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

```java
package com.fdx.cookbook;

import java.util.Date;

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


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

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

    public Integer getNote() {
        return mNote;
    }

    public User getUser() {
        return mUser;
    }

    public Date getDate() {
        return mDate;
    }

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

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

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

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

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

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

```java
package com.fdx.cookbook;

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

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

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

    public User(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;
    }

    public void setFamily(String family) {
```

```
55          mFamily = family;
56      }
57
58      public String getName() {
59          return mName;
60      }
61
62      public void setName(String name) {
63          mName = name;
64      }
65
66      public String getNameComplete() {
67          return mName+"@"+ mFamily;
68      }
69
70      public boolean IsEqual(User r){return (mId.toString().equals(r.
    getId().toString()));}
71
72      public Date getDate() {
73          return mDate;
74      }
75
76      public void setDate(Date date) {
77          mDate = date;
78      }
79
80      @Override
81      public boolean equals(Object o){
82          if (o == null) return false;
83          if (o == this) return true;
84          if (getClass() != o.getClass()) return false;
85          User u=(User) o;
86          return u.getId().toString().equals(mId.toString());
87      }
88      //todo P2 épurer affichage quand champs vides
89      //todo P2 tuto swipe page
90      //todo P1 messagerie
91
92  }
93
```

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

```java
package com.fdx.cookbook;

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

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


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

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



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

    public Recipe( UUID id){
```

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

```
109     }
110
111     public String getMessage() {
112         return mMessage;
113     }
114
115     public void setMessage(String message) {
116         mMessage=message;
117     }
118
119
120     // -----------------STEP-------------------
121     public void setStep(Integer i, String step) {
122         if ((i > 0) && (i <= NBSTEP_MAX)) {
123             mSteps[i - 1] = step;
124         }
125     }
126
127     public String getStep(Integer i){
128         if ((i>0)&&(i<=NBSTEP_MAX)) {return mSteps[i-1];}
129         else{ return "";}
130     }
131
132     public int getNbStep(){
133         int j=0;
134         for(int i=NBSTEP_MAX; i>0; i--){
135             if (!mSteps[i-1].isEmpty()) {j=i; break;}}
136         return j;
137     }
138
139     public int getNbStepMax(){return NBSTEP_MAX;}
140
141     //--------------Ingredients--------------------
142     public void setIngredient(Integer i, String ing) {
143         if ((i > 0) && (i <= NBING_MAX)) {
144             mIngredients[i-1] = ing;
145         }
146     }
147
148     public String getIngredient(Integer i){
149         if ((i>0)&&(i<=NBING_MAX)) {return mIngredients[i-1];}
150         else{ return "";}
151     }
152     public int getNbIng(){
153         int j=0;
154         for(int i=NBING_MAX; i>0; i--){
155             if (!mIngredients[i-1].isEmpty()) {j=i; break;}}
156         return j;
157     }
158
159     public int getNbIngMax(){return NBING_MAX;}
160
161     //------------------- Users  ----------------------------------
162     public UUID getId() {
```

```
163         return mId;
164     }
165
166     public User getUserFrom() {
167         return mIdFrom;
168     }
169
170     public void setUserFrom(User idFrom) {
171         mIdFrom = idFrom;
172     }
173
174     public void setId(UUID id) {mId = id; }
175
176     public User getOwner() {
177         return mOwner;
178     }
179
180     public String getOwnerIdString() {
181         return mOwner.getId().toString();
182     }
183
184     public void setOwner(User owner) {
185         mOwner = owner;
186     }
187
188     //------------------- Source ----------------------------------
189     public String getSource() {
190         if (mSource==null) return "";
191         return mSource;
192     }
193     public void setSource(String source) {
194         mSource = source;
195     }
196     public URL getSource_url() {
197         return mSource_url;
198     }
199
200     public String getSource_url_name() {
201         String ret=mSource_url.toString();
202         if (ret.equals("https:") || ret.equals("http:")){ret="";}
203         return ret;
204     }
205
206     public void setSource_url(URL source_url) {
207         mSource_url = source_url;
208     }
209
210     // ------------------------Enum -------------------------m
211
212     public RecipeSeason getSeason() {
213         return mSeason;
214     }
215
216     public void setSeason(RecipeSeason season) {
```

```
217         mSeason = season;
218     }
219
220     public boolean IsSummer(){
221         switch (mSeason) {
222             case SUMMER:
223                 return true;
224             case ALLYEAR:
225                 return true;
226             default:
227                 return false;
228         }
229     }
230     public boolean IsWinter(){
231         switch (mSeason) {
232             case WINTER:
233                 return true;
234             case ALLYEAR:
235                 return true;
236             default:
237                 return false;
238         }
239     }
240     public RecipeType getType() {
241         return mType;
242     }
243     public void setType(RecipeType type) {
244         mType = type;
245     }
246
247     public RecipeDifficulty getDifficulty() {
248         return mDifficulty;
249     }
250
251     public void setDifficulty(RecipeDifficulty difficulty) {
252         mDifficulty = difficulty;
253     }
254
255     public StatusRecipe getStatus() {return mStatus;}
256     public void setStatus(StatusRecipe s) {
257         mStatus = s;
258     }
259     public boolean IsVisible(){
260         if (mStatus==StatusRecipe.Visible) {return true;}
261         return false;
262     }
263     public boolean IsMessage(){
264         if (mStatus==StatusRecipe.Submitted) {return true;}
265         return false;
266     }
267     public boolean IsMarkedDeleted(){
268         if (mStatus==StatusRecipe.Deleted) {return true;}
269         return false;
270     }
```

```
271
272     public void updateTS(AsynCallFlag asyn, Boolean b){
273         switch(asyn){
274             case NEWRECIPE:{mTS_recipe=b;return;}
275             case NEWPHOTO:{mTS_photo=b;return;}
276             case NEWRATING:{mTS_note=b;return;}
277             case NEWCOMMENT:{mTS_comment=b;return;}
278         }
279         return;
280     }
281
282     public int getTS(AsynCallFlag asyn){
283         Boolean b=false;
284         switch(asyn){
285             case NEWRECIPE:{b=mTS_recipe; break;}
286             case NEWPHOTO:{b=mTS_photo; break;}
287             case NEWRATING:{b=mTS_note; break;}
288             case NEWCOMMENT:{b=mTS_comment; break;}
289         }
290         return (b ? 1:0);
291     }
292
293     //-------------------- photo filename--------------------
294     public String getPhotoFilename(){
295         return "IMG"+getId().toString()+".jpg";
296     }
297
298     //---------------------- Arraylist Comments et Notes
    -------------------------
299     public void addComment(Comment c){ mComments.add(c);}
300
301     public ArrayList<Comment> getComments() {return mComments;}
302
303     public Comment getComment(int i){
304         if (i<mComments.size()){
305             return mComments.get(i);
306         } else {return null;}
307
308     }
309
310     public int getNbComMax(){return NBCOM_MAX;} // nb max affiché
311
312     public void addNote(Note note){ mNotes.add(note);}
313
314     public ArrayList<Note> getNotes() {return mNotes;}
315
316     public Note getNote(int i){
317         if (mNotes==null) return null;
318         if (i<mNotes.size()){ return mNotes.get(i);}
319         else {return null;}
320     }
321
322     public double getNoteAvg() {
323         mNoteAvg=0;
```

```
324         if (mNotes==null) return mNoteAvg;
325         if (mNotes.size()!=0) {
326             for(Note n:mNotes){mNoteAvg+=n.getNote();}
327             mNoteAvg=mNoteAvg/mNotes.size();
328         } else {mNoteAvg=0;}
329         return mNoteAvg;
330     }
331     public int getNbNotes() {
332         if (mNotes==null) return 0;
333         if (mNotes.isEmpty()) return 0;
334         return mNotes.size();
335     }
336
337     //---------------------- Serialisation ------------------------
338     public String getSerializedComments(){
339         Gson gson = new Gson();
340         return gson.toJson(mComments);
341     }
342
343     public void getCommentsDeserialised(String raw){
344         Gson gson=new Gson();
345         Type listOfNotesObject = new TypeToken<ArrayList<Comment>>()
    {}.getType();
346         mComments=gson.fromJson(raw, listOfNotesObject);
347     }
348     public String getSerializedOwner(){
349         Gson gson = new Gson();
350         return gson.toJson(mOwner);
351     }
352     public String getSerializedFrom(){
353         Gson gson = new Gson();
354         return gson.toJson(mIdFrom);
355     }
356     public void getOwnerDeserialized(String raw){
357         Gson gson=new Gson();
358         mOwner=gson.fromJson(raw, User.class);
359     }
360     public void getFromDeserialized(String raw){
361         Gson gson=new Gson();
362         mIdFrom=gson.fromJson(raw, User.class);
363     }
364
365     public String getSerializedNotes(){
366         Gson gson = new Gson();
367         return gson.toJson(mNotes);
368     }
369
370     public void getNotesDeserialised(String raw){
371         Gson gson=new Gson();
372         Type listOfNotesObject = new TypeToken<ArrayList<Note>>() {}.
    getType();
373         mNotes=gson.fromJson(raw, listOfNotesObject);
374     }
375
```

```
376     // ******************* tests **********************
377     public Boolean hasNotChangedSince(Recipe r){
378         if (!mId.toString().equals(r.getId().toString())) return
    false;
379         if (!mOwner.getId().toString().equals(r.getOwner().getId().
    toString())) return false;
380         if (!mTitle.equals(r.getTitle())) return false;
381         if (!mSource.equals(r.getSource())) return false;
382         if (!mSource_url.equals(r.getSource_url())) return false;
383         if (mNbPers!=r.getNbPers()) return false;
384         for (int i = 0; i < r.getNbStepMax(); i++) {
385             if (!mSteps[i].equals(r.getStep(i+1))) return false;
386         }
387         for (int i = 0; i < r.getNbIngMax(); i++) {
388             if (!mIngredients[i].equals(r.getIngredient(i + 1)))
    return false;
389         }
390         if (!mSeason.toString().equals(r.getSeason().toString()))
    return false;
391         if (!mType.toString().equals(r.getType().toString())) return
    false;
392         if (!mDifficulty.toString().equals(r.getDifficulty().toString
    ())) return false;
393         return true;
394     }
395
396     public boolean containQuery(String s){ //for search request
397         if (mTitle.indexOf(s)!=-1) return true;
398         if (mOwner.getFamily().toString().indexOf(s)!=-1) return true
    ;
399         for(String sloop:mIngredients){
400             if (sloop==null) break;
401             if (sloop.indexOf(s)!=-1) return true;
402         }
403         for(String sloop:mSteps){
404             if (sloop==null) break;
405             if (sloop.indexOf(s)!=-1) return true;
406         }
407         for(Comment c:mComments){
408             if (c==null) break;
409             if (c.getTxt().indexOf(s)!=-1) return true;
410         }
411         return false;
412     }
413 }
414
```

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

```java
package com.fdx.cookbook;

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

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

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

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

    public String getTxt() {
        return mTxt;
    }

    public User getUser() {
        return mUser;
    }

    public Date getDate() {
        return mDate;
    }

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

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

    @Override
    public boolean equals(Object o){
        if (o == null) return false;
        if (o == this) return true;
```

```
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
```

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

```java
package com.fdx.cookbook;

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


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

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

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

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

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

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

```
 55
 56     private RecipeCursorWrapper queryRecipes(String whereClause,
    String[] whereArgs){
 57         Cursor cursor=mDatabase.query(
 58                 RecipeDbSchema.RecipeTable.NAME,
 59                 null, // selct all columns
 60                 whereClause,
 61                 whereArgs,
 62                 null, null, null
 63         );
 64         return new RecipeCursorWrapper(cursor);
 65     }
 66
 67     public void addRecipe(Recipe r){
 68         if (this.getRecipe(r.getId())==null) {
 69             ContentValues values=getContentValues(r);
 70             mDatabase.insert(RecipeDbSchema.RecipeTable.NAME, null,
    values);
 71         }
 72     }
 73
 74     public void removeRecipe(Recipe r){
 75         String uuidString=r.getId().toString();
 76         mDatabase.delete(RecipeDbSchema.RecipeTable.NAME,
 77                 RecipeDbSchema.RecipeTable.Cols.UUID+" =?",
 78                 new String[] {uuidString});
 79         return;
 80     }
 81
 82     public void markRecipeToDelete(Recipe r){
 83         r.setStatus(StatusRecipe.Deleted);
 84         updateRecipe(r);
 85         return;
 86     }
 87
 88     public List<Recipe> getRecipes(){
 89           List<Recipe> recipes=new ArrayList<>();
 90           RecipeCursorWrapper cursor=queryRecipes(null, null);
 91           try{
 92               cursor.moveToFirst();
 93               while (!cursor.isAfterLast()){
 94                   recipes.add(cursor.getRecipe());
 95                   cursor.moveToNext();
 96               }
 97           } finally { cursor.close();}
 98           return recipes;
 99     }
100
101     public List<Recipe> getRecipesVisibles(){
102         List<Recipe> recipes=this.getRecipes();
103         List<Recipe> recipeloop=this.getRecipes();
104         for(Recipe r:recipeloop){
105             if (!r.IsVisible()) recipes.remove(r);
106         }
```

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

```
139         return values;
140     }
141
142     public File getPhotoFile(Recipe r){
143         if (r==null) {return null;}
144         File filesDir= mContext.getFilesDir();
145         return new File(filesDir, r.getPhotoFilename());
146     }
147
148     public Boolean deleteImage(Recipe r){
149         Boolean success=false;
150         File filesDir= mContext.getFilesDir();
151         File im=new File(filesDir, r.getPhotoFilename());
152         if (im.exists()){
153             im.delete();
154             success=true;
155         }
156         return success;
157     }
158
159     public void clearCookBook(){
160         List<Recipe> recipes=new ArrayList<>();
161         recipes=getRecipes();
162         for (Recipe r:recipes){
163             deleteImage(r);
164             removeRecipe(r);
165         }
166     }
167
168     public void fillCookBook(List<Recipe> recipes){
169         for(Recipe r:recipes){
170             addRecipe(r);
171         }
172     }
173
174     public Boolean isThereMail(){
175         List<Recipe> recipes=this.getRecipes();
176         for(Recipe r:recipes){
177             if (r.IsMessage()) return true;
178         }
179         return false;
180     }
181 }
182
```

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

```
 1 package com.fdx.cookbook;
 2
 3 import java.util.UUID;
 4
 5 public class ListMask {
 6     private boolean SeasonFiltered;
 7     private RecipeSeason SeasonState;
 8     private boolean DifficultyFiltered;
 9     private RecipeDifficulty DifficultyState;
10     private boolean TypeFiltered;
11     private RecipeType TypeState;
12     private boolean UserFiltered;
13     private User UserState;
14     private boolean IsNoteSorted;
15     private boolean IsSearched;
16     private boolean IsTitleAlphaSorted;
17     private String Query;
18
19     public ListMask(User u){
20         SeasonFiltered=false;
21         DifficultyFiltered=false;
22         TypeFiltered=false;
23         UserFiltered=false;
24         IsNoteSorted=false;
25         IsSearched=false;
26         IsTitleAlphaSorted=false;
27         Query="";
28         UserState=u;
29         SeasonState=RecipeSeason.ALLYEAR;
30         DifficultyState=RecipeDifficulty.UNDEFINED;
31         TypeState=RecipeType.MAIN;
32     }
33
34     public void reset(){
35         SeasonFiltered=false;
36         DifficultyFiltered=false;
37         TypeFiltered=false;
38         UserFiltered=false;
39         IsNoteSorted=false;
40         IsSearched=false;
41         IsTitleAlphaSorted=false;
42         Query="";
43         SeasonState=RecipeSeason.ALLYEAR;
44         DifficultyState=RecipeDifficulty.UNDEFINED;
45         TypeState=RecipeType.MAIN;
46     }
47
48     public Integer toggleSun(){
49         Integer i;
50         if (SeasonFiltered){
51             SeasonFiltered=false;
52             SeasonState=RecipeSeason.ALLYEAR;
53             i=R.string.TSOFF;
54         } else {
```

