()));}
    );}
175         if ((mSortOption & maskSortDifficulty) == maskSortDifficulty)
     {
176             Collections.sort(recipes,
177                     (r1, r2)->{return(RecipeDifficulty.getIndex(r1.
    getDifficulty())
178                             -RecipeDifficulty.getIndex(r2.
    getDifficulty()));});}
179         if (mAdapter==null){
180             mAdapter=new RecipeAdapter(recipes);
181             mRecipeRecyclerView.setAdapter(mAdapter);
182         } else {
183             mAdapter.setRecipes(recipes);
184             mAdapter.notifyDataSetChanged();
185         }
186     }
187
188     private class RecipeHolder extends RecyclerView.ViewHolder
189                                 implements View.OnClickListener {
190         private TextView mTitleTextView;
191         private TextView mSourceTextView;
192         private TextView mNoteTextView;
```

```
193         private TextView mDifficulty;
194         private ImageView mEditIcon;
195         private ImageView mPhotoView;
196         private ImageView mSunIcon;
197         private ImageView mIceIcon;
198         private File mPhotoFile;
199         private Recipe mRecipe;
200         private RatingBar mRating;
201         public RecipeHolder(LayoutInflater inflater,ViewGroup parent)
    {
202             super(inflater.inflate(R.layout.list_item_recipe, parent,
     false));
203             itemView.setOnClickListener(this);
204             mTitleTextView= (TextView) itemView.findViewById(R.id.
    recipe_title);
205             mSourceTextView= (TextView) itemView.findViewById(R.id.
    recipe_source);
206             mNoteTextView= (TextView) itemView.findViewById(R.id.
    recipe_note);
207             mEditIcon=(ImageView) itemView.findViewById(R.id.
    recipe_edit);
208             mSunIcon=(ImageView) itemView.findViewById(R.id.
    recipe_img_sun);
209             mIceIcon=(ImageView) itemView.findViewById(R.id.
    recipe_img_ice);
210             mDifficulty=(TextView) itemView.findViewById(R.id.
    recipe_difficulty);
211             mPhotoView=(ImageView) itemView.findViewById(R.id.
    recipe_photo);
212             mPhotoFile=CookBook.get(getActivity()).getPhotoFile(
    mRecipe);
213             mRating=(RatingBar) itemView.findViewById(R.id.
    recipe_list_ratingBar);
214             mRating.setOnClickListener(new View.OnClickListener() {
215                 @Override
216                 public void onClick(View v) {
217                     //Toast.makeText(getActivity(), "Call star",
    Toast.LENGTH_SHORT).show();
218                 }
219             });
220             mTitleTextView.setOnClickListener(new View.
    OnClickListener() {
221                 @Override
222                 public void onClick(View v) {
223                     mSortOption=mSortOption ^ maskSortTitle;
224                     updateUI();
225                 }
226             });
227             mNoteTextView.setOnClickListener(new View.OnClickListener
    (){
228                 @Override
229                 public void onClick(View v){
230                     mSortOption=mSortOption ^ maskSortNote;
231                     updateUI();
```

```
232                 }
233             });
234             mSourceTextView.setOnClickListener(new View.
    OnClickListener() {
235                 @Override
236                 public void onClick(View v) {
237                     mSortOption=mSortOption ^ maskSortSource;
238                     updateUI();
239                 }
240             });
241             /*
242             mEditIcon.setOnClickListener(new View.OnClickListener() {
243                 @Override
244                 public void onClick(View v) {
245                 }
246             });*/
247             mSunIcon.setOnClickListener(new View.OnClickListener() {
248                 @Override
249                 public void onClick(View v) {
250                     mSortOption=mSortOption ^ maskSortSeason;
251                     updateUI();
252                 }
253             });
254             mIceIcon.setOnClickListener(new View.OnClickListener() {
255                 @Override
256                 public void onClick(View v) {
257                     mSortOption=mSortOption ^ maskSortSeason;
258                     updateUI();
259                 }
260             });
261             mDifficulty.setOnClickListener(new View.OnClickListener()
     {
262                 @Override
263                 public void onClick(View v) {
264                     mSortOption=mSortOption ^ maskSortDifficulty;
265                     updateUI();
266                 }
267             });
268             mEditIcon.setOnClickListener(new View.OnClickListener() {
269                 @Override
270                 public void onClick(View v) {
271                     Intent intent= RecipeActivity.newIntent(
    getActivity(),mRecipe.getId());
272                     startActivity(intent);
273                 }
274             });
275             mPhotoView.setOnClickListener(new View.OnClickListener()
    {
276                 @Override
277                 public void onClick(View v) {
278                     Intent intent=RecipeDisplayActivity.newIntent(
    getActivity(),mRecipe.getId());
279                     startActivity(intent);
280                 }
```

