```java
1 package com.fdx.cookbook;
2
3 import java.util.Date;
4
5 public class Note {
6     private Integer mNote;
7     private User mUser;
8     private Date mDate;
9
10
11     public Note(Integer note, User u){
12         mNote=note;
13         mUser=u;
14         mDate=new Date();
15     }
16
17     public Note(Integer note, User u, Date d){
18         mNote=note;
19         mUser=u;
20         mDate=d;
21     }
22
23     public Integer getNote() {
24         return mNote;
25     }
26
27     public User getUser() {
28         return mUser;
29     }
30
31     public Date getDate() {
32         return mDate;
33     }
34
35     public void setNote(Integer note) {
36         mNote = note;
37     }
38
39     public void setUser(User user) {
40         mUser = user;
41     }
42
43     public void setDate(Date date) {
44         mDate = date;
45     }
46
47     @Override
48     public boolean equals(Object o){
49         if (o == null) return false;
50         if (o == this) return true;
51         if (getClass() != o.getClass()) return false;
52         Note n=(Note) o;
53         boolean b=(this.mNote.equals(n.getNote()));
54         b=b && (mDate.toString().equals(n.getDate().toString()));
```

```
        b=b && (mUser.equals(n.getUser()));
        return b;
    }
}

```

```
1 package com.fdx.cookbook;
2
3 import java.util.Date;
4 import java.util.UUID;
5
6 public class User {
7     private UUID mId;
8     private String mFamily;
9     private String mName;
10    private Date mDate;
11    private static final String TAG = "DebugUser";
12
13    public User(String family, String name){
14        mFamily=family;
15        mName=name;
16        mId =UUID.randomUUID();
17        mDate=new Date();
18    }
19
20    public User(UUID uuid){
21        mId =uuid;
22        // waiting for user database
23        switch(mId.toString()){
24            case "c81d4e2e-bcf2-11e6-869b-7df92533d2db":
25                mFamily="Devaux_Lion de ML";
26                mName="Fabrice";
27                return;
28            case "c81d4e2e-bcf2-11e7-869b-7df92533d2db":
29                mFamily="Devaux_Lion de ML";
30                mName="Lucile";
31                return;
32            case "c81d4e2e-bcf3-11e6-869b-7df92533d2db":
33                mFamily="Devaux_Lion de ML";
34                mName="Véronique";
35                return;
36            default:
37                mFamily="not found";
38                mName="not found";
39        }
40    }
41
42    public UUID getId() {
43        return mId;
44    }
45
46    public void setId(UUID id) {
47        this.mId = id;
48    }
49
50    public String getFamily() {
51        return mFamily;
52    }
53
54    public void setFamily(String family) {
```

```
        mFamily = family;
    }

    public String getName() {
        return mName;
    }

    public void setName(String name) {
        mName = name;
    }

    public String getNameComplete() {
        return mName+"@"+ mFamily;
    }

    public boolean IsEqual(User r){return (mId.toString().equals(r.getId().toString()));}

    public Date getDate() {
        return mDate;
    }

    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;
        User u=(User) o;
        return u.getId().toString().equals(mId.toString());
    }
    //todo P2 épurer affichage quand champs vides
    //todo P2 tuto swipe page
    //todo P1 messagerie

}

```

```java
1 package com.fdx.cookbook;
2
3 import com.google.gson.Gson;
4 import com.google.gson.reflect.TypeToken;
5
6 import java.lang.reflect.Type;
7 import java.net.MalformedURLException;
8 import java.net.URL;
9 import java.util.ArrayList;
10 import java.util.Calendar;
11 import java.util.Date;
12 import java.util.UUID;
13
14
15 enum StatusRecipe {Submitted,Visible, Deleted }