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

```
109             IsSearched=true;
110             Query=s;
111             i=R.string.TSEON;
112         }
113         return i;
114     }
115
116     public Integer toggleDifficulty(RecipeDifficulty rd){
117         Integer i;
118         if (DifficultyFiltered){
119             DifficultyFiltered=false;
120             DifficultyState=RecipeDifficulty.UNDEFINED;
121             i=R.string.TDOFF;
122         }else{
123             DifficultyFiltered=true;
124             DifficultyState=rd;
125             i=R.string.TDON;
126         }
127         return i;
128     }
129
130     public Integer toggleType(RecipeType rt){
131         Integer i;
132         if (TypeFiltered){
133             TypeFiltered=false;
134             TypeState=RecipeType.MAIN;
135             i=R.string.TTOFF;
136         }else{
137             TypeFiltered=true;
138             TypeState=rt;
139             i=R.string.TTON;
140         }
141         return i;
142     }
143
144     public Integer toggleSortNote(){
145         Integer i;
146         if (IsNoteSorted){
147             IsNoteSorted=false;
148             i=R.string.TNOFF;
149         }else{
150             IsNoteSorted=true;
151             i=R.string.TNON;
152         }
153         return i;
154     }
155
156     public boolean isNoteSorted(){return IsNoteSorted;}
157
158     public boolean isRecipeSelected(Recipe r){
159         boolean b=false;
160         if (!r.IsVisible()) return false;
161         if (SeasonFiltered) {
162             if (SeasonState==RecipeSeason.SUMMER) b=(r.getSeason()==
```

```
RecipeSeason.WINTER);
            if (SeasonState==RecipeSeason.WINTER) b=(r.getSeason()==
RecipeSeason.SUMMER);
            if (b) return false;
        }
        if (DifficultyFiltered){
            if (r.getDifficulty()!=DifficultyState) return false;
        }
        if (TypeFiltered){
            if (r.getType()!=TypeState) return false;
        }
        if (UserFiltered){
            if (!r.getOwner().IsEqual(UserState)) return false;
        }
        if (IsSearched){
            if (!r.containQuery(Query)) return false;
        }
        return true;
    }

    public String convertToString(){
        String s,sep=";";
        s=(SeasonFiltered ? "Y":"N")+sep;    //0
        s+=SeasonState.toString()+sep;       //1
        s+=(DifficultyFiltered ? "Y":"N")+sep;  //2
        s+=DifficultyState.toString()+sep;       //3
        s+=(TypeFiltered ? "Y":"N")+sep;         //4
        s+=TypeState.toString()+sep;             //5
        s+=(IsNoteSorted ? "Y":"N")+sep;         //6
        s+=(IsTitleAlphaSorted ? "Y":"N")+sep;   //7
        s+=(IsSearched ? "Y":"N")+sep;           //8
        s+=Query+sep;                            //9
        s+=(UserFiltered ? "Y":"N")+sep;         //10
        s+=UserState.getFamily()+sep;            //11
        s+=UserState.getName()+sep;              //12
        s+=UserState.getId().toString();         //13
        return s;
    }

    public void updateFromString(String s){
        String sep=";";
        String[] tokens = s.split(sep);
        if (tokens.length<14) return;
        SeasonFiltered=( tokens[0].equals("Y")? true:false);
        SeasonState=RecipeSeason.valueOf(tokens[1]);
        DifficultyFiltered=( tokens[2].equals("Y")? true:false);
        DifficultyState=RecipeDifficulty.valueOf(tokens[3]);
        TypeFiltered=( tokens[4].equals("Y")? true:false);
        TypeState=RecipeType.valueOf(tokens[5]);
        IsNoteSorted=( tokens[6].equals("Y")? true:false);
        IsTitleAlphaSorted=( tokens[7].equals("Y")? true:false);
        IsSearched=( tokens[8].equals("Y")? true:false);
        Query=tokens[9];
        UserFiltered=( tokens[10].equals("Y")? true:false);
```

```
215             UserState=new User(tokens[11], tokens[12]);
216             UserState.setId(UUID.fromString(tokens[13]));
217         }
218 }
219
```

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

```
 1 package com.fdx.cookbook;
 2
 3 public enum RecipeType {
 4     APERITIF,STARTER,MAIN,DESSERT,SIDE,OTHER;
 5
 6         private static RecipeType[] list=RecipeType.values();
 7
 8         public static RecipeType getType(int i){
 9             return list[i];
10         }
11
12         public static int getIndex(RecipeType r){
13             for(int i=0;i<list.length;i++) {
14                 if (list[i].equals(r)){ return i;}
15             }
16             return 0;
17         }
18 }
19
```

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

```
 1 package com.fdx.cookbook;
 2
 3 import android.content.Context;
 4 import android.content.SharedPreferences;
 5 import android.preference.PreferenceManager;
 6
 7 import java.util.UUID;
 8
 9 public class SessionInfo {
10     private static SessionInfo ourInstance;
11     private User mUser;
12     private Context mContext;
13     private Boolean mIsConnected;
14     private Boolean mReqNewSession;
15     private String CB_FAMILY="family";
16     private String CB_NAME="name";
17     private String CB_ID="iduser";
18     private String NOT_FOUND="Not found";
19     private String mMaskSerialized;
20     private static final String TAG = "DebugSessionInfo";
21     private static String URLPATH="http://82.66.37.73:8085/cb/";
22     public static int CONNECT_TIMEOUT = 10000;
23     public static int READ_TIMEOUT = 10000;
24
25     public static SessionInfo get(Context context) {
26         if (ourInstance==null){
27             ourInstance= new SessionInfo(context);
28         }
29         return ourInstance;
30     }
31
32     private SessionInfo(Context context) {
33         mContext=context.getApplicationContext();
34         mIsConnected=false;
35         mReqNewSession=false;
36         mMaskSerialized="";
37         String sharedPref;
38         sharedPref= PreferenceManager.getDefaultSharedPreferences(
   context)
39                 .getString(CB_ID, NOT_FOUND);
40         if (sharedPref.equals(NOT_FOUND)){
41             mUser = new User(NOT_FOUND, NOT_FOUND);
42         } else {
43             mUser = new User(PreferenceManager.
   getDefaultSharedPreferences(context)
44                     .getString(CB_FAMILY, NOT_FOUND),
   PreferenceManager.getDefaultSharedPreferences(context)
45                     .getString(CB_NAME, NOT_FOUND));
46             mUser.setId(UUID.fromString(sharedPref));
47         }
48     }
49
50
51     public User getUser() {
```

```
 52          return mUser;
 53      }
 54
 55      public void setStoredUser(User user){
 56          PreferenceManager.getDefaultSharedPreferences(mContext)
 57                  .edit()
 58                  .putString(CB_FAMILY, user.getFamily())
 59                  .putString(CB_NAME, user.getName())
 60                  .putString(CB_ID, user.getId().toString())
 61                  .apply();
 62          mUser=user;
 63      }
 64      public void clearStoredUser(){
 65          SharedPreferences settings = mContext.getSharedPreferences("
    PreferencesName", Context.MODE_PRIVATE);
 66          settings.edit().remove("CB_FAMILY").commit();
 67          settings.edit().remove("CB_NAME").commit();
 68          settings.edit().remove("CB_ID").commit();
 69      }
 70
 71      public Boolean IsEmpty(){
 72          if (mUser.getName().equals(NOT_FOUND)){return true;}
 73          return false;
 74      }
 75
 76      public Boolean IsReqNewSession() {
 77          return mReqNewSession;
 78      }
 79
 80      public void setReqNewSession(Boolean reqNewSession) {
 81          mReqNewSession = reqNewSession;
 82      }
 83
 84      public void setConnection(Boolean b){
 85          mIsConnected=b;
 86      }
 87      public Boolean IsConnected(){
 88          return mIsConnected;
 89      }
 90
 91      public Context getContext(){return mContext;}
 92
 93      public String getURLPath(){return URLPATH;}
 94      public int getConnectTimeout(){return CONNECT_TIMEOUT;}
 95      public int getReadTimeout(){return READ_TIMEOUT;}
 96      public String getListMask(){return mMaskSerialized;}
 97      public void setListMask(String s){mMaskSerialized=s;}
 98
 99  }
100
```

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

```
 1 package com.fdx.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.net.HttpURLConnection;
19 import java.net.MalformedURLException;
20 import java.net.URL;
21 import java.net.URLEncoder;
22 import java.text.DecimalFormat;
23 import java.text.SimpleDateFormat;
24 import java.util.ArrayList;
25 import java.util.Date;
26 import java.util.HashMap;
27 import java.util.List;
28 import java.util.Map;
29 import java.util.UUID;
30
31 public class NetworkUtils {
32     private SessionInfo mSession;
33     private static final String TAG = "CB_NetworkUtils";
34     private static final String MYSQLDATEFORMAT="yyyy-MM-dd HH:mm:ss";
35
36     public NetworkUtils(Context c){
37         mSession=SessionInfo.get(c);
38     }
39
40     public String sendPostRequestJson(String requestURL,
41                                       HashMap<String, String>
   postDataParams) {
42
43         URL url;
44         try {
45             url = new URL(requestURL);
46             HttpURLConnection conn = (HttpURLConnection) url.
   openConnection();
47             conn.setReadTimeout(mSession.getReadTimeout());
48             conn.setConnectTimeout(mSession.getConnectTimeout());
49             conn.setRequestMethod("POST");
50             conn.setDoInput(true);
51             conn.setDoOutput(true);
52             OutputStream os = conn.getOutputStream();
```

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

```
101             return null;
102         }
103         return cs;
104     }
105
106     public Comment parseObjectComment(JSONObject obj){
107         try {
108         UUID uuid=UUID.fromString(obj.getString("id_user"));
109         User u=new User(obj.getString("family"), obj.getString("name"
    ) );
110         u.setId(uuid);
111         String s=obj.getString("date_comment");
112         Date date=new SimpleDateFormat(MYSQLDATEFORMAT).parse(s);
113         return new Comment(obj.getString("comment"),u,date);}
114         catch (Exception e){
115             //fdx Log.d(TAG, "Failure in parsing JSONObject Comments
     "+e);
116             return null;
117         }
118     }
119     public List<Note> parseNotesOfRecipe(String json){
120         Note n;
121         List<Note> ns=new ArrayList<>();
122         if ((json==null)||(json.equals(""))) return null;
123         try {
124             JSONArray jarr1=new JSONArray(json);
125             for (int i=0; i<jarr1.length(); i++){
126                 JSONObject obj = jarr1.getJSONObject(i);
127                 n=parseObjectNote(obj);
128                 if (n!=null) ns.add(n);
129             }
130         } catch (Exception e){
131             //fdx Log.d(TAG, "Failure in parsing JSON Array Comments
     "+e);
132             return null;
133         }
134         return ns;
135     }
136
137     public Note parseObjectNote(JSONObject obj){
138         try {
139             // uuid=UUID.fromString(obj.getString("id_recipe"));
140             UUID uuid=UUID.fromString(obj.getString("id_user"));
141             User u=new User(obj.getString("family"), obj.getString("
    name") );
142             u.setId(uuid);
143             String s=obj.getString("date_note");
144             Date date=new SimpleDateFormat(MYSQLDATEFORMAT).parse(s);
145             return new Note(obj.getInt("note"),u,date);}
146         catch (Exception e){
147             //fdx Log.d(TAG, "Failure in parsing JSONObject Note : "+
    e);
148             return null;
149         }
```

```
150     }
151     public Recipe parseObjectRecipeStamp(JSONObject obj){
152         try {
153             //---------------- uuid and users
154             UUID uuid = UUID.fromString(obj.getString("id_recipe"));
155             Recipe r = new Recipe(uuid);
156             uuid = UUID.fromString(obj.getString("id_owner"));
157             User u = new User(uuid);
158             r.setOwner(u);
159             //dates
160             Date date;
161             String s1 = obj.getString("lastupdate_recipe");
162             if (s1==null) return null;
163             if ((!s1.equals("null"))&&(s1.length()>5)) {
164                 date = new SimpleDateFormat(MYSQLDATEFORMAT).parse(s1
    );
165                 r.setDate(date);
166             } else {return null;}
167             s1 = obj.getString("lastupdate_photo");
168             if ((!s1.equals("null"))&&(s1.length()>5)) {
169                 date = new SimpleDateFormat(MYSQLDATEFORMAT).parse(s1
    );
170                 r.setDatePhoto(date);
171             }
172             s1=obj.getString("message");
173             if ((s1==null)||(s1.equals("null"))) s1="";
174             r.setMessage(s1);
175             r.setStatus(StatusRecipe.valueOf(obj.getString("status"))
    );
176             s1 = obj.getString("id_from");
177             if ((!s1.equals("null"))&&(s1.length()>5)) {
178                 uuid = UUID.fromString(s1);
179                 u = new User(uuid);
180                 r.setOwner(u);
181             }
182             return r;
183         }
184         catch (Exception e){
185             //fdx Log.d(TAG, "Failure in parsing JSONObject Recette
    Stamp : "+e);
186             return null;
187         }
188     }
189
190     public List<Recipe> parseStampsTiers(String json){
191         Recipe r;
192         List<Recipe> rs=new ArrayList<>();
193         if ((json==null)||(json.equals(""))) return null;
194         try {
195             JSONArray jarr1=new JSONArray(json);
196             for (int i=0; i<jarr1.length(); i++){
197                 JSONObject obj = jarr1.getJSONObject(i);
198                 r=parseObjectRecipeStamp(obj);
199                 if (r!=null) rs.add(r);
```

```
200             }
201         } catch (Exception e){
202             //fdx Log.d(TAG, "Failure in parsing JSON Array Comments
     "+e);
203             return null;
204         }
205         return rs;
206     }
207
208     public boolean parseObjectRecipe(Recipe r, JSONObject obj,
    boolean withphoto, boolean full){
209         try {
210             UUID uuid;
211             String s1, s2;
212             User u;
213             if (full) {
214                 uuid = UUID.fromString(obj.getString("id_owner"));
215                 s1 = obj.getString("owner_family");
216                 s2 = obj.getString("owner_name");
217                 u = new User(s1, s2);
218                 u.setId(uuid);
219                 r.setOwner(u);
220                 s1=obj.getString("message");
221                 if ((s1==null)||(s1.equals("null"))) s1="";
222                 r.setMessage(s1);
223                 r.setStatus(StatusRecipe.valueOf(obj.getString("
    status")));
224                 s1 = obj.getString("id_from");
225                 if ((!s1.equals("null"))&&(s1.length()>5)&&(s1!=null)
    ) {
226                     uuid = UUID.fromString(s1);
227                     s1 = obj.getString("from_family");
228                     s2 = obj.getString("from_name");
229                     u = new User(s1, s2);
230                     u.setId(uuid);
231                     r.setUserFrom(u);
232                 }
233             }
234             //------------ dates
235             Date date;
236             s1 = obj.getString("lastupdate_recipe");
237             if ((!s1.equals("null"))&&(s1.length()>5)&&(s1!=null)) {
238                 date = new SimpleDateFormat(MYSQLDATEFORMAT).parse(s1
    );
239                 r.setDate(date);
240             } else {return false;}
241             s1 = obj.getString("lastupdate_photo");
242             if ((!s1.equals("null"))&&(s1.length()>5)&&(s1!=null)) {
243                 date = new SimpleDateFormat(MYSQLDATEFORMAT).parse(s1
    );
244                 r.setDatePhoto(date);
245             }
246             //----------------- titre, source x2, nbpers
247             s1 = obj.getString("title");
```

```
248             URL url;
249             if (s1==null) return false;
250             if (s1.equals("null")) return false;
251             r.setTitle(s1);
252             r.setSource(obj.getString("source"));
253             s1 = obj.getString("source_url");
254             if (!s1.equals("null")&&(s1.length()>5)&&(s1!=null)) {
255                 try {
256                     url = new URL(s1);
257                     r.setSource_url(url);
258                 } catch (MalformedURLException e) {
259                     //fdx Log.d(TAG, "getURL from cookbook download
    failed in parse recipe");
260                 }
261             }
262             int nbp=obj.getInt("nb_pers");
263             if (nbp==0) return false;
264             r.setNbPers(nbp);
265             //---------------- Etapes and ing
266             DecimalFormat formatter = new DecimalFormat("00");
267             for (int i = 0; i < r.getNbStepMax(); i++) {
268                 s1 = "etape" + formatter.format(i + 1);
269                 s2=obj.getString(s1);
270                 if (!s2.equals("null")) {
271                     r.setStep(i + 1, s2);}
272             }
273             for (int i = 0; i < r.getNbIngMax(); i++) {
274                 s1 = "ing" + formatter.format(i + 1);
275                 s2=obj.getString(s1);
276                 if (!s2.equals("null")) {
277                     r.setIngredient(i + 1, s2);}
278             }
279             //----------------- enum
280             r.setSeason(RecipeSeason.valueOf(obj.getString("season"))
    );
281             r.setDifficulty(RecipeDifficulty.valueOf(obj.getString("
    difficulty")));
282             r.setType(RecipeType.valueOf(obj.getString("type")));
283             // --------------- photo
284             CookBook cb = CookBook.get(mSession.getContext());
285             File f=cb.getPhotoFile(r);
286             if (withphoto) {
287                 s1 = obj.getString("photo");
288                 if ((!s1.equals("null"))&&(!s1.equals(""))&&(s1!=null
    )) {
289                     Bitmap bm=PictureUtils.getBitmapFromString(s1);
290                     if (f.exists()){
291                         //fdx Log.d(TAG, "file  " + f.toString()+"
    ecrasée dans parsing recipe !");
292                         f.delete();
293                     }
294                     try {
295                         if(!f.createNewFile()) {
296                             //fdx Log.d(TAG, "Error in creating file
```

```
296   "+f.toString());
297                           }
298                       } catch (IOException e) {
299                           //fdx Log.d(TAG, "Error in creating file "+f.
    toString()+":" +e);
300                       }
301                       try {
302                           FileOutputStream out = new FileOutputStream(f
    );
303                           bm.compress(Bitmap.CompressFormat.JPEG, 100,
    out);
304                           out.flush();
305                           out.close();
306                       } catch (Exception e) {
307                           //fdx Log.d(TAG, "Storing bitmap error  " + e
    );
308                       }
309
310                   }
311               }
312
313               return true;
314           }
315           catch (Exception e){
316               //fdx Log.d(TAG, "Failure in parsing JSONObject Recette
     : "+e);
317               return false;
318           }
319       }
320
321       public boolean saveBmpInRecipe(Bitmap bmp, Recipe r){
322           CookBook cb = CookBook.get(mSession.getContext());
323           File f=cb.getPhotoFile(r);
324           if (f.exists()) f.delete();
325           try {
326               if(!f.createNewFile()) {
327                   //fdx Log.d(TAG, "Error in creating file  "+f.
    toString());
328               }
329               FileOutputStream out = new FileOutputStream(f);
330               bmp.compress(Bitmap.CompressFormat.JPEG, 100, out);
331               out.flush();
332               out.close();
333               return true;
334           } catch (Exception e) {
335               //fdx Log.d(TAG, "Storing bitmap error  " + e);
336               return false;
337           }
338       }
339
340
341   }
342
```