```
281             });
282         }
283         public void bind(Recipe recipe){
284             mRecipe=recipe;
285             mTitleTextView.setText(mRecipe.getTitle());
286             mSourceTextView.setText("("+mRecipe.getOwner().getName()+
    ")");
287             // mSourceTextView.setText(mRecipe.getFlag()); for debug
288             mRating.setRating((float) mRecipe.getNoteAvg());
289             DecimalFormat df = new DecimalFormat("#.#");
290             mNoteTextView.setText(df.format(mRecipe.getNoteAvg())+"/5
    ");
291             mPhotoFile=CookBook.get(getActivity()).getPhotoFile(
    mRecipe);
292             int idff= RecipeDifficulty.getIndex(mRecipe.getDifficulty
    ());
293             String[] stringArray = getResources().getStringArray(R.
    array.recipe_difficulty_array);
294             mDifficulty.setText(stringArray[idff]);
295             if (mPhotoFile==null || !mPhotoFile.exists()){
296                 mPhotoView.setImageDrawable(getResources().
    getDrawable(R.drawable.ic_recipe_see));
297             } else {
298                 Bitmap bitmap=PictureUtils.getScaledBitmap(mPhotoFile
    .getPath(), getActivity());
299                 mPhotoView.setImageBitmap(bitmap);
300             }
301             mIceIcon.setImageResource((mRecipe.IsWinter()) ? R.
    drawable.ic_recipe_ice : R.drawable.ic_recipe_ice_disabled);
302             mSunIcon.setImageResource((mRecipe.IsSummer()) ? R.
    drawable.ic_recipe_sun : R.drawable.ic_recipe_sun_disabled);
303             mEditIcon.setVisibility((mRecipe.getOwner().getId().
    equals(mSession.getUser().getId())) ? View.VISIBLE : View.GONE);
304         }
305
306         @Override
307         public void onClick(View v){
308             //todo P2 devrait se declencher sur le rating
309             FragmentManager manager = getFragmentManager();
310             RatePickerFragment dialog = RatePickerFragment.
    newInstance(mRecipe.getId());
311             dialog.setTargetFragment(RecipeListFragment.this,
    REQUEST_RATE);
312             dialog.show(manager, DIALOG_RATE);
313         }
314     }
315
316     private class RecipeAdapter extends RecyclerView.Adapter<
    RecipeHolder> {
317         private List<Recipe> mRecipes;
318         public RecipeAdapter(List<Recipe> recipes){
319             mRecipes=recipes;
320         }
321
```

```
322         @NonNull
323         @Override
324         public RecipeHolder onCreateViewHolder(@NonNull ViewGroup
    parent, int viewType) {
325             LayoutInflater layoutInflater=LayoutInflater.from(
    getActivity());
326             return new RecipeHolder(layoutInflater, parent);
327         }
328
329         @Override
330         public void onBindViewHolder(@NonNull RecipeHolder
    recipeHolder, int i) {
331             Recipe recipe=mRecipes.get(i);
332             recipeHolder.bind(recipe);
333         }
334
335         @Override
336         public int getItemCount() {
337             return mRecipes.size();
338         }
339
340         public void setRecipes(List<Recipe> recipes){
341             mRecipes=recipes;
342         }
343     }
344 }
345
```

```java
package com.example.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.example.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) {
                Log.d(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++){
```