16 public class Recipe {
17     private UUID mId;
18     private User mOwner;
19     private Date mLastUpdateRecipe;
20     private Date mLastUpdatePhoto;
21     private String mTitle;
22     private String mSource;
23     private URL mSource_url;
24     private int mNbPers;
25     private String[] mSteps;
26     private double mNoteAvg;
27     private ArrayList<Note> mNotes;
28     private ArrayList<Comment> mComments;
29     private RecipeSeason mSeason;
30     private RecipeDifficulty mDifficulty;
31     private String[] mIngredients;
32     private StatusRecipe mStatus;
33     private String mMessage;
34     private User mIdFrom;
35     private Boolean mTS_recipe;
36     private Boolean mTS_photo;
37     private Boolean mTS_comment;
38     private Boolean mTS_note;
39
40     private static final int NBSTEP_MAX=9;
41     private static final int NBING_MAX=15;
42     private static final int NBCOM_MAX=20;
43     private String TAG="CB_Recipe";
44     private String DEFAULT_URL="https://wwww.cookbookfamily.com";
45     private String UUIDNULL="00000000-0000-0000-0000-000000000000";
46
47
48
49     public Recipe() {
50         this(UUID.randomUUID());
51     }
52
53     public Recipe( UUID id){
54         mId=id;
```

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

```
    public String getMessage() {
        return mMessage;
    }

    public void setMessage(String message) {
        mMessage=message;
    }



    // -----------------STEP-------------------
    public void setStep(Integer i, String step) {
        if ((i > 0) && (i <= NBSTEP_MAX)) {
            mSteps[i - 1] = step;
        }
    }

    public String getStep(Integer i){
        if ((i>0)&&(i<=NBSTEP_MAX)) {return mSteps[i-1];}
        else{ return "";}
    }

    public int getNbStep(){
        int j=0;
        for(int i=NBSTEP_MAX; i>0; i--){
            if (!mSteps[i-1].isEmpty()) {j=i; break;}}
        return j;
    }

    public int getNbStepMax(){return NBSTEP_MAX;}

    //--------------Ingredients--------------------
    public void setIngredient(Integer i, String ing) {
        if ((i > 0) && (i <= NBING_MAX)) {
            mIngredients[i-1] = ing;
        }
    }

    public String getIngredient(Integer i){
        if ((i>0)&&(i<=NBING_MAX)) {return mIngredients[i-1];}
        else{ return "";}
    }
    public int getNbIng(){
        int j=0;
        for(int i=NBING_MAX; i>0; i--){
            if (!mIngredients[i-1].isEmpty()) {j=i; break;}}
        return j;
    }

    public int getNbIngMax(){return NBING_MAX;}

    //------------------- Users  -----------------------------------
    public UUID getId() {
        return mId;
```

```
    }

    public User getUserFrom() {
        return mIdFrom;
    }

    public void setUserFrom(User idFrom) {
        mIdFrom = idFrom;
    }

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

    public User getOwner() {
        return mOwner;
    }

    public String getOwnerIdString() {
        return mOwner.getId().toString();
    }

    public void setOwner(User owner) {
        mOwner = owner;
    }

    //------------------- Source ----------------------------------
    public String getSource() {
        if (mSource==null) return "";
        return mSource;
    }
    public void setSource(String source) {
        mSource = source;
    }
    public URL getSource_url() {
        return mSource_url;
    }

    public String getSource_url_name() {
        String ret=mSource_url.toString();
        if (ret.equals("https:") || ret.equals("http:")){ret="";}
        return ret;
    }

    public void setSource_url(URL source_url) {
        mSource_url = source_url;
    }

    // ------------------------Enum -------------------------m

    public RecipeSeason getSeason() {