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

```java
package com.fdx.cookbook;

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


import java.io.ByteArrayOutputStream;

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

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

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

```java
package com.fdx.cookbook;

public enum RecipeSeason {
    WINTER,SUMMER,ALLYEAR;

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

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

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

```

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

```
 1 package com.fdx.cookbook;
 2
 3 import android.app.Activity;
 4 import android.content.Context;
 5 import android.graphics.Bitmap;
 6 import android.os.AsyncTask;
 7 import android.support.v7.app.AppCompatActivity;
 8 import android.util.Log;
 9 import android.widget.Toast;
10
11 import org.json.JSONArray;
12 import org.json.JSONObject;
13
14 import java.io.File;
15 import java.io.InputStream;
16 import java.net.HttpURLConnection;
17 import java.net.URL;
18 import java.text.DateFormat;
19 import java.text.DecimalFormat;
20 import java.text.SimpleDateFormat;
21 import java.util.ArrayList;
22 import java.util.Calendar;
23 import java.util.Date;
24 import java.util.HashMap;
25 import java.util.List;
26 import java.util.UUID;
27
28 enum AsynCallFlag { NEWRECIPE, NEWPHOTO, NEWCOMMENT, NEWRATING,
   GLOBALSYNC, DELETERECIPE}
29
30 class AsyncCallClass extends AsyncTask<Void, Integer, Boolean> {
31     private static final String TAG = "CB_AsynCalls";
32     private static final String PHP204 = "return204.php";
33     private static final String PHPUPDATECREATE = "
   updateorcreaterecipe.php";
34     private static final String PHPUPUPLOADPHOTO = "
   uploadphotointorecipe.php";
35     private static final String PHPDELETERECIPE = "deleterecipe.php";
36     private static final String PHPGETCOMMENTSOFRECIPE = "
   getcommentsofrecipe.php";
37     private static final String PHPADDCOMMENTTORECIPE = "
   addcommentwithdate.php";
38     private static final String PHPADDNOTETORECIPE = "addnotewithdate.
   php";
39     private static final String PHPGETNOTESOFRECIPE = "
   getnotesofrecipe.php";
40     private static final String PHPGETSTAMPSTIERS = "
   getrecipestampstiers.php";
41     private static final String PHPGETRECIPEFROMCB = "getrecipefromcb.
   php";
42     private static final String PHPGETRECIPEFROMCBWITHPHOTO = "
   getrecipefromcbwithphoto.php";
43     private static final String PHPACCEPTRECIPE = "acceptrecipe.php";
44     private static final String MYSQLDATEFORMAT="yyyy-MM-dd HH:mm:ss";
```

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

```
 94              }
 95              if (syncNotesRecipe(r)) {
 96                  deBugShow( "Notes de " + r.getTitle()+" updaté");
 97                  isChanged=true;
 98                  r.updateTS(AsynCallFlag.NEWRATING,false);
 99                  mCookbook.updateRecipe(r);
100              }
101              if (r.IsMarkedDeleted()){
102                  if (deleteRecipeFromCB(r)) {
103                      deBugShow( "Recette effacée : " + r.getTitle());
104                      mCookbook.removeRecipe(r);
105                      mCookbook.deleteImage(r);
106                      isChanged=true;
107                  } else{
108                      deBugShow("Recette " + r.getTitle()+" non effacée
    ");
109                  }
110              }
111          }
112          return isChanged;
113      }
114
115      @Override
116      protected void onPreExecute() {
117          super.onPreExecute();
118      }
119
120      @Override
121      protected void onPostExecute(Boolean isChanged) {
122          super.onPostExecute(isChanged);
123          if (isChanged)
124          Toast.makeText(mContext, mContext.getString(R.string.P1_sync)
    , Toast.LENGTH_SHORT).show();
125      }
126
127      private void deBugShow(String s){
128          //fdx Log.d(TAG, s);
129      }
130
131      private Boolean test204() {
132          try {
133              URL url = new URL(mSession.getURLPath() + PHP204);
134              HttpURLConnection conn = (HttpURLConnection) url.
    openConnection();
135              conn.setConnectTimeout(mSession.getConnectTimeout());
136              conn.setReadTimeout(mSession.getReadTimeout());
137              conn.setRequestMethod("HEAD");
138              InputStream in = conn.getInputStream();
139              int status = conn.getResponseCode();
140              in.close();
141              conn.disconnect();
142              return (status == HttpURLConnection.HTTP_NO_CONTENT);
143          } catch (Exception e) {
144              deBugShow( "Test 204 : " + e);
```

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

```
195         s1 = dateFormat.format(r.getDatePhoto());
196         data.put("date", s1);
197         String result = mNetUtils.sendPostRequestJson(mSession.
    getURLPath() + PHPUPUPLOADPHOTO, data);
198         if (result==null) return false;
199         if (!result.trim().equals("1")) {
200             deBugShow( "Retour de "+ PHPUPUPLOADPHOTO+" = "+result);
201             return false;}
202         return true;
203     }
204     private Boolean deleteRecipeFromCB(Recipe r) {
205         HashMap<String, String> data = new HashMap<>();
206         data.put("idrecipe", r.getId().toString());
207         data.put("iduser", mSession.getUser().getId().toString());
208         String result = mNetUtils.sendPostRequestJson(mSession.
    getURLPath() + PHPDELETERECIPE, data);
209         if (result==null) return false;
210         if (!result.trim().equals("1")) {
211             deBugShow( "Retour de "+ PHPDELETERECIPE+" = "+result);
212             return false;}
213         return true;
214     }
215
216     private Boolean syncCommentsRecipe(Recipe r) {
217         boolean ret=false;
218         HashMap<String, String> data = new HashMap<>();
219         data.put("idrecipe", r.getId().toString());
220         String result = mNetUtils.sendPostRequestJson(mSession.
    getURLPath() + PHPGETCOMMENTSOFRECIPE, data);
221         List<Comment> downloadedComments=mNetUtils.
    parseCommentsOfRecipe(result);
222         if (downloadedComments==null) downloadedComments=new
    ArrayList<>();
223         List<Comment> recipeComments=r.getComments();
224         if (recipeComments==null) recipeComments=new ArrayList<>();
225         //----------- boucle local
226         List<Comment> cloc= new ArrayList<>();
227         Comment c,ci;
228         if (!recipeComments.isEmpty()){
229             for (int i = 0; i < recipeComments.size(); i++){
230                 ci=recipeComments.get(i);
231                 c=new Comment(ci.getTxt(),ci.getUser(),ci.getDate());
232                 if (downloadedComments.isEmpty()) {cloc.add(c);} else
     {
233                 if (downloadedComments.indexOf(c)==-1) cloc.add(c);}
234             }
235         }
236         //----------- boucle serveur
237         List<Comment> cser= new ArrayList<>();
238         if (!downloadedComments.isEmpty()){
239             for (int i = 0; i < downloadedComments.size(); i++){
240                 ci=downloadedComments.get(i);
241                 c=new Comment(ci.getTxt(),ci.getUser(),ci.getDate());
242                 if (recipeComments.isEmpty()) {cser.add(c);} else {
```

```
243                 if (recipeComments.indexOf(c)==-1) cser.add(c);}
244             }
245         }
246         if (!cser.isEmpty()){
247             for (int i = 0; i < cser.size(); i++){
248                 r.addComment(cser.get(i));
249             }
250             if (cser.size()>0) {
251                 mCookbook.updateRecipe(r);
252                 ret=true;
253             }
254         }
255         if (!cloc.isEmpty()) {
256             if (!uploadComments(r,cloc)) {ret=false;}
257             else {ret=true;}
258         }
259         return ret;
260     }
261
262     private Boolean uploadComments(Recipe r, List<Comment> cs){
263         if (cs==null) return false;
264         if (cs.size()==0) return true;
265         HashMap<String, String> data = new HashMap<>();
266         DateFormat dateFormat = new SimpleDateFormat(MYSQLDATEFORMAT)
    ;
267         String s1;
268         boolean b=true;
269         for (Comment c:cs) {
270             data.clear();
271             data.put("idrecipe", r.getId().toString());
272             data.put("idfrom", c.getUser().getId().toString());
273             data.put("comment", c.getTxt());
274             s1 = dateFormat.format(c.getDate());
275             data.put("date",s1  );
276             String result = mNetUtils.sendPostRequestJson(mSession.
    getURLPath() + PHPADDCOMMENTTORECIPE, data);
277             if (result==null) b=false;
278             if (!result.equals("1")) {
279                 deBugShow( "Retour de "+ PHPADDCOMMENTTORECIPE+" = "+
    result);
280                 b=false;}
281         }
282         return b;
283     }
284     private Boolean syncNotesRecipe(Recipe r) {
285         boolean ret=false;
286         HashMap<String, String> data = new HashMap<>();
287         data.put("idrecipe", r.getId().toString());
288         String result = mNetUtils.sendPostRequestJson(mSession.
    getURLPath() + PHPGETNOTESOFRECIPE, data);
289         List<Note> downloadedNotes=mNetUtils.parseNotesOfRecipe(
    result);
290         if (downloadedNotes==null) downloadedNotes=new ArrayList<>();
291         List<Note> recipeNotes=r.getNotes();
```

```
292         if (recipeNotes==null) recipeNotes=new ArrayList<>();
293         //----------- boucle local
294         List<Note> cloc= new ArrayList<>();
295         Note c,ci;
296         if (!recipeNotes.isEmpty()){
297             for (int i = 0; i < recipeNotes.size(); i++){
298                 ci=recipeNotes.get(i);
299                 c=new Note(ci.getNote(),ci.getUser(),ci.getDate());
300                 if (downloadedNotes.isEmpty()) {
301                     cloc.add(c);}
302                 else {
303                     if (downloadedNotes.indexOf(c)==-1) cloc.add(c);}
304             }
305         }
306         //----------- boucle serveur
307         List<Note> cser= new ArrayList<>();
308         if (!downloadedNotes.isEmpty()){
309             for (int i = 0; i < downloadedNotes.size(); i++){
310                 ci=downloadedNotes.get(i);
311                 c=new Note(ci.getNote(),ci.getUser(),ci.getDate());
312                 if (recipeNotes.isEmpty()){
313                     cser.add(c);}
314                 else{
315                     if (recipeNotes.indexOf(c)==-1) cser.add(c);}
316             }
317         }
318         if (!cser.isEmpty()){
319             for (int i = 0; i < cser.size(); i++){
320                 r.addNote(cser.get(i));
321             }
322             if (cser.size()>0) {
323                 mCookbook.updateRecipe(r);
324                 ret=true;
325             }
326         }
327         if (!cloc.isEmpty()) {
328             if (!uploadNotes(r,cloc)) {ret=false;}
329             else {ret=true;}
330         }
331         return ret;
332     }
333     private Boolean uploadNotes(Recipe r, List<Note> cs){
334         if (cs==null) return false;
335         if (cs.size()==0) return true;
336         HashMap<String, String> data = new HashMap<>();
337         DateFormat dateFormat = new SimpleDateFormat(MYSQLDATEFORMAT)
    ;
338         DecimalFormat formatter = new DecimalFormat("00");
339         String s1;
340         boolean b=true;
341         for (Note c:cs) {
342             data.clear();
343             data.put("idrecipe", r.getId().toString());
344             data.put("idfrom", c.getUser().getId().toString());
```

```
345             data.put("note", formatter.format(c.getNote()));
346             s1 = dateFormat.format(c.getDate());
347             data.put("date",s1  );
348             String result = mNetUtils.sendPostRequestJson(mSession.
    getURLPath() + PHPADDNOTETORECIPE, data);
349             if (result==null) b=false;
350             if (!result.equals("1")) {
351                 deBugShow( "Retour de "+ PHPADDNOTETORECIPE+" = "+
    result);
352                 b=false;}
353         }
354         return b;
355     }
356     private Boolean syncTiersRecipe(User user) {
357         boolean ret=false;
358         HashMap<String, String> data = new HashMap<>();
359         data.put("iduser", user.getId().toString());
360         String result = mNetUtils.sendPostRequestJson(mSession.
    getURLPath() + PHPGETSTAMPSTIERS, data);
361         if (result==null)return ret;
362         List<Recipe> tiersRecipe=mNetUtils.parseStampsTiers(result);
363         if (tiersRecipe==null) return ret;
364         boolean withphoto=false,update=false;
365         Recipe rloc, rnew;
366         for(Recipe r:tiersRecipe) {
367             rloc=mCookbook.getRecipe(r.getId());
368             if (rloc==null) { // case recipe tiers not found locally
     => download
369                 rnew=downloadRecipe(r);
370                 if (rnew==null) {
371                     deBugShow( "Erreur ds download de recette "+ rnew
    .getTitle());
372                     ret=false;
373                 } else {
374                     mCookbook.addRecipe(rnew);
375                     deBugShow( "Recette "+ rnew.getTitle()+"
    downloadée");
376                     ret=true;
377                 }
378             } else {
379                 if (rloc.IsVisible()){ // case recipe tiers active
    and to be updated if necessary
380                     withphoto=IsAfterAndNonNull(r.getDatePhoto(),
    rloc.getDatePhoto());
381                     update=IsAfterAndNonNull(r.getDate(), rloc.
    getDate());
382                     if (r.IsMessage()){// case recipe tiers active
    and status needs update
383                         if (recipeAccepted(r)){
384                             deBugShow("Recette "+ rloc.getTitle()+"
    made visible on server");
385                         }
386                     }
387                     if(update || withphoto){
```