```
49              ing[i]=getString(getColumnIndex(RecipeTable.Cols.ING[i]))
    ;
50              r.setIngredient(i+1, ing[i]);
51          }
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 comments = getString(getColumnIndex(RecipeTable.Cols.
    COMMENTS));
57          r.getCommentsDeserialised(comments);
58          String status = getString(getColumnIndex(RecipeTable.Cols.
    STATUS));
59          r.setStatus(StatusRecipe.valueOf(status));
60          String notes = getString(getColumnIndex(RecipeTable.Cols.
    NOTES));
61          r.getNotesDeserialised(notes);
62          String message = getString(getColumnIndex(RecipeTable.Cols.
    MESSAGE));
63          r.setMessage(message);
64          String fromString = getString(getColumnIndex(RecipeTable.Cols
    .MESSAGE_FROM));
65          r.getFromDeserialized(fromString);
66          int tsrecipe = getInt(getColumnIndex(RecipeTable.Cols.
    TS_RECIPE));
67          r.updateTS(AsynCallFlag.NEWRECIPE, (tsrecipe==1));
68          int tsphoto = getInt(getColumnIndex(RecipeTable.Cols.TS_PHOTO
    ));
69          r.updateTS(AsynCallFlag.NEWPHOTO, (tsphoto==1));
70          int tscomment = getInt(getColumnIndex(RecipeTable.Cols.
    TS_COMMENT));
71          r.updateTS(AsynCallFlag.NEWCOMMENT, (tscomment==1));
72          int tsnote = getInt(getColumnIndex(RecipeTable.Cols.TS_NOTE))
    ;
73          r.updateTS(AsynCallFlag.NEWRATING, (tsnote==1));
74          return r;
75      }
76  }
77
```

```java
package com.example.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.example.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.example.cookbook;

import android.content.Intent;
import android.content.pm.PackageManager;
import android.graphics.Bitmap;
import android.os.Bundle;
import android.provider.MediaStore;
import android.support.v4.app.Fragment;
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.EditText;
import android.widget.ImageView;
import android.widget.RatingBar;
import android.widget.ScrollView;
import android.widget.Spinner;
import android.widget.TextView;

import java.io.File;
import java.text.DecimalFormat;
import java.util.UUID;

public class RecipeDisplayFragment extends Fragment {
    private static final String ARG_RECIPE_ID="recipe_id";
    private static final String TAG = "CB_RecipeDisplayFrag";
    private static final int REQUEST_PHOTO= 2;
    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 ImageView mSunIcon;
    private ImageView mIceIcon;
    private TextView mDSourceText;
    private TextView mDSourceUrl;
    private TextView mDIngTitle;
    private TextView mDComTitle;
    private TextView[] mDStepText;
    private TextView[] mDIngText;
```

```
55      private TextView[] mDComText;
56      private EditText mDNexComment;
57      private ImageView mDEnterComment;
58      private ScrollView mScroll;
59
60      public static RecipeDisplayFragment newInstance(UUID recipeId) {
61          Bundle args=new Bundle();
62          args.putSerializable(ARG_RECIPE_ID, recipeId);
63          RecipeDisplayFragment fragment=new RecipeDisplayFragment();
64          fragment.setArguments(args);
65          return fragment;
66      }
67
68      @Override
69      public void onCreate(Bundle savedInstanceState) {
70          super.onCreate(savedInstanceState);
71          mRecipe=new Recipe();
72          UUID recipeId=(UUID) getArguments().getSerializable(
    ARG_RECIPE_ID);
73          mRecipe=CookBook.get(getActivity()).getRecipe(recipeId);
74          mPhotoFile=CookBook.get(getActivity()).getPhotoFile(mRecipe);
75          mSession= SessionInfo.get(getActivity());
76          setHasOptionsMenu(true);
77          mStepNb=mRecipe.getNbStep();
78          mIngNb=mRecipe.getNbIng();
79      }
80      @Override
81      public void onPause(){
82          super.onPause();
83          CookBook.get(getActivity()).updateRecipe(mRecipe);
84      }
85
86      @Override
87      public void onCreateOptionsMenu(Menu menu, MenuInflater inflater)
     {
88          super.onCreateOptionsMenu(menu, inflater);
89          inflater.inflate(R.menu.fragment_recipe_display, menu);
90          MenuItem menuItem = menu.findItem(R.id.recipe_menu_edit);
91          if(!mRecipe.getOwner().getId().equals(mSession.getUser().
    getId())){
92              //getActivity().invalidateOptionsMenu();
93              menuItem.setVisible(false);
94          }
95      }
96
97      @Override
98      public boolean onOptionsItemSelected(MenuItem item) {
99          switch (item.getItemId()) {
100             case R.id.recipe_menu_report:
101                 Intent i=new Intent(Intent.ACTION_SEND);
102                 i.setType("text/plain");
103                 i.putExtra(Intent.EXTRA_TEXT, getRecipeReport());
104                 i.putExtra(Intent.EXTRA_SUBJECT,getString(R.string.
    recipe_report_subject));
```