        return mSeason;
    }

    public void setSeason(RecipeSeason season) {
        mSeason = season;
```

```
    }

    public boolean IsSummer(){
        switch (mSeason) {
            case SUMMER:
                return true;
            case ALLYEAR:
                return true;
            default:
                return false;
        }
    }
    public boolean IsWinter(){
        switch (mSeason) {
            case WINTER:
                return true;
            case ALLYEAR:
                return true;
            default:
                return false;
        }
    }

    public RecipeDifficulty getDifficulty() {
        return mDifficulty;
    }

    public void setDifficulty(RecipeDifficulty difficulty) {
        mDifficulty = difficulty;
    }

    public StatusRecipe getStatus() {return mStatus;}
    public void setStatus(StatusRecipe s) {
        mStatus = s;
    }
    public boolean IsVisible(){
        if (mStatus==StatusRecipe.Visible) {return true;}
        return false;
    }
    public boolean IsMessage(){
        if (mStatus==StatusRecipe.Submitted) {return true;}
        return false;
    }
    public boolean IsMarkedDeleted(){
        if (mStatus==StatusRecipe.Deleted) {return true;}
        return false;
    }

    public void updateTS(AsynCallFlag asyn, Boolean b){
        switch(asyn){
            case NEWRECIPE:{mTS_recipe=b;return;}
            case NEWPHOTO:{mTS_photo=b;return;}
            case NEWRATING:{mTS_note=b;return;}
            case NEWCOMMENT:{mTS_comment=b;return;}
```

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

```
    public double getNoteAvg() {
        mNoteAvg=0;
        if (mNotes==null) return mNoteAvg;
        if (mNotes.size()!=0) {
            for(Note n:mNotes){mNoteAvg+=n.getNote();}
            mNoteAvg=mNoteAvg/mNotes.size();
        } else {mNoteAvg=0;}
        return mNoteAvg;
    }

    //--------------------- Serialisation ------------------------
    public String getSerializedComments(){
        Gson gson = new Gson();
        return gson.toJson(mComments);
    }

    public void getCommentsDeserialised(String raw){
        Gson gson=new Gson();
        Type listOfNotesObject = new TypeToken<ArrayList<Comment>>() {}.getType();
        mComments=gson.fromJson(raw, listOfNotesObject);
    }
    public String getSerializedOwner(){
        Gson gson = new Gson();
        return gson.toJson(mOwner);
    }
    public String getSerializedFrom(){
        Gson gson = new Gson();
        return gson.toJson(mIdFrom);
    }
    public void getOwnerDeserialized(String raw){
        Gson gson=new Gson();
        mOwner=gson.fromJson(raw, User.class);
    }
    public void getFromDeserialized(String raw){
        Gson gson=new Gson();
        mIdFrom=gson.fromJson(raw, User.class);
    }

    public String getSerializedNotes(){
        Gson gson = new Gson();
        return gson.toJson(mNotes);
    }

    public void getNotesDeserialised(String raw){
        Gson gson=new Gson();
        Type listOfNotesObject = new TypeToken<ArrayList<Note>>() {}.getType();
        mNotes=gson.fromJson(raw, listOfNotesObject);
    }

    // ******************* tests *********************
    public Boolean hasNotChangedSince(Recipe r){
        if (!mId.toString().equals(r.getId().toString())) return
```

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

```java
1 package com.fdx.cookbook;
2
3 import java.text.DateFormat;
4 import java.text.SimpleDateFormat;
5 import java.util.Date;
6
7 public class Comment {
8     private String mTxt;
9     private User mUser;
10    private Date mDate;
11
12    public Comment(){
13        mTxt="";
14        mUser=new User("","");
15        mDate=new Date();
16    }
17
18    public Comment(String s, User u){
19        mTxt=s;