```
388                         if (updateRecipe(rloc, withphoto)){
389                             mCookbook.updateRecipe(rloc);
390                             deBugShow( "Recette "+ rloc.getTitle()+"
    updatée (with photo :"+withphoto+")");
391                             ret=true;
392                         }
393                     }
394                     else continue;
395                 }
396             }
397         }
398         return ret;
399     }
400     private boolean IsAfterAndNonNull(Date ds, Date dl){
401         Calendar c=Calendar.getInstance();
402         c.set(2000,0,1,0,0, 0);
403         Date d0=c.getTime();
404         boolean morerecent=(ds.compareTo(dl)==1);
405         boolean isnotnull=(ds.compareTo(d0)==1);
406         return morerecent && isnotnull;
407     }
408
409     private Recipe downloadRecipe(Recipe ref){
410         if (ref==null) return null;
411         HashMap<String,String> data = new HashMap<>();
412         data.put("idrecipe", ref.getId().toString().trim());
413         data.put("iduser", mSession.getUser().getId().toString().trim
    ());
414         String result = mNetUtils.sendPostRequestJson(mSession.
    getURLPath()+PHPGETRECIPEFROMCBWITHPHOTO,data);
415         Recipe r=new Recipe();
416         try {
417             JSONArray jarr1 = new JSONArray(result);
418             if (jarr1.length()==0){
419                 deBugShow( " No json objects from download <"+ref.
    getId()+">");
420                 return null;
421             } else {
422                 if (jarr1.length()>1) deBugShow( " Recette non unique
     dans cookbook : <"+ref.getId()+">");
423                 JSONObject obj = jarr1.getJSONObject(0);
424                 r = new Recipe(UUID.fromString(obj.getString("
    id_recipe")));
425                 if (!mNetUtils.parseObjectRecipe(r,obj, true, true))
    return null;
426             }
427         } catch (Exception e) {
428             deBugShow( " Error parsing CB in downloadRecipe" + e);
429         }
430         return r;
431     }
432     private Boolean updateRecipe(Recipe ref, boolean withphoto){
433         if (ref==null) return null;
434         HashMap<String,String> data = new HashMap<>();
```

```
435         data.put("idrecipe", ref.getId().toString().trim());
436         data.put("withphoto", (withphoto)?"1":"0");
437         String result = mNetUtils.sendPostRequestJson(mSession.
    getURLPath()+ PHPGETRECIPEFROMCB,data);
438         Recipe r=new Recipe();
439         try {
440             JSONArray jarr1 = new JSONArray(result);
441             if (jarr1.length()!=1){
442                 deBugShow( " Number of json objects from
    downloadRecipes =" + jarr1.length()+" !");
443                 return null;
444             } else {
445                 JSONObject obj = jarr1.getJSONObject(0);
446                 if (!mNetUtils.parseObjectRecipe(ref,obj, withphoto,
    false)) {
447                     deBugShow( " Null recipe from JSOn object" +
    result);
448                     return false;
449                 }
450             }
451         } catch (Exception e) {
452             deBugShow( " Error parsing CB in downloadRecipe" + e);
453             return false;
454         }
455         return true;
456     }
457
458     private Boolean recipeAccepted(Recipe r){
459         if (r==null) return null;
460         HashMap<String,String> data = new HashMap<>();
461         data.put("idrecipe", r.getId().toString().trim());
462         data.put("iduser", mSession.getUser().getId().toString());
463         String result = mNetUtils.sendPostRequestJson(mSession.
    getURLPath()+ PHPACCEPTRECIPE,data);
464         if (!result.equals("1")) {
465             deBugShow( "Retour de "+ PHPACCEPTRECIPE+" = "+result);
466             return false;}
467         return true;
468     }
469 }
470
```

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

```java
package com.fdx.cookbook;

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

import java.util.UUID;

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

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

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

}

```

```java
package com.fdx.cookbook;

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

```

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

```java
package com.fdx.cookbook;

import android.content.Intent;
import android.graphics.drawable.AnimationDrawable;
import android.os.AsyncTask;
import android.os.Bundle;
import android.support.v4.content.ContextCompat;
import android.support.v7.app.AppCompatActivity;
import android.text.Editable;
import android.text.TextWatcher;
import android.text.method.HideReturnsTransformationMethod;
import android.text.method.PasswordTransformationMethod;
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 android.widget.Toast;

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

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

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

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

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

```
145         mEnterFamily.addTextChangedListener(new TextWatcher() {
146             @Override
147             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) {}
148 
149             @Override
150             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
151                 String z=s.toString();
152                 mFamilyEntered=z;
153                 Integer d= getColorOnCondition(z,NEW_FAMILY);
154                 mEnterFamily.setBackgroundColor(ContextCompat.
    getColor(getApplicationContext(), d));
155                 buttonEnabled((d==R.color.bg_enter_val));
156             }
157 
158             @Override
159             public void afterTextChanged(Editable s) {}
160         });
161         mEnterMember.addTextChangedListener(new TextWatcher() {
162             @Override
163             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) {}
164 
165             @Override
166             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
167                 String z=s.toString();
168                 mMemberEntered=z;
169                 Integer d= getColorOnCondition(z,NEW_MEMBER);
170                 mEnterMember.setBackgroundColor(ContextCompat.
    getColor(getApplicationContext(), d));
171                 buttonEnabled((d==R.color.bg_enter_val));
172             }
173 
174             @Override
175             public void afterTextChanged(Editable s) {}
176         });
177         mEnterPwd.addTextChangedListener(new TextWatcher() {
178             @Override
179             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) {}
180 
181             @Override
182             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
183                 String z=s.toString();
184                 mPwdEntered=z;
185                 Integer d= getColorOnCondition(z,NEW_PWD);
186                 mEnterPwd.setBackgroundColor(ContextCompat.getColor(
    getApplicationContext(), d));
187                 buttonEnabled((d==R.color.bg_enter_val));
188             }
189 
```

```
190             @Override
191             public void afterTextChanged(Editable s) {}
192         });
193         mNewSession.setOnClickListener(new View.OnClickListener() {
194             @Override
195             public void onClick(View v) {
196                 if(testConnectStatus()){
197                     mState=NEW_SESSION;
198                     updateDisplayOnAsync(true);
199                     goAsyncTasks();
200                 } else {
201                 tc.testGo();}
202             }
203         });
204         mNewMember.setOnClickListener(new View.OnClickListener() {
205             @Override
206             public void onClick(View v) {
207                 if(testConnectStatus()){
208                     mState=NEW_MEMBER;
209                     updateDisplayOnAsync(true);
210                     goAsyncTasks();
211                 } else {
212                 tc.testGo();}
213             }
214         });
215         mNewFamily.setOnClickListener(new View.OnClickListener() {
216             @Override
217             public void onClick(View v) {
218                 if(testConnectStatus()){
219                     mState=NEW_FAMILY;
220                     updateDisplayOnAsync(true);
221                     goAsyncTasks();
222                 } else {
223                 tc.testGo(); }
224             }
225         });
226     }
227
228     public void ShowHidePass(View view){
229         if(view.getId()==R.id.show_pass_btn){
230             if(mEnterPwd.getTransformationMethod().equals(
    PasswordTransformationMethod.getInstance())){
231                 ((ImageView)(view)).setImageResource(R.drawable.
    ic_pwd_no_vis);
232                 mEnterPwd.setTransformationMethod(
    HideReturnsTransformationMethod.getInstance());
233             }
234             else{
235                 ((ImageView)(view)).setImageResource(R.drawable.
    ic_pwd_vis);
236                 mEnterPwd.setTransformationMethod(
    PasswordTransformationMethod.getInstance());
237
238             }
```

```
239         }
240     }
241     /*private void updateTop(){
242         mEnterFamilyLbl.setText(R.string.P0LF);
243         mEnterFamily.setText(mFamilyEntered);
244         mEnterMemberLbl.setText(R.string.P0LM);
245         mEnterMember.setText(mMemberEntered);
246         mEnterPwdLbl.setText(R.string.P0LP);
247         mEnterPwd.setText(mPwdEntered);
248         mEnterMessage.setText("");
249         mNewSession.setText(R.string.P0BP);
250         mNewMember.setText(R.string.P0BM);
251         mNewFamily.setText(R.string.P0BF);
252     }*/
253
254     private void updateDisplayOnAsync(Boolean b){
255         int indNet,indClick;
256         if (b){
257             indNet=View.VISIBLE;
258             indClick=View.INVISIBLE;
259             frameNetAnim.start();
260             mProgressBar.setProgress(1);
261         }
262         else {
263             indNet=View.INVISIBLE;
264             indClick=View.VISIBLE;
265             frameNetAnim.stop();
266             mProgressBar.setProgress(0);
267         }
268         mNetAnim.setVisibility(indNet);
269         mProgressBar.setVisibility(indNet);
270         mNewFamily.setVisibility(indClick);
271         mNewMember.setVisibility(indClick);
272         mNewSession.setVisibility(indClick);
273     }
274
275     private Boolean IsLenOK(String s, int min, int max){
276         int l=s.length();
277         return ((l>min)&&(l<max));
278     }
279
280     private Integer getColorOnCondition(String z, Integer state){
281         state=state-1;
282         int retDraw =0;
283         int TRUE=R.color.bg_enter_val;
284         int FALSE=R.color.light_red;
285         retDraw=((Pattern.matches(REGEX[state],z)&&(IsLenOK(z,MINMAX[
    state][0],MINMAX[state][1])))?
286                 TRUE : FALSE);
287         Integer err_mess[]={R.string.P0ERUF,R.string.P0RUM,R.string.
    P0RUP};
288         if (retDraw==FALSE){
289         mEnterMessage.setText(getResources().getString(err_mess[state
    ],MINMAX[state][0],MINMAX[state][1]));}
```

```
290         else {mEnterMessage.setText("");}
291         return retDraw;
292     }
293 
294     private Boolean testConnectStatus(){
295         mEnterFamily.setBackgroundColor(ContextCompat.getColor(
    getApplicationContext(), R.color.bg_enter_val));
296         mEnterMember.setBackgroundColor(ContextCompat.getColor(
    getApplicationContext(), R.color.bg_enter_val));
297         mEnterPwd.setBackgroundColor(ContextCompat.getColor(
    getApplicationContext(), R.color.bg_enter_val));
298         Boolean ret=true;
299         //String message="";
300         if(!mSession.IsConnected()){
301             ret=false;
302             Toast.makeText(getApplicationContext(), getString(R.
    string.P0ER_nocon), Toast.LENGTH_LONG).show();
303             //message += getResources().getString(R.string.P0ER_nocon
    )+"\n";
304         }
305         //mEnterMessage.setText(message);
306         return ret;
307     }
308 
309     private void buttonEnabled(Boolean b){
310         mNewSession.setEnabled(b);
311         mNewMember.setEnabled(b);
312         mNewFamily.setEnabled(b);
313     }
314 
315     /**
    ************************************************************************
    ******************
316     *                                ASYNC                                *
317 
    ************************************************************************
    ********************/
318 
319     private void goAsyncTasks() {
320 
321         class GoAsyncTasks extends AsyncTask<Void, Void, Boolean> {
322 
323             @Override
324             protected void onPreExecute() {
325                 super.onPreExecute();
326             }
327 
328             @Override
329             protected void onPostExecute(Boolean b) {
330                 super.onPostExecute(b);
331                 updateDisplayOnAsync(false);
332                 if (b) {
333                     mSession.setStoredUser(mUser);
```

```
334                     Intent intent=new Intent(getApplicationContext(),
     RecipeListActivity.class);
335                     startActivity(intent);
336                     finish();
337                 }
338                 return;
339             }
340 
341             @Override
342             protected Boolean doInBackground(Void... voids) {
343                 CookBook cb=CookBook.get(getApplicationContext());
344                 String s=getFamilyComp();
345                 mProgressBar.setProgress(2);
346                 if (s==null){
347                     mEnterMessage.setText(R.string.P0ER_com);
348                     return false; } else {
349                     memberFamily=parseJsonFamily(s); }
350                 mProgressBar.setProgress(5);
351                 Integer cas=goFamilyGet();
352                 if (cas==0) { return false;}
353                 if ((cas==NEW_FAMILY)||(cas==NEW_MEMBER)){
354                     if (createMember(cas)){
355                         mProgressBar.setProgress(10);
356                         mEnterMessage.setText(getString(R.string.
    P0OKM));
357                     } else {
358                         mEnterMessage.setText(getString(R.string.
    P0ERM_new));
359                         return false;}
360                 }
361                 cb.clearCookBook();
362                 s=downloadCB();
363                 if (s==null){
364                     //fdx Log(TAG, "download cookbook failed");
365                     return false;
366                 }
367                 mProgressBar.setProgress(50);
368                 mRecipes=parseJsonCB(s);
369                 if (!downloadNotesAndParse()){
370                     //fdx Log(TAG, "Failure in reading and parsing
    Notes");
371                     return false;
372                 }
373                 mProgressBar.setProgress(70);
374                 if (!downloadCommentsAndParse()){
375                     //fdx Log(TAG, "Failure in reading and parsing
    Comments");
376                     return false;
377                 }
378                 mProgressBar.setProgress(70);
379                 cb.fillCookBook(mRecipes);
380                 mProgressBar.setProgress(90);
381                 return true;
382             }
```

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

```
434         String json = mNetUtils.sendPostRequestJson(mSession.
    getURLPath()+PHPREQGETNOTES,data);
435         //User u;
436         //String s;
437         UUID uuid;
438         //Date date;
439         Note n;
440         try {
441             JSONArray jarr1=new JSONArray(json);
442             for (int i=0; i<jarr1.length(); i++){
443                 JSONObject obj = jarr1.getJSONObject(i);
444                 uuid=UUID.fromString(obj.getString("id_recipe"));
445                 n=mNetUtils.parseObjectNote(obj);
446                 for(Recipe r:mRecipes){
447                     if (r.getId().equals(uuid)){
448                         r.addNote(n);
449                         break;
450                     }
451                 }
452             }
453         } catch (Exception e){
454                 //fdx Log(TAG, "Failure in parsing JSON Notes "+e);
455                 return false;
456             }
457         return true;
458     }
459
460     private Boolean downloadCommentsAndParse(){
461         HashMap<String,String> data = new HashMap<>();
462         data.put("iduser", mUser.getId().toString().trim());
463         String json = mNetUtils.sendPostRequestJson(mSession.
    getURLPath()+PHPREQGETCOMMENTS,data);
464         UUID uuid;
465         Comment c;
466         try {
467             JSONArray jarr1=new JSONArray(json);
468             for (int i=0; i<jarr1.length(); i++){
469                 JSONObject obj = jarr1.getJSONObject(i);
470                 c=mNetUtils.parseObjectComment(obj);
471                 uuid=UUID.fromString(obj.getString("id_recipe"));
472                 if (c!=null) {
473                     for(Recipe r:mRecipes){
474                         if (r.getId().equals(uuid)){
475                             r.addComment(c);
476                             break;
477                         }
478                     }
479                 }
480             }
481         } catch (Exception e){
482             //fdx Log(TAG, "Failure in parsing JSON Array Comments "+
    e);
483             return false;
484         }
```

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

```
536                     mEnterFamily.setBackgroundColor(ContextCompat.
    getColor(getApplicationContext(), R.color.light_red)); }
537                 else {
538                     //mEnterMessage.setText(R.string.P0OKF);
539                     User u= new User("","");
540                     for(User user:memberFamily){
541                         if (user.getName().equals(mMemberEntered)){u=
    user;}
542                     }
543                     if (u.getName().equals("")) {
544                         mEnterMessage.setText(R.string.P0ERF_nomem);
545                         mEnterMember.setBackgroundColor(ContextCompat
    .getColor(getApplicationContext(), R.color.light_red));
546                     } else {
547                         if (!mPwdRead.equals(mPwdEntered.trim())){
548                             mEnterMessage.setText(R.string.
    P0ERP_wrong);
549                             mEnterPwd.setBackgroundColor(
    ContextCompat.getColor(getApplicationContext(), R.color.light_red));
550                         } else {
551                             mUser=u;
552                         mEnterMessage.setText(getString(R.string.
    P0OKP, u.getName()));
553                         return NEW_SESSION; }
554                     }
555                 }
556                 break;
557             }
558             case NEW_FAMILY : {
559                 if (memberFamily.size()==0) {
560                     mEnterMessage.setText(R.string.P0OK_done);
561                     return NEW_FAMILY;
562                 }
563                 else {
564                     mEnterMessage.setText(R.string.P0ERF_notnew);
565                     mEnterFamily.setBackgroundColor(ContextCompat.
    getColor(getApplicationContext(), R.color.light_red));
566                 }
567                 break;
568             }
569             case NEW_MEMBER : {
570                 if (memberFamily.size()==0) {
571                     mEnterMessage.setText(R.string.P0ERM_nofam);
572                     mEnterFamily.setBackgroundColor(ContextCompat.
    getColor(getApplicationContext(), R.color.light_red));
573                 }
574                 else {
575                     // family found
576                     User u= new User("","");
577                     for(User user:memberFamily){
578                         if (user.getName().equals(mMemberEntered)){u=
    user;}
579                     }
580                     if (u.getName().equals("")) {
```

```
581                             if (!mPwdRead.equals(mPwdEntered.trim())){
582                                 mEnterMessage.setText(R.string.
    P0ERP_wrong);
583                                 mEnterPwd.setBackgroundColor(
    ContextCompat.getColor(getApplicationContext(), R.color.light_red));
584                             } else {
585                             mEnterMessage.setText(R.string.P0OKM);
586                             return NEW_MEMBER;}
587                         } else {
588                             mEnterMessage.setText(R.string.P0ERM_notnew);
589                             mEnterMember.setBackgroundColor(ContextCompat
    .getColor(getApplicationContext(), R.color.light_red));
590                         }
591                     }
592                     break;
593                 }
594                 default : {
595                     //fdx Log(TAG, "Switch case on mState anomaly : "+
    mState);
596                 }
597             }
598             return 0;
599         }
600 }
601
```