```
105                 startActivity(i);
106                 return true;
107             case R.id.recipe_menu_edit:
108                 Intent intent= RecipeActivity.newIntent(getActivity()
    ,mRecipe.getId());
109                 startActivity(intent);
110                 return true;
111             case R.id.recipe_mail:
112                 // call
113                 return true;
114             case R.id.recipe_menu_delete:
115                 CookBook.get(getActivity()).markRecipeToDelete(
    mRecipe);
116                 getActivity().onBackPressed();
117             default:
118                 return super.onOptionsItemSelected(item);
119         }
120     }
121
122     @Override
123     public View onCreateView(LayoutInflater inflater, ViewGroup
    container, Bundle savedInstanceState){
124         final View v=inflater.inflate(R.layout.
    fragment_display_recipe, container, false);
125         mScroll=(ScrollView) v.findViewById(R.id.
    fragment_recipe_scroll);
126         mDTitleText =(TextView) v.findViewById(R.id.
    recipe_display_title);
127         mDPhotoView=(ImageView) v.findViewById(R.id.
    recipe_display_photo);
128         updatePhotoView();
129         mDRatingBar=(RatingBar) v.findViewById(R.id.
    recipe_display_ratingBar);
130         mDRatingBarText=(TextView) v.findViewById(R.id.
    recipe_display_ratingBar_txt);
131         mDDifficulty=(TextView) v.findViewById(R.id.
    recipe_display_difficulty);
132         mSunIcon=(ImageView) v.findViewById(R.id.recipe_img_sun) ;
133         mIceIcon=(ImageView) v.findViewById(R.id.recipe_img_ice) ;
134         mDAuthorText=(TextView) v.findViewById(R.id.
    recipe_display_author);
135         mDSourceText=(TextView) v.findViewById(R.id.
    recipe_display_source);
136         mDSourceUrl=(TextView) v.findViewById(R.id.
    recipe_display_source_url);
137         mDIngTitle=(TextView) v.findViewById(R.id.
    recipe_display_title_ing);
138         mDComTitle=(TextView)v.findViewById(R.id.
    recipe_display_comment_title);
139         final int[] rIngID= {R.id.recipe_display_I01,R.id.
    recipe_display_I02,R.id.recipe_display_I03,
140                 R.id.recipe_display_I04,R.id.recipe_display_I05,R.id.
    recipe_display_I06,
141                 R.id.recipe_display_I07,R.id.recipe_display_I08,R.id.
```

```
141 recipe_display_I09,
142                 R.id.recipe_display_I10,R.id.recipe_display_I11,R.id.
    recipe_display_I12,
143                 R.id.recipe_display_I13,R.id.recipe_display_I14,R.id.
    recipe_display_I15};
144         mDIngText= new TextView[mRecipe.getNbIngMax()];
145         for(int i=0;i<mRecipe.getNbIngMax();i++) {
146             mDIngText[i] = (TextView) v.findViewById(rIngID[i]);
147         }
148         final int[] rID= {R.id.recipe_display_S1,R.id.
    recipe_display_S2,R.id.recipe_display_S3,
149                 R.id.recipe_display_S4,R.id.recipe_display_S5,R.id.
    recipe_display_S6,
150                 R.id.recipe_display_S7,R.id.recipe_display_S8,R.id.
    recipe_display_S9};
151         mDStepText=new TextView[mRecipe.getNbStepMax()];
152         for(int i=0;i<mRecipe.getNbStepMax();i++) {
153             mDStepText[i] = (TextView) v.findViewById(rID[i]);
154         }
155         final int[] rComID= {R.id.recipe_display_C01,R.id.
    recipe_display_C02,R.id.recipe_display_C03,
156                 R.id.recipe_display_C04,R.id.recipe_display_C05,R.id.
    recipe_display_C06,
157                 R.id.recipe_display_C07,R.id.recipe_display_C08,R.id.
    recipe_display_C09,
158                 R.id.recipe_display_C10,R.id.recipe_display_C11,R.id.
    recipe_display_C12,
159                 R.id.recipe_display_C13,R.id.recipe_display_C14,R.id.
    recipe_display_C15,
160                 R.id.recipe_display_C16,R.id.recipe_display_C17,R.id.
    recipe_display_C18,
161                 R.id.recipe_display_C19,R.id.recipe_display_C20};
162         mDComText=new TextView[mRecipe.getNbComMax()];
163         for(int i=0;i<mRecipe.getNbComMax();i++) {
164             mDComText[i] = (TextView) v.findViewById(rComID[i]);
165         }
166         updateUI();
167         mDNexComment=(EditText) v.findViewById(R.id.
    recipe_display_enter_comment);
168         mDNexComment.setText("");
169         mDNexComment.setHint(getString(R.string.recipe_comment_hint))
    ;
170         //todo test comment versus pattern
171         mDNexComment.addTextChangedListener(new TextWatcher() {
172             @Override
173             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) {
174
175             }
176
177             @Override
178             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
179                 newcomment=s.toString();
```