20        mUser=u;
21        mDate=new Date();
22    }
23    public Comment(String s, User u, Date d){
24        mTxt=s;
25        mUser=u;
26        mDate=d;
27    }
28
29    public String getTxt() {
30        return mTxt;
31    }
32
33    public User getUser() {
34        return mUser;
35    }
36
37    public Date getDate() {
38        return mDate;
39    }
40
41    public void setDate(Date date) {
42        mDate = date;
43    }
44
45    public String toTxt(){
46        DateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy");
47        String s= dateFormat.format(mDate);
48        return mTxt+" ("+mUser.getNameComplete()+") - " + s;
49    }
50
51    @Override
52    public boolean equals(Object o){
53        if (o == null) return false;
54        if (o == this) return true;
```

```
55          if (getClass() != o.getClass()) return false;
56          Comment c=(Comment) o;
57          boolean b=(this.mTxt.equals(c.getTxt()));
58          b=b && (mDate.toString().equals(c.getDate().toString()));
59          b=b && (mUser.equals(c.getUser()));
60          return b;
61      }
62
63      public String convert(){
64          String s1=">"+mTxt+"< (";
65          s1=s1 + mDate.toString()+") de ";
66          s1=s1+mUser.getNameComplete();
67          return s1;
68      }
69  }
70
```

```java
package com.fdx.cookbook;

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


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());                          //last modification
        ContentValues values=getContentValues(r);
        mDatabase.update(RecipeDbSchema.RecipeTable.NAME, values,
                RecipeDbSchema.RecipeTable.Cols.UUID+" =?",
                new String[] {uuidString});
    }
```

```java

    private RecipeCursorWrapper queryRecipes(String whereClause, String[] whereArgs){
        Cursor cursor=mDatabase.query(
                RecipeDbSchema.RecipeTable.NAME,
                null, // selct all columns
                whereClause,
                whereArgs,
                null, null, null
        );
        return new RecipeCursorWrapper(cursor);
    }

    public void addRecipe(Recipe r){
        ContentValues values=getContentValues(r);
        mDatabase.insert(RecipeDbSchema.RecipeTable.NAME, null, values);
    }

    public void removeRecipe(Recipe r){
        String uuidString=r.getId().toString();
        mDatabase.delete(RecipeDbSchema.RecipeTable.NAME,
                RecipeDbSchema.RecipeTable.Cols.UUID+" =?",
                new String[] {uuidString});
        return;
    }

    public void markRecipeToDelete(Recipe r){
        r.setStatus(StatusRecipe.Deleted);
        updateRecipe(r);
        return;
    }

    public List<Recipe> getRecipes(){
          List<Recipe> recipes=new ArrayList<>();
          RecipeCursorWrapper cursor=queryRecipes(null, null);
          try{
              cursor.moveToFirst();
              while (!cursor.isAfterLast()){
                  recipes.add(cursor.getRecipe());
                  cursor.moveToNext();
              }
          } finally { cursor.close();}
          return recipes;
    }

    private static ContentValues getContentValues(Recipe recipe) {
        ContentValues values=new ContentValues();
        values.put(RecipeDbSchema.RecipeTable.Cols.UUID, recipe.getId().toString());
        values.put(RecipeDbSchema.RecipeTable.Cols.OWNER, recipe.getSerializedOwner());
        values.put(RecipeDbSchema.RecipeTable.Cols.TITLE, recipe.getTitle());
```

```
        values.put(RecipeDbSchema.RecipeTable.Cols.SOURCE, recipe.getSource());
        values.put(RecipeDbSchema.RecipeTable.Cols.SOURCE_URL, recipe.getSource_url().toString());
        values.put(RecipeDbSchema.RecipeTable.Cols.DATE, recipe.getDate().getTime());