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

```java
package com.fdx.cookbook;

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

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

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

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

    public void testGo(){

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

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

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

        testCon test = new testCon();
```

```
53          test.execute();
54      }
55  }
56
```

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

```java
package com.fdx.cookbook;

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

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

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

```

```
53      }
54
55      @Override
56      public void onUpgrade(SQLiteDatabase db, int oldVersion, int
    newVersion) {
57
58      }
59  }
60
```

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

```java
package com.fdx.cookbook;

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

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

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

```

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

```java
package com.fdx.cookbook;

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

import java.util.UUID;

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

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

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

```java
package com.fdx.cookbook;

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

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

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

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

```
104     public void onPause(){
105         super.onPause();
106     }
107 
108     @Override
109     public void onCreateOptionsMenu(Menu menu, MenuInflater inflater)
     {
110         super.onCreateOptionsMenu(menu, inflater);
111         inflater.inflate(R.menu.fragment_recipe, menu);
112     }
113 
114     @Override
115     public boolean onOptionsItemSelected(MenuItem item) {
116         switch (item.getItemId()) {
117             case R.id.recipe_menu_done:
118                 if (mRecipe.getTitle().length()>0){
119                     if (!mRecipe.hasNotChangedSince(mRecipeInit)){
120                     mRecipe.updateTS(AsynCallFlag.NEWRECIPE, true);
121                     mRecipe.setDate(new Date());
122                     }
123                     if (mBmp!=null){
124                         NetworkUtils networkutils=new NetworkUtils(
    getContext());
125                         networkutils.saveBmpInRecipe(mBmp, mRecipe);
126                         ResizePhoto(mRecipe);
127                         mRecipe.updateTS(AsynCallFlag.NEWPHOTO,true);
128                     }
129                     if (IsRecipeNew(mRecipeId)){
130                         CookBook.get(getActivity()).addRecipe(mRecipe
    );
131                     } else {
132                         CookBook.get(getActivity()).updateRecipe(
    mRecipe);
133                     }
134 
135                 }
136                 getActivity().onBackPressed();
137                 return true;
138             case R.id.recipe_menu_delete:
139                 if (!IsRecipeNew(mRecipeId)){
140                     CookBook.get(getActivity()).markRecipeToDelete(
    mRecipe);
141                 }
142                 getActivity().onBackPressed();
143                 return true;
144             default:
145                 return super.onOptionsItemSelected(item);
146         }
147     }
148 
149     @Override
150     public View onCreateView(LayoutInflater inflater, ViewGroup
    container, Bundle savedInstanceState){
151         final View v=inflater.inflate(R.layout.fragment_recipe_edit,
```

```
151 container, false);
152         mScroll=(ScrollView) v.findViewById(R.id.
    fragment_recipe_scroll);
153         /*PackageManager packageManager=getActivity().
    getPackageManager();
154             final Intent captureImage=new Intent(MediaStore.
    ACTION_IMAGE_CAPTURE);
155             boolean canTakePhoto=(mPhotoFile != null) &&
156                     (captureImage.resolveActivity(packageManager)!=
    null);*/
157         mPhotoView=(ImageView) v.findViewById(R.id.recipe_photo);
158         mPhotoView.setOnClickListener(new View.OnClickListener() {
159             @Override
160             public void onClick(View v) {
161                 Intent intent = new Intent();
162                 intent.setType("image/*");
163                 intent.setAction(Intent.ACTION_GET_CONTENT);
164                 startActivityForResult(Intent.createChooser(intent, "
    Select Picture"), PICK_IMAGE);
165             }
166         });
167         updatePhotoView();
168         mTitleField= (EditText) v.findViewById(R.id.recipe_title);
169         mTitleField.setText(mRecipe.getTitle());
170         mTitleField.addTextChangedListener(new TextWatcher() {
171             @Override
172             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) {
173             }
174
175             @Override
176             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
177                 mRecipe.setTitle(s.toString());
178             }
179
180             @Override
181             public void afterTextChanged(Editable s) {
182             }
183         });
184
185         mSourceField= (EditText) v.findViewById(R.id.recipe_source);
186         mSourceField.setText(mRecipe.getSource());
187         mSourceField.addTextChangedListener(new TextWatcher() {
188             @Override
189             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) {
190             }
191
192             @Override
193             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
194                 mRecipe.setSource(s.toString());
195             }
```

```
196
197             @Override
198             public void afterTextChanged(Editable s) {
199             }
200         });
201
202         mSourceUrl=(EditText) v.findViewById(R.id.recipe_source_url);
203         mSourceUrl.setText(mRecipe.getSource_url().toString());
204         mSourceUrl.addTextChangedListener(new TextWatcher() {
205             @Override
206             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) {
207
208             }
209
210             @Override
211             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
212                 if (s.toString().equals("")) {} else {
213                     try {
214                         URL url=new URL(s.toString());
215                         mRecipe.setSource_url(url);
216                     } catch (MalformedURLException e) {
217                         //fdx Log(TAG, "onTextChanged de mSource_url
     >" +s.toString()+"< Failed >"+ e);
218                     }
219                 }
220             }
221
222             @Override
223             public void afterTextChanged(Editable s) {
224             }
225         });
226
227         mNbPersField= (EditText) v.findViewById(R.id.recipe_nbpers);
228         mNbPersField.setText(mRecipe.getNbPers()+"");
229         mNbPersField.addTextChangedListener(new TextWatcher() {
230             @Override
231             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) {
232
233             }
234
235             @Override
236             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
237                 if (s.toString().equals("")) {} else {
238                     int nb_entered = Integer.parseInt(s.toString());
239                     if ((nb_entered > 0) && (nb_entered < 13)) {
240                         mRecipe.setNbPers(nb_entered);
241                     }
242                 }
243             }
244
```

```
245             @Override
246             public void afterTextChanged(Editable s) {
247
248             }
249         });
250
251         mSeasonSpinner= (Spinner) v.findViewById(R.id.recipe_season);
252         ArrayAdapter<CharSequence> adapterSeason = ArrayAdapter.
    createFromResource(getContext(),
253                 R.array.recipe_season_array, R.layout.
    edit_spinner_item);
254         mSeasonSpinner.setAdapter(adapterSeason);
255         mSeasonSpinner.setSelection(RecipeSeason.getIndex(mRecipe.
    getSeason()));
256         mSeasonSpinner.setOnItemSelectedListener(new AdapterView.
    OnItemSelectedListener() {
257             @Override
258             public void onItemSelected(AdapterView<?> parent, View
    view, int position, long id) {
259                 mRecipe.setSeason(RecipeSeason.getSeason(position));
260             }
261
262             @Override
263             public void onNothingSelected(AdapterView<?> parent) {
264
265             }
266         });
267
268         mTypeSpinner= (Spinner) v.findViewById(R.id.recipe_type);
269         ArrayAdapter<CharSequence> adapterType = ArrayAdapter.
    createFromResource(getContext(),
270                 R.array.recipe_type_array, R.layout.edit_spinner_item
    );
271         mTypeSpinner.setAdapter(adapterType);
272         mTypeSpinner.setSelection(RecipeType.getIndex(mRecipe.getType
    ()));
273         mTypeSpinner.setOnItemSelectedListener(new AdapterView.
    OnItemSelectedListener() {
274             @Override
275             public void onItemSelected(AdapterView<?> parent, View
    view, int position, long id) {
276                 mRecipe.setType(RecipeType.getType(position));
277             }
278
279             @Override
280             public void onNothingSelected(AdapterView<?> parent) {
281
282             }
283         });
284
285         mDifficultySpinner= (Spinner) v.findViewById(R.id.
    recipe_difficulty);
286         ArrayAdapter<CharSequence> adapterDifficulty = ArrayAdapter.
    createFromResource(getContext(),
```

```
287                 R.array.recipe_difficulty_array, R.layout.
    edit_spinner_item);
288         mDifficultySpinner.setAdapter(adapterDifficulty);
289         mDifficultySpinner.setSelection(RecipeDifficulty.getIndex(
    mRecipe.getDifficulty()));
290         mDifficultySpinner.setOnItemSelectedListener(new AdapterView.
    OnItemSelectedListener() {
291             @Override
292             public void onItemSelected(AdapterView<?> parent, View
    view, int position, long id) {
293                 mRecipe.setDifficulty(RecipeDifficulty.getDifficulty(
    position));
294             }
295
296             @Override
297             public void onNothingSelected(AdapterView<?> parent) {
298
299             }
300         });
301         final int[] rID= {R.id.recipe_S1,R.id.recipe_S2,R.id.
    recipe_S3,R.id.recipe_S4,
302                 R.id.recipe_S5,R.id.recipe_S6,R.id.recipe_S7,R.id.
    recipe_S8,R.id.recipe_S9};
303         final int[] rID_in= {R.id.recipe_S1_in,R.id.recipe_S2_in,R.id
    .recipe_S3_in,R.id.recipe_S4_in,
304                 R.id.recipe_S5_in,R.id.recipe_S6_in,R.id.recipe_S7_in
    ,R.id.recipe_S8_in,R.id.recipe_S9_in};
305         mStepTextNum= new TextView[mRecipe.getNbStepMax()];
306         mStepTextEdit=new EditText[mRecipe.getNbStepMax()];
307         for(int i=0;i<mRecipe.getNbStepMax();i++) {
308             mStepTextNum[i] = (TextView) v.findViewById(rID[i]);
309             mStepTextEdit[i] = (EditText) v.findViewById(rID_in[i]);
310         }
311         mStepInc=(ImageButton) v.findViewById(R.id.step_add);
312         updateListStep(v);
313
314         mStepTextEdit[0].addTextChangedListener(new TextWatcher() {
315             @Override
316             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) { }
317             @Override
318             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
319                 mRecipe.setStep(0+1, s.toString());
320             }
321             @Override
322             public void afterTextChanged(Editable s) {}
323         });
324         mStepTextEdit[1].addTextChangedListener(new TextWatcher() {
325             @Override
326             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) { }
327             @Override
328             public void onTextChanged(CharSequence s, int start, int
```

```
328 before, int count) {
329                 mRecipe.setStep(1+1, s.toString());
330             }
331             @Override
332             public void afterTextChanged(Editable s) {}
333         });
334         mStepTextEdit[2].addTextChangedListener(new TextWatcher() {
335             @Override
336             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) { }
337             @Override
338             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
339                 mRecipe.setStep(2+1, s.toString());
340             }
341             @Override
342             public void afterTextChanged(Editable s) {}
343         });
344         mStepTextEdit[3].addTextChangedListener(new TextWatcher() {
345             @Override
346             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) { }
347             @Override
348             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
349                 mRecipe.setStep(3+1, s.toString());
350             }
351             @Override
352             public void afterTextChanged(Editable s) {}
353         });
354         mStepTextEdit[4].addTextChangedListener(new TextWatcher() {
355             @Override
356             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) { }
357             @Override
358             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
359                 mRecipe.setStep(4+1, s.toString());
360             }
361             @Override
362             public void afterTextChanged(Editable s) {}
363         });
364         mStepTextEdit[5].addTextChangedListener(new TextWatcher() {
365             @Override
366             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) { }
367             @Override
368             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
369                 mRecipe.setStep(5+1, s.toString());
370             }
371             @Override
372             public void afterTextChanged(Editable s) {}
373         });
```

```
374         mStepTextEdit[6].addTextChangedListener(new TextWatcher() {
375             @Override
376             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) { }
377             @Override
378             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
379                 mRecipe.setStep(6+1, s.toString());
380             }
381             @Override
382             public void afterTextChanged(Editable s) {}
383         });
384         mStepTextEdit[7].addTextChangedListener(new TextWatcher() {
385             @Override
386             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) { }
387             @Override
388             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
389                 mRecipe.setStep(7+1, s.toString());
390             }
391             @Override
392             public void afterTextChanged(Editable s) {}
393         });
394         mStepTextEdit[8].addTextChangedListener(new TextWatcher() {
395             @Override
396             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) { }
397             @Override
398             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
399                 mRecipe.setStep(8+1, s.toString());
400             }
401             @Override
402             public void afterTextChanged(Editable s) {}
403         });
404
405         mStepInc.setOnClickListener(new View.OnClickListener() {
406             @Override
407             public void onClick(View v) {
408                 updateListStep(v);
409             }
410         });
411
412         final int[] rIngID= {R.id.recipe_I01,R.id.recipe_I02,R.id.
    recipe_I03,R.id.recipe_I04,
413                 R.id.recipe_I05,R.id.recipe_I06,R.id.recipe_I07,R.id.
    recipe_I08,
414                 R.id.recipe_I09,R.id.recipe_I10,R.id.recipe_I11,R.id.
    recipe_I12,
415                 R.id.recipe_I13,R.id.recipe_I14,R.id.recipe_I15};
416         final int[] rIngID_in= {R.id.recipe_I01_in,R.id.recipe_I02_in
    ,R.id.recipe_I03_in,R.id.recipe_I04_in,
417                 R.id.recipe_I05_in,R.id.recipe_I06_in,R.id.
```

```
417 recipe_I07_in,R.id.recipe_I08_in,
418                 R.id.recipe_I09_in,R.id.recipe_I10_in,R.id.
    recipe_I11_in,R.id.recipe_I12_in,
419                 R.id.recipe_I13_in,R.id.recipe_I14_in,R.id.
    recipe_I15_in};
420         mIngTextNum= new TextView[mRecipe.getNbIngMax()];
421         mIngTextEdit=new EditText[mRecipe.getNbIngMax()];
422         for(int i=0;i<mRecipe.getNbIngMax();i++) {
423             mIngTextNum[i] = (TextView) v.findViewById(rIngID[i]);
424             mIngTextEdit[i] = (EditText) v.findViewById(rIngID_in[i])
    ;
425         }
426         mIngInc=(ImageButton) v.findViewById(R.id.ing_add);
427         mIngTextEdit[0].setText(mRecipe.getIngredient(0 + 1));
428         mIngTextEdit[1].setText(mRecipe.getIngredient(1 + 1));
429         mIngTextEdit[2].setText(mRecipe.getIngredient(2 + 1));
430         mIngInc=(ImageButton) v.findViewById(R.id.ing_add);
431         updateListIng(v);
432
433         mIngTextEdit[0].addTextChangedListener(new TextWatcher() {
434             @Override
435             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) { }
436             @Override
437             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
438                 mRecipe.setIngredient(0+1, s.toString());
439             }
440             @Override
441             public void afterTextChanged(Editable s) {}
442         });
443         mIngTextEdit[1].addTextChangedListener(new TextWatcher() {
444             @Override
445             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) { }
446             @Override
447             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
448                 mRecipe.setIngredient(1+1, s.toString());
449             }
450             @Override
451             public void afterTextChanged(Editable s) {}
452         });
453         mIngTextEdit[2].addTextChangedListener(new TextWatcher() {
454             @Override
455             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) { }
456             @Override
457             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
458                 mRecipe.setIngredient(2+1, s.toString());
459             }
460             @Override
461             public void afterTextChanged(Editable s) {}
```

```
462         });
463         mIngTextEdit[3].addTextChangedListener(new TextWatcher() {
464             @Override
465             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) { }
466             @Override
467             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
468                 mRecipe.setIngredient(3+1, s.toString());
469             }
470             @Override
471             public void afterTextChanged(Editable s) {}
472         });
473         mIngTextEdit[4].addTextChangedListener(new TextWatcher() {
474             @Override
475             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) { }
476             @Override
477             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
478                 mRecipe.setIngredient(4+1, s.toString());
479             }
480             @Override
481             public void afterTextChanged(Editable s) {}
482         });
483         mIngTextEdit[5].addTextChangedListener(new TextWatcher() {
484             @Override
485             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) { }
486             @Override
487             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
488                 mRecipe.setIngredient(5+1, s.toString());
489             }
490             @Override
491             public void afterTextChanged(Editable s) {}
492         });
493         mIngTextEdit[6].addTextChangedListener(new TextWatcher() {
494             @Override
495             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) { }
496             @Override
497             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
498                 mRecipe.setIngredient(6+1, s.toString());
499             }
500             @Override
501             public void afterTextChanged(Editable s) {}
502         });
503         mIngTextEdit[7].addTextChangedListener(new TextWatcher() {
504             @Override
505             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) { }
506             @Override
```

```
507             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
508                 mRecipe.setIngredient(7+1, s.toString());
509             }
510             @Override
511             public void afterTextChanged(Editable s) {}
512         });
513         mIngTextEdit[8].addTextChangedListener(new TextWatcher() {
514             @Override
515             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) { }
516             @Override
517             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
518                 mRecipe.setIngredient(8+1, s.toString());
519             }
520             @Override
521             public void afterTextChanged(Editable s) {}
522         });
523         mIngTextEdit[9].addTextChangedListener(new TextWatcher() {
524             @Override
525             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) { }
526             @Override
527             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
528                 mRecipe.setIngredient(9+1, s.toString());
529             }
530             @Override
531             public void afterTextChanged(Editable s) {}
532         });
533         mIngTextEdit[10].addTextChangedListener(new TextWatcher() {
534             @Override
535             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) { }
536             @Override
537             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
538                 mRecipe.setIngredient(10+1, s.toString());
539             }
540             @Override
541             public void afterTextChanged(Editable s) {}
542         });
543         mIngTextEdit[11].addTextChangedListener(new TextWatcher() {
544             @Override
545             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) { }
546             @Override
547             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
548                 mRecipe.setIngredient(11+1, s.toString());
549             }
550             @Override
551             public void afterTextChanged(Editable s) {}
```

```
552         });
553         mIngTextEdit[12].addTextChangedListener(new TextWatcher() {
554             @Override
555             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) { }
556             @Override
557             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
558                 mRecipe.setIngredient(12+1, s.toString());
559             }
560             @Override
561             public void afterTextChanged(Editable s) {}
562         });
563         mIngTextEdit[13].addTextChangedListener(new TextWatcher() {
564             @Override
565             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) { }
566             @Override
567             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
568                 mRecipe.setIngredient(13+1, s.toString());
569             }
570             @Override
571             public void afterTextChanged(Editable s) {}
572         });
573         mIngTextEdit[14].addTextChangedListener(new TextWatcher() {
574             @Override
575             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) { }
576             @Override
577             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
578                 mRecipe.setIngredient(14+1, s.toString());
579             }
580             @Override
581             public void afterTextChanged(Editable s) {}
582         });
583         mIngInc.setOnClickListener(new View.OnClickListener() {
584             @Override
585             public void onClick(View v) {
586                 updateListIng(v);
587             }
588         });
589
590        return v;
591     }
592
593
594     @Override
595     public void onActivityResult(int requestCode, int resultCode,
    Intent data) {
596         if (resultCode != Activity.RESULT_OK) {
597             return;
598         }
```

```
599         if (requestCode==PICK_IMAGE){
600             if (data == null) return ;
601             try{
602             InputStream inputStream = getContext().getContentResolver
    ().openInputStream(data.getData());
603             mBmp= BitmapFactory.decodeStream(inputStream);
604             mPhotoView.setImageBitmap(mBmp);
605             } catch(Exception e) {
606                 deBug("recover data error:"+e);
607             }
608         }
609     }
610 
611     private void updatePhotoView(){
612         if (mPhotoFile==null || !mPhotoFile.exists()){
613             mPhotoView.setImageDrawable(getResources().getDrawable(R.
    drawable.ic_recipe_camera));
614         } else {
615             Bitmap bitmap=PictureUtils.getScaledBitmap(mPhotoFile.
    getPath(), getActivity());
616             mPhotoView.setImageBitmap(bitmap);
617         }
618     }
619 
620     private void updateListStep(View v) {
621         String cas;
622         int i_cur=mRecipe.getNbStep();
623         int i_max=mRecipe.getNbStepMax();
624         for (int i=0;i<mRecipe.getNbStepMax();i++) {
625             cas = "NOTVISIBLE";
626             if (i == 0) cas = "FIRST";
627             if (i < i_cur) cas = "ACTIVE";
628             if ((i == i_cur)&&(i_cur<i_max)) cas = "EMPTY_READY";
629             if ((i == i_cur)&&(i_cur==i_max)) cas = "ACTIVE";
630             switch (cas) {
631                 case "FIRST":
632                     if (mRecipe.getNbStep() == 0) {
633                         mStepTextEdit[i].setText("");
634                         mStepTextEdit[i].setHint(R.string.P3S1H);}
635                     else mStepTextEdit[i].setText(mRecipe.getStep(i +
     1));
636                     break;
637                 case "ACTIVE":
638                     mStepTextNum[i].setVisibility(View.VISIBLE);
639                     mStepTextEdit[i].setVisibility(View.VISIBLE);
640                     mStepTextEdit[i].setText(mRecipe.getStep(i + 1));
641                     mStepTextEdit[i].setHint("");
642                     break;
643                 case "EMPTY_READY":
644                     mStepTextNum[i].setVisibility(View.VISIBLE);
645                     mStepTextEdit[i].setVisibility(View.VISIBLE);
646                     mStepTextEdit[i].setText("");
647                     mStepTextEdit[i].setHint(R.string.P3S1H);
648                     break;
```

```
649                 case "NOTVISIBLE":
650                     mStepTextNum[i].setVisibility(View.GONE);
651                     mStepTextEdit[i].setVisibility(View.GONE);
652                     mStepTextEdit[i].setText("");
653                     mStepTextEdit[i].setHint("");
654             }
655         }
656     }
657
658     private void updateListIng(View v) {
659         String cas;
660         int i_cur=mRecipe.getNbIng();
661         int i_max=mRecipe.getNbIngMax();
662         for (int i=0;i<mRecipe.getNbIngMax();i++) {
663             cas = "NOTVISIBLE";
664             if (i == 0) cas = "FIRST";
665             if (i < i_cur) cas = "ACTIVE";
666             if ((i == i_cur)&&(i_cur<i_max)) cas = "EMPTY_READY";
667             if ((i == i_cur)&&(i_cur==i_max)) cas = "ACTIVE";
668             switch (cas) {
669                 case "FIRST":
670                     if (mRecipe.getNbIng() == 0) {
671                         mIngTextEdit[i].setText("");
672                         mIngTextEdit[i].setHint(R.string.P3IT);}
673                     else mIngTextEdit[i].setText(mRecipe.
    getIngredient(i + 1));
674                     break;
675                 case "ACTIVE":
676                     mIngTextNum[i].setVisibility(View.VISIBLE);
677                     mIngTextEdit[i].setVisibility(View.VISIBLE);
678                     mIngTextEdit[i].setText(mRecipe.getIngredient(i +
     1));
679                     mIngTextEdit[i].setHint("");
680                     break;
681                 case "EMPTY_READY":
682                     mIngTextNum[i].setVisibility(View.VISIBLE);
683                     mIngTextEdit[i].setVisibility(View.VISIBLE);
684                     mIngTextEdit[i].setText("");
685                     mIngTextEdit[i].setHint(R.string.P3IT);
686                     break;
687                 case "NOTVISIBLE":
688                     mIngTextNum[i].setVisibility(View.GONE);
689                     mIngTextEdit[i].setVisibility(View.GONE);
690                     mIngTextEdit[i].setText("");
691                     mIngTextEdit[i].setHint("");
692             }
693         }
694     }
695
696     private Boolean IsRecipeNew(UUID uuid){
697         return uuid.toString().equals(UUIDNULL);
698     }
699
700     private Boolean ResizePhoto(Recipe r) {
```

```
701         File f = CookBook.get(getActivity()).getPhotoFile(r);
702         if (f==null || !f.exists()){
703             deBug( "No file from Cookbook for this recipe :" + r.
    getTitle()+" Error CB004");
704             return false;
705         } else {
706             Bitmap bitmap=PictureUtils.getScaledBitmap(mPhotoFile.
    getPath(), getActivity(), 600);
707             f.delete();
708             try {
709                 if(!f.createNewFile()) {
710                     deBug("File not deleted ! Error CB001");
711                     return false;
712                 }
713             } catch (IOException e) {
714                 deBug("Error in creating new file : "+e+" Error CB002
    ");
715                 return false;
716             }
717             try {
718
719                 FileOutputStream fOut = new FileOutputStream(f);
720                 bitmap.compress(Bitmap.CompressFormat.JPEG, 100, fOut
    );
721                 fOut.flush();
722                 fOut.close();
723                 return true;
724             } catch (IOException e) {
725                 deBug("Error in reducing and saving file "+" Error
    CB003");
726                 return false;
727             }
728         }
729     }
730     private void deBug(String s){
731         // Log(TAG, s);
732     }
733 }
734
```