```
180             }
181
182             @Override
183             public void afterTextChanged(Editable s) {
184
185             }
186         });
187         mDEnterComment=(ImageView) v.findViewById(R.id.
    recipe_img_enter);
188         mDEnterComment.setOnClickListener(new View.OnClickListener()
    {
189             @Override
190             public void onClick(View v) {
191                 mRecipe.addComment(new Comment(newcomment, mSession.
    getUser()));
192                 mRecipe.updateTS(AsynCallFlag.NEWCOMMENT,true);
193                 mDNexComment.setText("...");
194                 updateUI();
195             }
196         });
197         return v;
198     }
199     private void updatePhotoView(){
200         if (mPhotoFile==null || !mPhotoFile.exists()){
201             mDPhotoView.setImageDrawable(getResources().getDrawable(R
    .drawable.ic_recipe_camera));
202         } else {
203             Bitmap bitmap=PictureUtils.getScaledBitmap(mPhotoFile.
    getPath(), getActivity());
204             mDPhotoView.setImageBitmap(bitmap);
205         }
206     }
207     private void updateUI(){
208         String s;
209         User u=mRecipe.getOwner();
210         int ingMax=mRecipe.getNbIngMax();
211         int stepMax=mRecipe.getNbStepMax();
212         int comMax=mRecipe.getNbComMax();
213         int NbCom=mRecipe.getComments().size();
214         int gone=View.GONE;
215         int visible=View.VISIBLE;
216         DecimalFormat df = new DecimalFormat("#.#");
217         mDTitleText.setText(mRecipe.getTitle());
218         mDRatingBar.setRating((float) mRecipe.getNoteAvg());
219         mDRatingBarText.setText(df.format(mRecipe.getNoteAvg())+"  ("
    + mRecipe.getNotes().size()+")");
220         s=getString(R.string.recipe_display_author,u.getName(),u.
    getFamily());
221         int idiff= RecipeDifficulty.getIndex(mRecipe.getDifficulty())
    ;
222         String[] stringArrayDiff = getResources().getStringArray(R.
    array.recipe_difficulty_array);
223         mDDifficulty.setText(stringArrayDiff[idiff]);
224         mSunIcon.setImageResource((mRecipe.IsSummer()) ? R.drawable.
```