        if (recipe.getDatePhoto()!=null){
        values.put(RecipeDbSchema.RecipeTable.Cols.DATE_PHOTO, recipe.getDatePhoto().getTime());}
        values.put(RecipeDbSchema.RecipeTable.Cols.NBPERS, recipe.getNbPers());
        for(int i=0;i<recipe.getNbStepMax();i++){
            values.put(RecipeDbSchema.RecipeTable.Cols.STEP[i], recipe.getStep(i+1));
        }
        for(int i=0;i<recipe.getNbIngMax();i++){
            values.put(RecipeDbSchema.RecipeTable.Cols.ING[i], recipe.getIngredient(i+1));
        }
        values.put(RecipeDbSchema.RecipeTable.Cols.SEASON, recipe.getSeason().name());
        values.put(RecipeDbSchema.RecipeTable.Cols.DIFFICULTY, recipe.getDifficulty().name());
        values.put(RecipeDbSchema.RecipeTable.Cols.COMMENTS, recipe.getSerializedComments());
        values.put(RecipeDbSchema.RecipeTable.Cols.STATUS, recipe.getStatus().toString());
        values.put(RecipeDbSchema.RecipeTable.Cols.NOTES, recipe.getSerializedNotes());
        values.put(RecipeDbSchema.RecipeTable.Cols.MESSAGE, recipe.getMessage());
        values.put(RecipeDbSchema.RecipeTable.Cols.MESSAGE_FROM, recipe.getSerializedFrom());
        values.put(RecipeDbSchema.RecipeTable.Cols.TS_RECIPE, recipe.getTS(AsynCallFlag.NEWRECIPE));
        values.put(RecipeDbSchema.RecipeTable.Cols.TS_PHOTO, recipe.getTS(AsynCallFlag.NEWPHOTO));
        values.put(RecipeDbSchema.RecipeTable.Cols.TS_COMMENT, recipe.getTS(AsynCallFlag.NEWCOMMENT));
        values.put(RecipeDbSchema.RecipeTable.Cols.TS_NOTE, recipe.getTS(AsynCallFlag.NEWRATING));
        return values;
    }

    public File getPhotoFile(Recipe r){
        if (r==null) {return null;}
        File filesDir= mContext.getFilesDir();
        return new File(filesDir, r.getPhotoFilename());
    }

    public Boolean deleteImage(Recipe r){
        Boolean success=false;
        File filesDir= mContext.getFilesDir();
        File im=new File(filesDir, r.getPhotoFilename());
```

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

```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 UserFiltered;
11    private User UserState;
12    private boolean IsNoteSorted;
13
14    public ListMask(User u){
15        SeasonFiltered=false;
16        DifficultyFiltered=false;
17        UserFiltered=false;
18        IsNoteSorted=false;
19        UserState=u;
20        SeasonState=RecipeSeason.ALLYEAR;
21        DifficultyState=RecipeDifficulty.UNDEFINED;
22    }
23
24    public void reset(){
25        SeasonFiltered=false;
26        DifficultyFiltered=false;
27        UserFiltered=false;
28        IsNoteSorted=false;
29        SeasonState=RecipeSeason.ALLYEAR;
30        DifficultyState=RecipeDifficulty.UNDEFINED;
31    }
32
33    public String display(){
34        // for debug
35        String s="";
36        if (SeasonFiltered){
37            s=s+SeasonState.toString()+" ";
38        }else{
39            s=s+"No Season ";
40        }
41        if (DifficultyFiltered){
42            s=s+DifficultyState.toString()+" ";
43        } else {
44            s=s+"No dif ";
45        }
46
47        if (UserFiltered){
48            s=s+"Name "+UserState.getName()+" ";
49        } else {
50                s=s+"No User ";
51        }
52
53        if (IsNoteSorted){
54            s=s+" SortN ";
```

```
        } else {
            s=s+"No SortN ";
        }
        return s;
    }

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

    public Integer toggleWinter(){
        Integer i;
        if (SeasonFiltered){
            SeasonFiltered=false;
            SeasonState=RecipeSeason.ALLYEAR;
            i=R.string.TWOFF;
        } else {
            SeasonFiltered=true;
            SeasonState=RecipeSeason.WINTER;
            i=R.string.TWON;
        }
        return i;
    }

    public Integer toggleUser(User u){