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

```java
package com.fdx.cookbook;

import android.support.v4.app.Fragment;

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

```

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

```
 1 package com.fdx.cookbook;
 2
 3 import android.app.Activity;
 4 import android.content.Intent;
 5 import android.graphics.Bitmap;
 6 import android.net.Uri;
 7 import android.os.AsyncTask;
 8 import android.os.Bundle;
 9 import android.support.annotation.NonNull;
10 import android.support.v4.app.Fragment;
11 import android.support.v4.app.FragmentManager;
12 import android.support.v4.view.MenuCompat;
13 import android.support.v7.app.AppCompatActivity;
14 import android.support.v7.widget.LinearLayoutManager;
15 import android.support.v7.widget.RecyclerView;
16 import android.support.v7.widget.SearchView;
17 import android.util.Log;
18 import android.view.LayoutInflater;
19 import android.view.Menu;
20 import android.view.MenuInflater;
21 import android.view.MenuItem;
22 import android.view.View;
23 import android.view.ViewGroup;
24 import android.widget.ImageView;
25 import android.widget.RatingBar;
26
27 import android.widget.TextView;
28 import android.widget.Toast;
29
30 import java.io.File;
31 import java.text.DecimalFormat;
32 import java.util.ArrayList;
33 import java.util.Collections;
34 import java.util.List;
35 import java.util.UUID;
36
37 public class RecipeListFragment extends Fragment {
38     private RecyclerView mRecipeRecyclerView;
39     private RecipeAdapter mAdapter;
40     private SessionInfo mSession;
41     private Recipe mRecipeInit;
42     private MenuItem mMessageItem;
43     private static final String SAVED_SORT_STATUS="sort";
44     private AsyncCallClass mInstanceAsync;
45     private ListMask mListMask;
46     private static final String TAG = "CB_ListFra";
47     private static final String DIALOG_RATE = "DialogRate";
48     private static final int REQUEST_RATE = 0;
49     private static final String UUIDNULL="00000000-0000-0000-0000-
   000000000000";
50
51     @Override
52     public void onCreate(Bundle savedInstanceState) {
53         super.onCreate(savedInstanceState);
```

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

```
 98     public void onPause(){
 99         super.onPause();
100         mSession.setListMask(mListMask.convertToString());
101     }
102 
103     @Override
104     public void onSaveInstanceState(Bundle outState) {
105         super.onSaveInstanceState(outState);
106         outState.putString(SAVED_SORT_STATUS, mListMask.
    convertToString());
107     }
108 
109     @Override
110     public void onCreateOptionsMenu(Menu menu, MenuInflater inflater)
    {
111         super.onCreateOptionsMenu(menu, inflater);
112         inflater.inflate(R.menu.fragment_recipe_list, menu);
113         mMessageItem = menu.findItem(R.id.new_mail);
114         CookBook cookbook=CookBook.get(getActivity());
115         mMessageItem.setVisible(cookbook.isThereMail());
116         MenuCompat.setGroupDividerEnabled(menu, true);
117         MenuItem searchItem = menu.findItem(R.id.menu_item_search);
118         final SearchView searchView = (SearchView) searchItem.
    getActionView();
119         searchView.setOnQueryTextListener(new SearchView.
    OnQueryTextListener() {
120             @Override
121             public boolean onQueryTextSubmit(String s) {
122                 Toast.makeText(getContext(), getString(mListMask.
    toggleSearch(s)),
123                         Toast.LENGTH_SHORT ).show();
124                 updateUI();
125                 return true;
126             }
127             @Override
128             public boolean onQueryTextChange(String s) {
129                 if (s.equals("")){
130                     Toast.makeText(getContext(), getString(mListMask.
    toggleSearch(s)),
131                         Toast.LENGTH_SHORT ).show();
132                     updateUI();
133                 }
134                 return false;
135             }
136         });
137     }
138 
139     @Override
140     public boolean onOptionsItemSelected(MenuItem item){
141         switch (item.getItemId()){
142             case R.id.new_recipe:
143                 Intent intent= RecipeActivity.newIntent(getActivity()
    , UUID.fromString( UUIDNULL));
144                 startActivity(intent);
```

```
145                 return true;
146             case R.id.list_logout:
147                 mSession.setReqNewSession(true);
148                 Intent intent2=new Intent(getActivity().
    getApplicationContext(), SplashActivity.class);
149                 startActivity(intent2);
150                 return true;
151             case R.id.list_sync:
152                 startSync();
153                 return true;
154             case R.id.new_mail:
155                 CookBook cookbook=CookBook.get(getActivity());
156                 if (cookbook.isThereMail()) {
157                 Intent intent3= RecipeMailDisplayActivity.newIntent(
    getActivity());
158                 startActivity(intent3); } else {
159                 mMessageItem.setVisible(cookbook.isThereMail());}
160                 return true;
161             case R.id.list_filter:
162                 mListMask.reset();
163                 updateUI();
164                 return true;
165             case R.id.feedback:
166                 Intent intent3 = new Intent(Intent.ACTION_SENDTO);
167                 intent3.putExtra(Intent.EXTRA_SUBJECT, "Feedback on
    application");
168                 intent3.putExtra(Intent.EXTRA_TEXT, "Your comment");
169                 intent3.setData(Uri.parse("mailto:cookbookfamily.
    founder@gmail.com"));
170                 intent3.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
171                 startActivity(intent3);
172                 return true;
173             default:
174                 return super.onOptionsItemSelected(item);
175         }
176     }
177
178     private void startSync(){
179         AsyncTask.Status as=mInstanceAsync.getStatus();
180         if (as== AsyncTask.Status.RUNNING){
181         }
182         if (as== AsyncTask.Status.PENDING){
183             mInstanceAsync.execute();
184         }
185         if (as== AsyncTask.Status.FINISHED){
186             mInstanceAsync=new AsyncCallClass(getContext());
187             mInstanceAsync.execute();
188         }
189     }
190
191     private void updateUI() {
192         CookBook cookbook=CookBook.get(getActivity());
193         List<Recipe> recipes_in=cookbook.getRecipes();
194         List<Recipe> recipes=new ArrayList<>();
```

```
195         recipes.addAll(recipes_in);
196         for(Recipe r:recipes_in){
197             if (!mListMask.isRecipeSelected(r)) recipes.remove(r);
198         }
199         if (recipes.isEmpty()){
200             Integer message;
201             if (cookbook.getRecipesVisibles().isEmpty())
202                 message = (cookbook.isThereMail() ? R.string.TEW : R.
    string.TEZ);
203             else message=R.string.TEY;
204             Toast.makeText(getContext(), getString(message), Toast.
    LENGTH_LONG).show();
205         } else {
206             if (mListMask.isTitleSorted()) {
207                 Collections.sort(recipes,
208                         (r1, r2) -> {
209                             return (r1.getTitle().compareTo(r2.
    getTitle()));
210                         });
211             } else {
212                 Collections.sort(recipes,
213                         (r1, r2) -> {
214                             return (r2.getDate().compareTo(r1.getDate
    ()));
215                         });
216             }
217             if (mListMask.isNoteSorted()) {
218                 Collections.sort(recipes,
219                         (r1, r2) -> {
220                             return (r2.getNbNotes()-r1.getNbNotes()+(
    int)(100* (r2.getNoteAvg() - r1.getNoteAvg())));
221                         });
222             }
223         }
224         if (mAdapter==null){
225             mAdapter=new RecipeAdapter(recipes);
226             mRecipeRecyclerView.setAdapter(mAdapter);
227         } else {
228             mAdapter.setRecipes(recipes);
229             mAdapter.notifyDataSetChanged();
230         }
231     }
232
233     private class RecipeHolder extends RecyclerView.ViewHolder
234                                     implements View.OnClickListener {
235         private TextView mTitleTextView;
236         private TextView mSourceTextView;
237         private TextView mNoteTextView;
238         private TextView mRatingTextView;
239         private TextView mDifficulty;
240         private TextView mType;
241         private ImageView mEditIcon;
242         private ImageView mPhotoView;
243         private ImageView mSunIcon;
```