```
224 ic_recipe_sun : R.drawable.ic_recipe_sun_disabled);
225         mIceIcon.setImageResource((mRecipe.IsWinter()) ? R.drawable.
    ic_recipe_ice : R.drawable.ic_recipe_ice_disabled);
226         mDAuthorText.setText(s);
227         mDSourceText.setText(mRecipe.getSource());
228         if (mRecipe.getSource_url_name().equals(DEFAULT_URL)) {
229             mDSourceUrl.setText(""); }
230         else {mDSourceUrl.setText(mRecipe.getSource_url_name());}
231         s=getString(R.string.recipe_display_title_ing, ""+mRecipe.
    getNbPers());
232         mDIngTitle.setText(s);
233         for(int i=0;i<ingMax;i++){
234             if (mIngNb>0){mDIngText[i].setText("- "+mRecipe.
    getIngredient(i+1));}
235             if (i>=0){mDIngText[i].setVisibility((mIngNb>i)? visible:
    gone);}
236         }
237         for(int i=0;i<stepMax;i++){
238             if (mStepNb>0){mDStepText[i].setText((i+1)+". "+mRecipe.
    getStep(i+1));}
239             if (i>=0){mDStepText[i].setVisibility((mStepNb>i)?
    visible:gone);}
240         }
241         s=getString(R.string.recipe_display_title_comment, ""+mRecipe
    .getComments().size());
242         mDComTitle.setText(s);
243         for(int i=0;i<comMax;i++){
244             if (NbCom>0){
245                 if (i<NbCom) {mDComText[i].setText("- "+mRecipe.
    getComment(NbCom-i-1).toTxt());}
246                 else {mDComText[i].setText("");}
247             }
248             if (i>=0){mDComText[i].setVisibility((NbCom>i)? visible:
    gone);}
249         }
250     }
251     private String getRecipeReport(){
252         String report;
253         Integer iplus;
254         report =getString(R.string.recipe_report_title, mRecipe.
    getTitle())+"\n";
255         report +=getString(R.string.recipe_report_owner,mRecipe.
    getOwner().getNameComplete())+"\n";
256         //String dateFormat = "dd MMM yyyy";
257         //String dateString=DateFormat.format(dateFormat,mRecipe.
    getDate()).toString();
258         if(!mRecipe.getSource_url_name().equals("")){
259             report += getString(R.string.recipe_report_url, mRecipe.
    getSource_url_name())+"\n";
260         }
261         for(int i=0;i<mRecipe.getNbIng();i++){
262             report += getString(R.string.recipe_report_ing, mRecipe.
    getIngredient(i+1))+"\n";
263         }
```

```
264          for(int i=0;i<mRecipe.getNbStep();i++){
265              iplus=i+1;
266              report += getString(R.string.recipe_report_step, iplus+""
    ,
267                      mRecipe.getStep(i+1))+"\n";
268          }
269          report +=getString(R.string.recipe_report_final);
270          return report;
271      }
272
273  }
274
```

```java
package com.example.cookbook;

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

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();
        }
    }
}

```

```
1  package com.example.cookbook;
2
3  import android.content.Context;
4  import android.content.Intent;
5  import android.support.v4.app.Fragment;
6
7  public class RecipeMailDisplayActivity extends SingleFragmentActivity
   {
8      //private static final String EXTRA_RECIPE_ID="com.example.
   cookbook.recipe_id";
9      public static Intent newIntent(Context packageContexte) {
10         Intent intent=new Intent(packageContexte,
   RecipeMailDisplayActivity.class);
11         //intent.putExtra(EXTRA_RECIPE_ID, recipeId);
12         return intent;
13     }
14
15     @Override
16     protected Fragment createFragment() {
17         //UUID recipeId=(UUID) getIntent().getSerializableExtra(
   EXTRA_RECIPE_ID);
18         return RecipeMailDisplayFragment.newInstance();
19     }
20 }
21
```

```java
package com.example.cookbook;

import android.graphics.Bitmap;
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 java.io.File;
import java.util.ArrayList;
import java.util.List;

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

    public static RecipeMailDisplayFragment newInstance() {
        //Bundle args=new Bundle();
        //args.putSerializable(ARG_RECIPE_ID, recipeId);
        RecipeMailDisplayFragment fragment=new
RecipeMailDisplayFragment();
        //fragment.setArguments(args);
        return fragment;
    }

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

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
container,
                             Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_recipe_list,
container, false);
        mRecipeRecyclerView = (RecyclerView) view.findViewById(R.id.
recipe_recycler_view);
        mRecipeRecyclerView.setLayoutManager(new LinearLayoutManager(
getActivity()));
        updateUI();
        return view;
    }
```