        Integer i;
        if (UserFiltered){
            UserFiltered=false;
            i=R.string.TROFF;
        }else{
            UserState=u;
            UserFiltered=true;
            i=R.string.TRON;
        }
        return i;
    }

    public Integer toggleDifficulty(RecipeDifficulty rd){
        Integer i;
        if (DifficultyFiltered){
            DifficultyFiltered=false;
            DifficultyState=RecipeDifficulty.UNDEFINED;
            i=R.string.TDOFF;
        }else{
```

```
            DifficultyFiltered=true;
            DifficultyState=rd;
            i=R.string.TDON;
        }
        return i;
    }

    public Integer toggleSortNote(){
        Integer i;
        if (IsNoteSorted){
            IsNoteSorted=false;
            i=R.string.TNOFF;
        }else{
            IsNoteSorted=true;
            i=R.string.TNON;
        }
        return i;
    }

    public boolean isNoteSorted(){return IsNoteSorted;}

    public boolean isRecipeSelected(Recipe r){
        boolean b=false;
        if (!r.IsVisible()) return false;
        if (SeasonFiltered) {
            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 (UserFiltered){
            if (!r.getOwner().IsEqual(UserState)) return false;
        }
        return true;
    }

    public String convertToString(){
        String s,sep=";";
        s=(SeasonFiltered ? "Y":"N")+sep;
        s+=SeasonState.toString()+sep;
        s+=(DifficultyFiltered ? "Y":"N")+sep;
        s+=DifficultyState.toString()+sep;
        s+=(IsNoteSorted ? "Y":"N")+sep;
        s+=(UserFiltered ? "Y":"N")+sep;
        s+=UserState.getFamily()+sep;
        s+=UserState.getName()+sep;
        s+=UserState.getId().toString();
        return s;
    }

```

```
    public void updateFromString(String s){
        String sep=";";
        String[] tokens = s.split(sep);
        if (tokens.length<9) 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]);
        IsNoteSorted=( tokens[4].equals("Y")? true:false);
        UserFiltered=( tokens[5].equals("Y")? true:false);
        UserState=new User(tokens[6], tokens[7]);
        UserState.setId(UUID.fromString(tokens[8]));
    }
}
```

```
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 // to invoke :
10 //      public SessionInfo mSession;
11 //      mSession= SessionInfo.get(getApplicationContext());
12 //           or   SessionInfo.get(getActivity());
13 //**************************************************************
14 public class SessionInfo {
15     private static SessionInfo ourInstance;
16     private User mUser;
17     private Context mContext;
18     private Boolean mIsConnected;
19     private Boolean mReqNewSession;
20     private String CB_FAMILY="family";
21     private String CB_NAME="name";
22     private String CB_ID="iduser";
23     private String NOT_FOUND="Not found";
24     private static final String TAG = "DebugSessionInfo";
25     private static String URLPATH="http://82.66.37.73:8085/cb/";
26     public static int CONNECT_TIMEOUT = 7000;
27     public static int READ_TIMEOUT = 7000;
28
29     public static SessionInfo get(Context context) {
30         if (ourInstance==null){
31             ourInstance= new SessionInfo(context);
32         }
33         return ourInstance;
34     }
35
36     private SessionInfo(Context context) {
37         mContext=context.getApplicationContext();
38         mIsConnected=false;
39         mReqNewSession=false;
40         String sharedPref;
41         sharedPref= PreferenceManager.getDefaultSharedPreferences(
   context)
42                 .getString(CB_ID, NOT_FOUND);
43         if (sharedPref.equals(NOT_FOUND)){
44             mUser = new User(NOT_FOUND, NOT_FOUND);
45         } else {
46             mUser = new User(PreferenceManager.