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

```
280                 }
281             });
282             mNoteTextView.setOnClickListener(new View.OnClickListener
    (){
283                 @Override
284                 public void onClick(View v){
285                     Toast.makeText(getContext(), getString(mListMask.
    toggleSortNote()),
286                             Toast.LENGTH_SHORT ).show();
287                     updateUI();
288                 }
289             });
290             mSourceTextView.setOnClickListener(new View.
    OnClickListener() {
291                 @Override
292                 public void onClick(View v) {
293                     Toast.makeText(getContext(), getString(mListMask.
    toggleUser(mRecipe.getOwner()),
294                             mRecipe.getOwner().getName()),Toast.
    LENGTH_SHORT ).show();
295                     updateUI();
296                 }
297             });
298             mSunIcon.setOnClickListener(new View.OnClickListener() {
299                 @Override
300                 public void onClick(View v) {
301                     Toast.makeText(getContext(), getString(mListMask.
    toggleSun()),Toast.LENGTH_SHORT ).show();
302                     updateUI();
303                 }
304             });
305             mIceIcon.setOnClickListener(new View.OnClickListener() {
306                 @Override
307                 public void onClick(View v) {
308                     Toast.makeText(getContext(), getString(mListMask.
    toggleWinter()),
309                             Toast.LENGTH_SHORT ).show();
310                     updateUI();
311                 }
312             });
313             mDifficulty.setOnClickListener(new View.OnClickListener()
     {
314                 @Override
315                 public void onClick(View v) {
316                     Toast.makeText(getContext(),
317                             getString(mListMask.toggleDifficulty(
    mRecipe.getDifficulty())),
318                             Toast.LENGTH_SHORT ).show();
319                     updateUI();
320                 }
321             });
322             mType.setOnClickListener(new View.OnClickListener() {
323                 @Override
324                 public void onClick(View v) {
```

```
325                     Toast.makeText(getContext(),
326                             getString(mListMask.toggleType(mRecipe.
    getType())),
327                             Toast.LENGTH_SHORT ).show();
328                     updateUI();
329                 }
330             });
331             mEditIcon.setOnClickListener(new View.OnClickListener() {
332                 @Override
333                 public void onClick(View v) {
334                     Intent intent= RecipeActivity.newIntent(
    getActivity(),mRecipe.getId());
335                     startActivity(intent);
336                 }
337             });
338             mPhotoView.setOnClickListener(new View.OnClickListener()
    {
339                 @Override
340                 public void onClick(View v) {
341                     Intent intent=RecipeDisplayActivity.newIntent(
    getActivity(),mRecipe.getId());
342                     startActivity(intent);
343                 }
344             });
345         }
346         public void bind(Recipe recipe){
347             mRecipe=recipe;
348             mTitleTextView.setText(mRecipe.getTitle());
349             mSourceTextView.setText(getString(R.string.P1_de, mRecipe
    .getOwner().getName()));
350             //mSourceTextView.setText(mRecipe.getFlag()); // for
    debug
351             mRating.setRating((float) mRecipe.getNoteAvg());
352             DecimalFormat df = new DecimalFormat("#.#");
353             mNoteTextView.setText(df.format(mRecipe.getNoteAvg())+
    " ("+mRecipe.getNbNotes()+")");
354             mPhotoFile=CookBook.get(getActivity()).getPhotoFile(
    mRecipe);
355             int idff= RecipeDifficulty.getIndex(mRecipe.getDifficulty
    ());
356             String[] stringArray = getResources().getStringArray(R.
    array.recipe_difficulty_array);
357             mDifficulty.setText(stringArray[idff]);
358             int ityp= RecipeType.getIndex(mRecipe.getType());
359             String[] stringArrayTyp = getResources().getStringArray(R
    .array.recipe_type_array);
360             mType.setText(stringArrayTyp[ityp]);
361             if (mPhotoFile==null || !mPhotoFile.exists()){
362                 mPhotoView.setImageDrawable(getResources().
    getDrawable(R.drawable.ic_recipe_see));
363             } else {
364                 Bitmap bitmap=PictureUtils.getScaledBitmap(mPhotoFile
    .getPath(), getActivity());
365                 mPhotoView.setImageBitmap(bitmap);
```

```
366             }
367             mIceIcon.setImageResource((mRecipe.IsWinter()) ? R.
    drawable.ic_recipe_ice : R.drawable.ic_recipe_ice_disabled);
368             mSunIcon.setImageResource((mRecipe.IsSummer()) ? R.
    drawable.ic_recipe_sun : R.drawable.ic_recipe_sun_disabled);
369             mEditIcon.setVisibility((mRecipe.getOwner().getId().
    equals(mSession.getUser().getId())) ? View.VISIBLE : View.GONE);
370         }
371
372         @Override
373         public void onClick(View v){
374
375         }
376     }
377
378     private class RecipeAdapter extends RecyclerView.Adapter<
    RecipeHolder> {
379         private List<Recipe> mRecipes;
380         public RecipeAdapter(List<Recipe> recipes){
381             mRecipes=recipes;
382         }
383
384         @NonNull
385         @Override
386         public RecipeHolder onCreateViewHolder(@NonNull ViewGroup
    parent, int viewType) {
387             LayoutInflater layoutInflater=LayoutInflater.from(
    getActivity());
388             return new RecipeHolder(layoutInflater, parent);
389         }
390
391         @Override
392         public void onBindViewHolder(@NonNull RecipeHolder
    recipeHolder, int i) {
393             Recipe recipe=mRecipes.get(i);
394             recipeHolder.bind(recipe);
395         }
396
397         @Override
398         public int getItemCount() {
399             return mRecipes.size();
400         }
401
402         public void setRecipes(List<Recipe> recipes){
403             mRecipes=recipes;
404         }
405     }
406 }
407
```

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

```
 1 package com.fdx.cookbook;
 2
 3 import android.database.Cursor;
 4 import android.database.CursorWrapper;
 5 import android.util.Log;
 6
 7 import java.net.MalformedURLException;
 8 import java.net.URL;
 9 import java.util.Date;
10 import java.util.UUID;
11
12 import static com.fdx.cookbook.RecipeDbSchema.*;
13
14 public class RecipeCursorWrapper extends CursorWrapper {
15     public RecipeCursorWrapper(Cursor cursor){
16         super(cursor);
17     }
18     private static final String TAG = "DebugCursorWrapper";
19
20     public Recipe getRecipe(){
21         String uuidString = getString(getColumnIndex(RecipeTable.Cols.
   UUID));
22         Recipe r=new Recipe(UUID.fromString(uuidString));
23         String ownerString = getString(getColumnIndex(RecipeTable.Cols
   .OWNER));
24         r.getOwnerDeserialized(ownerString);
25         String title = getString(getColumnIndex(RecipeTable.Cols.TITLE
   ));
26         r.setTitle(title);
27         String source = getString(getColumnIndex(RecipeTable.Cols.
   SOURCE));
28         r.setSource(source);
29         String sourceURL=getString(getColumnIndex(RecipeTable.Cols.
   SOURCE_URL));
30         try {
31             URL url = new URL(sourceURL);
32             r.setSource_url(url);}
33             catch (MalformedURLException e) {
34                 //fdx Log(TAG, "getURL from cursor failed");
35             }
36         long date = getLong(getColumnIndex(RecipeTable.Cols.DATE));
37         r.setDate(new Date(date));
38         date = getLong(getColumnIndex(RecipeTable.Cols.DATE_PHOTO));
39         r.setDatePhoto(new Date(date));
40         int nbpers=getInt(getColumnIndex(RecipeTable.Cols.NBPERS));
41         r.setNbPers(nbpers);
42         String[] step= new String[r.getNbStepMax()];
43         for(int i=0;i<r.getNbStepMax();i++){
44             step[i]=getString(getColumnIndex(RecipeTable.Cols.STEP[i])
   );
45             r.setStep(i+1, step[i]);
46         }
47         String[] ing=new String[r.getNbIngMax()];
48         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 type = getString(getColumnIndex(RecipeTable.Cols.TYPE)
    );
57          r.setType(RecipeType.valueOf(type));
58          String comments = getString(getColumnIndex(RecipeTable.Cols.
    COMMENTS));
59          r.getCommentsDeserialised(comments);
60          String status = getString(getColumnIndex(RecipeTable.Cols.
    STATUS));
61          r.setStatus(StatusRecipe.valueOf(status));
62          String notes = getString(getColumnIndex(RecipeTable.Cols.
    NOTES));
63          r.getNotesDeserialised(notes);
64          String message = getString(getColumnIndex(RecipeTable.Cols.
    MESSAGE));
65          r.setMessage(message);
66          String fromString = getString(getColumnIndex(RecipeTable.Cols
    .MESSAGE_FROM));
67          r.getFromDeserialized(fromString);
68          int tsrecipe = getInt(getColumnIndex(RecipeTable.Cols.
    TS_RECIPE));
69          r.updateTS(AsynCallFlag.NEWRECIPE, (tsrecipe==1));
70          int tsphoto = getInt(getColumnIndex(RecipeTable.Cols.TS_PHOTO
    ));
71          r.updateTS(AsynCallFlag.NEWPHOTO, (tsphoto==1));
72          int tscomment = getInt(getColumnIndex(RecipeTable.Cols.
    TS_COMMENT));
73          r.updateTS(AsynCallFlag.NEWCOMMENT, (tscomment==1));
74          int tsnote = getInt(getColumnIndex(RecipeTable.Cols.TS_NOTE))
    ;
75          r.updateTS(AsynCallFlag.NEWRATING, (tsnote==1));
76          return r;
77      }
78  }
79
```

```java
package com.fdx.cookbook;

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

import java.util.UUID;

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

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

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

```

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

```
 1 package com.fdx.cookbook;
 2
 3 import android.content.DialogInterface;
 4 import android.content.Intent;
 5 import android.graphics.Bitmap;
 6 import android.os.AsyncTask;
 7 import android.os.Bundle;
 8 import android.support.v4.app.Fragment;
 9 import android.support.v7.app.AlertDialog;
10 import android.text.Editable;
11 import android.text.TextWatcher;
12 import android.view.LayoutInflater;
13 import android.view.Menu;
14 import android.view.MenuInflater;
15 import android.view.MenuItem;
16 import android.view.View;
17 import android.view.ViewGroup;
18 import android.widget.EditText;
19 import android.widget.ImageView;
20 import android.widget.LinearLayout;
21 import android.widget.RatingBar;
22 import android.widget.ScrollView;
23 import android.widget.TextView;
24 import android.widget.Toast;
25
26 import java.io.File;
27 import java.io.InputStream;
28 import java.net.HttpURLConnection;
29 import java.net.URL;
30 import java.text.DecimalFormat;
31 import java.util.HashMap;
32 import java.util.UUID;
33 import java.util.regex.Pattern;
34
35 public class RecipeDisplayFragment extends Fragment {
36     private static final String ARG_RECIPE_ID="recipe_id";
37     private static final String TAG = "CB_RecipeDisplayFrag";
38     private Recipe mRecipe;
39     private File mPhotoFile;
40     private int mStepNb;
41     private int mIngNb;
42     private String newcomment;
43     private String DEFAULT_URL="https://wwww.cookbookfamily.com";
44     private SessionInfo mSession;
45     private ImageView mDPhotoView;
46     private TextView mDTitleText;
47     private RatingBar mDRatingBar;
48     private TextView mDRatingBarText;
49     private TextView mDAuthorText;
50     private TextView mDDifficulty;
51     private TextView mDType;
52     private ImageView mSunIcon;
53     private ImageView mIceIcon;
54     private TextView mDSourceText;
```