```
50
51      @Override
52      public void onResume(){
53          super.onResume();
54          updateUI();
55      }
56
57      private void updateUI() {
58          CookBook cookbook=CookBook.get(getActivity());
59          List<Recipe> recipes_in=cookbook.getRecipes();
60          List<Recipe> recipes=new ArrayList<>();
61          for(Recipe r:recipes_in){
62              if (r.IsMessage()) recipes.add(r);
63          }
64          if (mAdapter==null){
65              mAdapter=new RecipeMailDisplayFragment.RecipeAdapter(
    recipes);
66              mRecipeRecyclerView.setAdapter(mAdapter);
67          } else {
68              mAdapter.setRecipes(recipes);
69              mAdapter.notifyDataSetChanged();
70          }
71      }
72  // ------------------------------------RECIPE HOLDER
     ------------------------
73      private class RecipeHolder extends RecyclerView.ViewHolder
74              implements View.OnClickListener {
75          private TextView mTitleTextView;
76          private TextView mMessage;
77          private TextView mFrom;
78          private ImageView mPhotoView;
79          private ImageView mDelete;
80          private ImageView mAdd;
81          private File mPhotoFile;
82          private Recipe mRecipe;
83
84          public RecipeHolder(LayoutInflater inflater,ViewGroup parent)
    {
85              super(inflater.inflate(R.layout.list_item_mail_display,
    parent, false));
86              itemView.setOnClickListener(this);
87              mTitleTextView= (TextView) itemView.findViewById(R.id.
    recipe_MD_title);
88              mMessage=(TextView) itemView.findViewById(R.id.
    mail_display_message);
89              mFrom=(TextView) itemView.findViewById(R.id.
    mail_display_author);
90              mDelete=(ImageView) itemView.findViewById(R.id.
    mail_display_delete);
91              mAdd=(ImageView) itemView.findViewById(R.id.
    mail_display_add);
92              mPhotoView=(ImageView) itemView.findViewById(R.id.
    recipe_MD_photo);
93              mPhotoFile=CookBook.get(getActivity()).getPhotoFile(
```

```
 93 mRecipe);
 94
 95             mDelete.setOnClickListener(new View.OnClickListener() {
 96                 @Override
 97                 public void onClick(View v) {
 98                     Log.d(TAG, "Delete :" + mRecipe.getTitle());
 99                     updateUI();
100                 }
101             });
102             mAdd.setOnClickListener(new View.OnClickListener() {
103                 @Override
104                 public void onClick(View v) {
105                     Log.d(TAG, "Add :" + mRecipe.getTitle());
106                     updateUI();
107                 }
108             });
109
110         }
111         public void bind(Recipe recipe){
112             mRecipe=recipe;
113             mTitleTextView.setText(mRecipe.getTitle());
114             mMessage.setText(mRecipe.getMessage());
115             mFrom.setText(mRecipe.getUserFrom().getNameComplete());
116             mPhotoFile=CookBook.get(getActivity()).getPhotoFile(
    mRecipe);
117             if (mPhotoFile==null || !mPhotoFile.exists()){
118                 mPhotoView.setImageDrawable(getResources().
    getDrawable(R.drawable.ic_recipe_see));
119             } else {
120                 Bitmap bitmap=PictureUtils.getScaledBitmap(mPhotoFile
    .getPath(), getActivity());
121                 mPhotoView.setImageBitmap(bitmap);
122             }
123         }
124     @Override
125     public void onClick(View v){    }
126     }
127     // ----------------------------------ADAPTER
    ------------------------
128     private class RecipeAdapter extends RecyclerView.Adapter<
    RecipeMailDisplayFragment.RecipeHolder> {
129         private List<Recipe> mRecipes;
130         public RecipeAdapter(List<Recipe> recipes){
131             mRecipes=recipes;
132         }
133
134         @NonNull
135         @Override
136         public RecipeMailDisplayFragment.RecipeHolder
    onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
137             LayoutInflater layoutInflater=LayoutInflater.from(
    getActivity());
138             return new RecipeMailDisplayFragment.RecipeHolder(
    layoutInflater, parent);
```

```
139         }
140
141         @Override
142         public void onBindViewHolder(@NonNull
    RecipeMailDisplayFragment.RecipeHolder recipeHolder, int i) {
143             Recipe recipe=mRecipes.get(i);
144             recipeHolder.bind(recipe);
145         }
146
147         @Override
148         public int getItemCount() {
149             return mRecipes.size();
150         }
151
152         public void setRecipes(List<Recipe> recipes){
153             mRecipes=recipes;
154         }
155     }
156
157 }
158
```