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

```
104         inflater.inflate(R.menu.fragment_recipe_display, menu);
105         MenuItem menuItem = menu.findItem(R.id.recipe_menu_edit);
106         if(!mRecipe.getOwner().getId().equals(mSession.getUser().
    getId())){
107             menuItem.setVisible(false);
108         }
109     }
110 
111     @Override
112     public boolean onOptionsItemSelected(MenuItem item) {
113         switch (item.getItemId()) {
114             case R.id.recipe_menu_report:
115                 Intent i=new Intent(Intent.ACTION_SEND);
116                 i.setType("text/plain");
117                 i.putExtra(Intent.EXTRA_TEXT, getRecipeReport());
118                 i.putExtra(Intent.EXTRA_SUBJECT,getString(R.string.
    P2MU_send));
119                 startActivity(i);
120                 return true;
121             case R.id.recipe_menu_edit:
122                 Intent intent= RecipeActivity.newIntent(getActivity()
    ,mRecipe.getId());
123                 startActivity(intent);
124                 return true;
125             case R.id.recipe_mail:
126                 LinearLayout layout = new LinearLayout(getContext());
127                 layout.setOrientation(LinearLayout.VERTICAL);
128                 final EditText familyBox = new EditText(getContext())
    ;
129                 familyBox.setHint(getString(R.string.P0HF));
130                 layout.addView(familyBox);
131                 final EditText memberBox = new EditText(getContext())
    ;
132                 memberBox.setHint(getString(R.string.P0HM));
133                 layout.addView(memberBox);
134                 final EditText messageBox = new EditText(getContext()
    );
135                 messageBox.setHint(getString(R.string.P4M_mes));
136                 layout.addView(messageBox);
137                 AlertDialog.Builder builder = new AlertDialog.Builder
    (getContext());
138                 builder.setTitle(getString(R.string.P4M_title));
139                 builder.setView(layout);
140                 builder.setPositiveButton("OK", new DialogInterface.
    OnClickListener() {
141                     @Override
142                     public void onClick(DialogInterface dialog, int
    which) {
143                         mToFamily = familyBox.getText().toString();
144                         mToMember = memberBox.getText().toString();
145                         mToMessage = messageBox.getText().toString();
146                         Boolean b=(Pattern.matches(REGEX_FAMILY,
    mToFamily));
147                         b=b&&(Pattern.matches(REGEX_MEMBER,mToMember)
```

```
147 );
148                         b=b&&(IsLenOK(mToFamily,MINMAX[0][0],MINMAX[0
    ][1]));
149                         b=b&&(IsLenOK(mToMember,MINMAX[1][0],MINMAX[1
    ][1]));
150                         b=b&&(Pattern.matches(REGEX_FAMILY,mToMessage
    ));
151                         if (b) {
152                         sendMailAsync sendmail = new sendMailAsync();
153                         sendmail.execute();
154                         } else {
155                             Toast.makeText(getActivity(),getString(R.
    string.P4M_err), Toast.LENGTH_SHORT).show();
156                         }
157                     }
158                 });
159                 builder.setNegativeButton("Cancel", new
    DialogInterface.OnClickListener() {
160                     @Override
161                     public void onClick(DialogInterface dialog, int
    which) {
162                         dialog.cancel();
163                     }
164                 });
165                 builder.show();
166                 return true;
167             case R.id.recipe_menu_delete:
168                 CookBook.get(getActivity()).markRecipeToDelete(
    mRecipe);
169                 getActivity().onBackPressed();
170             default:
171                 return super.onOptionsItemSelected(item);
172         }
173     }
174
175     @Override
176     public View onCreateView(LayoutInflater inflater, ViewGroup
    container, Bundle savedInstanceState){
177         final View v=inflater.inflate(R.layout.
    fragment_display_recipe, container, false);
178         mScroll=(ScrollView) v.findViewById(R.id.
    fragment_recipe_scroll);
179         mDTitleText =(TextView) v.findViewById(R.id.
    recipe_display_title);
180         mDPhotoView=(ImageView) v.findViewById(R.id.
    recipe_display_photo);
181         updatePhotoView();
182         mDRatingBar=(RatingBar) v.findViewById(R.id.
    recipe_display_ratingBar);
183         mDRatingBarText=(TextView) v.findViewById(R.id.
    recipe_display_ratingBar_txt);
184         mDDifficulty=(TextView) v.findViewById(R.id.
    recipe_display_difficulty);
185         mDType=(TextView) v.findViewById(R.id.recipe_display_type);
```

```
186         mSunIcon=(ImageView) v.findViewById(R.id.recipe_img_sun) ;
187         mIceIcon=(ImageView) v.findViewById(R.id.recipe_img_ice) ;
188         mDAuthorText=(TextView) v.findViewById(R.id.
    recipe_display_author);
189         mDSourceText=(TextView) v.findViewById(R.id.
    recipe_display_source);
190         mDSourceUrl=(TextView) v.findViewById(R.id.
    recipe_display_source_url);
191         mDIngTitle=(TextView) v.findViewById(R.id.
    recipe_display_title_ing);
192         mDStepTitle=(TextView) v.findViewById(R.id.
    recipe_display_title_step);
193         mDSourceTitle=(TextView) v.findViewById(R.id.
    recipe_display_source_title);
194         mDComTitle=(TextView)v.findViewById(R.id.
    recipe_display_comment_title);
195         final int[] rIngID= {R.id.recipe_display_I01,R.id.
    recipe_display_I02,R.id.recipe_display_I03,
196                 R.id.recipe_display_I04,R.id.recipe_display_I05,R.id.
    recipe_display_I06,
197                 R.id.recipe_display_I07,R.id.recipe_display_I08,R.id.
    recipe_display_I09,
198                 R.id.recipe_display_I10,R.id.recipe_display_I11,R.id.
    recipe_display_I12,
199                 R.id.recipe_display_I13,R.id.recipe_display_I14,R.id.
    recipe_display_I15};
200         mDIngText= new TextView[mRecipe.getNbIngMax()];
201         for(int i=0;i<mRecipe.getNbIngMax();i++) {
202             mDIngText[i] = (TextView) v.findViewById(rIngID[i]);
203         }
204         final int[] rID= {R.id.recipe_display_S1,R.id.
    recipe_display_S2,R.id.recipe_display_S3,
205                 R.id.recipe_display_S4,R.id.recipe_display_S5,R.id.
    recipe_display_S6,
206                 R.id.recipe_display_S7,R.id.recipe_display_S8,R.id.
    recipe_display_S9};
207         mDStepText=new TextView[mRecipe.getNbStepMax()];
208         for(int i=0;i<mRecipe.getNbStepMax();i++) {
209             mDStepText[i] = (TextView) v.findViewById(rID[i]);
210         }
211         final int[] rComID= {R.id.recipe_display_C01,R.id.
    recipe_display_C02,R.id.recipe_display_C03,
212                 R.id.recipe_display_C04,R.id.recipe_display_C05,R.id.
    recipe_display_C06,
213                 R.id.recipe_display_C07,R.id.recipe_display_C08,R.id.
    recipe_display_C09,
214                 R.id.recipe_display_C10,R.id.recipe_display_C11,R.id.
    recipe_display_C12,
215                 R.id.recipe_display_C13,R.id.recipe_display_C14,R.id.
    recipe_display_C15,
216                 R.id.recipe_display_C16,R.id.recipe_display_C17,R.id.
    recipe_display_C18,
217                 R.id.recipe_display_C19,R.id.recipe_display_C20};
218         mDComText=new TextView[mRecipe.getNbComMax()];
```

```
219         for(int i=0;i<mRecipe.getNbComMax();i++) {
220             mDComText[i] = (TextView) v.findViewById(rComID[i]);
221         }
222         updateUI();
223         mDNexComment=(EditText) v.findViewById(R.id.
    recipe_display_enter_comment);
224         mDNexComment.setText("");
225         mDNexComment.setHint(getString(R.string.P2C_hint));
226         //todo test comment versus pattern
227         mDNexComment.addTextChangedListener(new TextWatcher() {
228             @Override
229             public void beforeTextChanged(CharSequence s, int start,
    int count, int after) {
230
231             }
232
233             @Override
234             public void onTextChanged(CharSequence s, int start, int
    before, int count) {
235                 newcomment=s.toString();
236             }
237
238             @Override
239             public void afterTextChanged(Editable s) {
240
241             }
242         });
243         mDEnterComment=(ImageView) v.findViewById(R.id.
    recipe_img_enter);
244         mDEnterComment.setOnClickListener(new View.OnClickListener()
    {
245             @Override
246             public void onClick(View v) {
247                 mRecipe.addComment(new Comment(newcomment, mSession.
    getUser()));
248                 mRecipe.updateTS(AsynCallFlag.NEWCOMMENT,true);
249                 mDNexComment.setText("...");
250                 updateUI();
251             }
252         });
253         return v;
254     }
255     private void updatePhotoView(){
256         if (mPhotoFile==null || !mPhotoFile.exists()){
257             mDPhotoView.setImageDrawable(getResources().getDrawable(R
    .drawable.ic_recipe_camera));
258         } else {
259             Bitmap bitmap=PictureUtils.getScaledBitmap(mPhotoFile.
    getPath(), getActivity());
260             mDPhotoView.setImageBitmap(bitmap);
261         }
262     }
263     private void updateUI(){
264         String s,s2;
```

```
265         User u=mRecipe.getOwner();
266         int ingMax=mRecipe.getNbIngMax();
267         int stepMax=mRecipe.getNbStepMax();
268         int comMax=mRecipe.getNbComMax();
269         int NbCom=mRecipe.getComments().size();
270         int gone=View.GONE, visible=View.VISIBLE;
271         DecimalFormat df = new DecimalFormat("#.#");
272         mDTitleText.setText(mRecipe.getTitle());
273         mDRatingBar.setRating((float) mRecipe.getNoteAvg());
274         mDRatingBarText.setText(df.format(mRecipe.getNoteAvg())+"  ("
    + mRecipe.getNotes().size()+")");
275         s=getString(R.string.P2U_txt,u.getName(),u.getFamily());
276         int idiff= RecipeDifficulty.getIndex(mRecipe.getDifficulty())
    ;
277         String[] stringArrayDiff = getResources().getStringArray(R.
    array.recipe_difficulty_array);
278         mDDifficulty.setText(stringArrayDiff[idiff]);
279         int ityp= RecipeType.getIndex(mRecipe.getType());
280         String[] stringArrayTyp = getResources().getStringArray(R.
    array.recipe_type_array);
281         mDType.setText(stringArrayTyp[ityp]);
282         mSunIcon.setImageResource((mRecipe.IsSummer()) ? R.drawable.
    ic_recipe_sun : R.drawable.ic_recipe_sun_disabled);
283         mIceIcon.setImageResource((mRecipe.IsWinter()) ? R.drawable.
    ic_recipe_ice : R.drawable.ic_recipe_ice_disabled);
284         mDAuthorText.setText(s);
285         // source
286         s=mRecipe.getSource();
287         s2=mRecipe.getSource_url_name();
288         if (s.equals("")) mDSourceText.setVisibility(gone);
289         else mDSourceText.setText(s);
290         if ((s2.equals(DEFAULT_URL)) || (s2.equals(""))) {
291             mDSourceUrl.setVisibility(gone);
292             if (s.equals(""))  mDSourceTitle.setVisibility(gone);
293         }
294         else mDSourceUrl.setText(s2);
295         // ingredients
296         s=getString(R.string.P2IT, ""+mRecipe.getNbPers());
297         if (mRecipe.getNbIng()>0) mDIngTitle.setText(s); else
    mDIngTitle.setVisibility(gone);
298         for(int i=0;i<ingMax;i++){
299             if (mIngNb>0){mDIngText[i].setText("- "+mRecipe.
    getIngredient(i+1));}
300             if (i>=0){mDIngText[i].setVisibility((mIngNb>i)? visible:
    gone);}
301         }
302         // steps
303         if (mRecipe.getNbStep()==0)  mDStepTitle.setVisibility(gone);
304         for(int i=0;i<stepMax;i++){
305             if (mStepNb>0){mDStepText[i].setText((i+1)+". "+mRecipe.
    getStep(i+1));}
306             if (i>=0){mDStepText[i].setVisibility((mStepNb>i)?
    visible:gone);}
307         }
```

```
308         // comments
309         s=getString(R.string.P2CT, ""+mRecipe.getComments().size());
310         mDComTitle.setText(s);
311         for(int i=0;i<comMax;i++){
312             if (NbCom>0){
313                 if (i<NbCom) {mDComText[i].setText("- "+mRecipe.
    getComment(NbCom-i-1).toTxt());}
314                 else {mDComText[i].setText("");}
315             }
316             if (i>=0){mDComText[i].setVisibility((NbCom>i)? visible:
    gone);}
317         }
318     }
319     private Boolean IsLenOK(String s, int min, int max){
320         int l=s.length();
321         return ((l>min)&&(l<max));
322     }
323     private String getRecipeReport(){
324         String report;
325         Integer iplus;
326         report =getString(R.string.P2RE_title, mRecipe.getTitle())+"\
    n";
327         report +=getString(R.string.P2RE_user,mRecipe.getOwner().
    getNameComplete())+"\n";
328         if(!mRecipe.getSource().equals("")){
329             report +=  getString(R.string.P2RE_src, mRecipe.getSource
    ())+"\n";
330         }
331         if(!mRecipe.getSource_url_name().equals("")){
332             report +=  getString(R.string.P2RE_url, mRecipe.
    getSource_url_name())+"\n";
333         }
334         if (mRecipe.getNbIng()>0){
335             report +=getString(R.string.P1R_ing)+"\n";
336             for(int i=0;i<mRecipe.getNbIng();i++){
337                 report += getString(R.string.P2RE_ing, mRecipe.
    getIngredient(i+1))+"\n";
338             }
339         }
340         if (mRecipe.getNbStep()>0) {
341             report +=getString(R.string.P1R_step)+"\n";
342             for (int i = 0; i < mRecipe.getNbStep(); i++) {
343                 iplus = i + 1;
344                 report += getString(R.string.P2RE_step, iplus + "",
345                         mRecipe.getStep(i + 1)) + "\n";
346             }
347         }
348         report +=getString(R.string.P2RE_end);
349         return report;
350     }
351
352     /**
    ************************************************************************
    ******************
```

```
353      *                                ASYNC                                 *
354     ************************************************************************
        **********************/
355
356
357     class sendMailAsync extends AsyncTask<Void, Void, Boolean> {
358         private static final String PHP204 = "return204.php";
359         private static final String MYSQLDATEFORMAT="yyyy-MM-dd HH:mm
        :ss";
360         private static final String PHPSENDMAIL = "sendmail.php";
361         @Override
362         protected void onPreExecute() {
363             super.onPreExecute();
364             Toast.makeText(getActivity(),getString(R.string.P4_start)
        , Toast.LENGTH_SHORT).show();
365         }
366
367         @Override
368         protected void onPostExecute(Boolean b) {
369             super.onPostExecute(b);
370             if (b) {
371                 Toast.makeText(getActivity(), getString(R.string.
        P4_OK,mToMember,mToFamily), Toast.LENGTH_LONG).show();
372             }
373             else {
374                 Toast.makeText(getActivity(), getString(R.string.
        P4_fail), Toast.LENGTH_LONG).show();
375             }
376          }
377
378         @Override
379         protected Boolean doInBackground(Void... voids) {
380             NetworkUtils mNetUtils = new NetworkUtils(getContext());
381             if (!test204()) {
382                 return false;
383             }
384             HashMap<String, String> data = new HashMap<>();
385             data.put("idrecipe", mRecipe.getId().toString());
386             if ((mToFamily==null)||(mToFamily.length()<5)) return
        false;
387             data.put("family",mToFamily.trim());
388             if ((mToMember==null)||(mToMember.length()<5)) return
        false;
389             data.put("membre", mToMember.trim());
390             data.put("idfrom", mSession.getUser().getId().toString())
        ;
391             if (mToMessage==null) return false;
392             data.put("message", mToMessage.trim());
393             String result = mNetUtils.sendPostRequestJson(mSession.
        getURLPath() + PHPSENDMAIL, data);
394             if (result==null) return false;
395             if (!result.trim().equals("1")) {
```

```
396                 //fdx Log(TAG, "Retour de "+PHPSENDMAIL+" =>"+result
    +"<");
397                 return false;}
398             return true;
399         }
400
401         private Boolean test204() {
402             try {
403                 URL url = new URL(mSession.getURLPath() + PHP204);
404                 HttpURLConnection conn = (HttpURLConnection) url.
    openConnection();
405                 conn.setConnectTimeout(mSession.getConnectTimeout());
406                 conn.setReadTimeout(mSession.getReadTimeout());
407                 conn.setRequestMethod("HEAD");
408                 InputStream in = conn.getInputStream();
409                 int status = conn.getResponseCode();
410                 in.close();
411                 conn.disconnect();
412                 return (status == HttpURLConnection.HTTP_NO_CONTENT);
413             } catch (Exception e) {
414                 //fdx Log(TAG, "Test 204 : " + e);
415                 return false;
416             }
417         }
418
419     }
420
421 }
```

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

```java
package com.fdx.cookbook;

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

import com.fdx.cookbook.R;

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

```

```java
package com.fdx.cookbook;

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

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

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

```

```java
package com.fdx.cookbook;

import android.graphics.Bitmap;
import android.os.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.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

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() {
        RecipeMailDisplayFragment fragment=new
RecipeMailDisplayFragment();
        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;
    }

    @Override
    public void onResume(){
```

```
50          super.onResume();
51          updateUI();
52      }
53
54      private void updateUI() {
55          CookBook cookbook=CookBook.get(getActivity());
56          List<Recipe> recipes_in=cookbook.getRecipes();
57          List<Recipe> recipes=new ArrayList<>();
58          for(Recipe r:recipes_in){
59              if (r.IsMessage()) recipes.add(r);
60          }
61          if (recipes.isEmpty()){
62              Toast.makeText(getContext(), getString(R.string.P4M0),
    Toast.LENGTH_LONG ).show();
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(
    mRecipe);
 94
 95             mDelete.setOnClickListener(new View.OnClickListener() {
 96                 @Override
 97                 public void onClick(View v) {
 98                     mRecipe.setStatus(StatusRecipe.Deleted);
 99                     CookBook cookbook=CookBook.get(getActivity());
100                     cookbook.updateRecipe(mRecipe);
101                     updateUI();
102                 }
103             });
104             mAdd.setOnClickListener(new View.OnClickListener() {
105                 @Override
106                 public void onClick(View v) {
107                     mRecipe.setStatus(StatusRecipe.Visible);
108                     CookBook cookbook=CookBook.get(getActivity());
109                     cookbook.updateRecipe(mRecipe);
110                     updateUI();
111                 }
112             });
113
114         }
115         public void bind(Recipe recipe){
116             mRecipe=recipe;
117             mTitleTextView.setText(mRecipe.getTitle());
118             mMessage.setText(mRecipe.getMessage());
119             mFrom.setText(mRecipe.getUserFrom().getNameComplete());
120             mPhotoFile=CookBook.get(getActivity()).getPhotoFile(
    mRecipe);
121             if (mPhotoFile==null || !mPhotoFile.exists()){
122                 mPhotoView.setImageDrawable(getResources().
    getDrawable(R.drawable.ic_recipe_see));
123             } else {
124                 Bitmap bitmap=PictureUtils.getScaledBitmap(mPhotoFile
    .getPath(), getActivity());
125                 mPhotoView.setImageBitmap(bitmap);
126             }
127         }
128     @Override
129     public void onClick(View v){    }
130     }
131     // ----------------------------------ADAPTER
    ------------------------
132     private class RecipeAdapter extends RecyclerView.Adapter<
    RecipeMailDisplayFragment.RecipeHolder> {
133         private List<Recipe> mRecipes;
134         public RecipeAdapter(List<Recipe> recipes){
135             mRecipes=recipes;
136         }
137
138         @NonNull
139         @Override
140         public RecipeMailDisplayFragment.RecipeHolder
```

```
onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
            LayoutInflater layoutInflater=LayoutInflater.from(
getActivity());
            return new RecipeMailDisplayFragment.RecipeHolder(
layoutInflater, parent);
        }

        @Override
        public void onBindViewHolder(@NonNull
RecipeMailDisplayFragment.RecipeHolder recipeHolder, int i) {
            Recipe recipe=mRecipes.get(i);
            recipeHolder.bind(recipe);
        }

        @Override
        public int getItemCount() {
            return mRecipes.size();
        }

        public void setRecipes(List<Recipe> recipes){
            mRecipes=recipes;
        }
    }

}
```