   getDefaultSharedPreferences(context)
47                     .getString(CB_FAMILY, NOT_FOUND),
   PreferenceManager.getDefaultSharedPreferences(context)
48                     .getString(CB_NAME, NOT_FOUND));
49             mUser.setId(UUID.fromString(sharedPref));
50         }
51     }
```

```
52
53
54      public User getUser() {
55          return mUser;
56      }
57
58      public void setStoredUser(User user){
59          PreferenceManager.getDefaultSharedPreferences(mContext)
60                  .edit()
61                  .putString(CB_FAMILY, user.getFamily())
62                  .putString(CB_NAME, user.getName())
63                  .putString(CB_ID, user.getId().toString())
64                  .apply();
65          mUser=user;
66      }
67      public void clearStoredUser(){
68          SharedPreferences settings = mContext.getSharedPreferences("
    PreferencesName", Context.MODE_PRIVATE);
69          settings.edit().remove("CB_FAMILY").commit();
70          settings.edit().remove("CB_NAME").commit();
71          settings.edit().remove("CB_ID").commit();
72
73      }
74
75      public Boolean IsEmpty(){
76          if (mUser.getName().equals(NOT_FOUND)){return true;}
77          return false;
78      }
79
80      public Boolean IsReqNewSession() {
81          return mReqNewSession;
82      }
83
84      public void setReqNewSession(Boolean reqNewSession) {
85          mReqNewSession = reqNewSession;
86      }
87
88      public void setConnection(Boolean b){
89          mIsConnected=b;
90      }
91      public Boolean IsConnected(){
92          return mIsConnected;
93      }
94
95      public Context getContext(){return mContext;}
96
97      public String getURLPath(){return URLPATH;}
98      public int getConnectTimeout(){return CONNECT_TIMEOUT;}
99      public int getReadTimeout(){return READ_TIMEOUT;}
100
101 }
102
```

```java
package com.fdx.cookbook;

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

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

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

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

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

    /*public String sendGetRequest(String uri) {
        try {
            URL url = new URL(uri);
            HttpURLConnection con = (HttpURLConnection) url.openConnection();
            BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(con.getInputStream()));
            String result;
            StringBuilder sb = new StringBuilder();
            while((result = bufferedReader.readLine())!=null){
                sb.append(result);
            }
            return sb.toString();
        } catch (Exception e) {
            return null;
```

```
        }
    }

    public String sendPostRequest(String requestURL,
                                  HashMap<String, String> postDataParams) {
        URL url;
        String response = "";
        try {
            url = new URL(requestURL);

            HttpURLConnection conn = (HttpURLConnection) url.openConnection();
            conn.setReadTimeout(mSession.getReadTimeout());
            conn.setConnectTimeout(mSession.getConnectTimeout());
            conn.setRequestMethod("POST");
            conn.setDoInput(true);
            conn.setDoOutput(true);
            OutputStream os = conn.getOutputStream();
            BufferedWriter writer = new BufferedWriter(
                    new OutputStreamWriter(os, "UTF-8"));
            writer.write(getPostDataString(postDataParams));
            writer.flush();
            writer.close();
            os.close();
            int responseCode = conn.getResponseCode();
            if (responseCode == HttpURLConnection.HTTP_OK) {
                BufferedReader br = new BufferedReader(new InputStreamReader(conn.getInputStream()));
                response = br.readLine();
            } else {
                response = "Error Registering";
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return response;
    }
    */

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

        URL url;
        try {
            url = new URL(requestURL);
            HttpURLConnection conn = (HttpURLConnection) url.openConnection();
            conn.setReadTimeout(mSession.getReadTimeout());
            conn.setConnectTimeout(mSession.getConnectTimeout());
            conn.setRequestMethod("POST");
            conn.setDoInput(true);
            conn.setDoOutput(true);
```

```
            OutputStream os = conn.getOutputStream();
            BufferedWriter writer = new BufferedWriter(
                    new OutputStreamWriter(os, "UTF-8"));
            writer.write(this.getPostDataString(postDataParams));
            writer.flush();
            writer.close();
            os.close();
            StringBuilder sb = new StringBuilder();
            BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(conn.getInputStream()));
            String json;
            while ((json = bufferedReader.readLine()) != null) {
                sb.append(json + "\n");
            }
            return sb.toString().trim();
        } catch (Exception e) {
            Log.d(TAG, ">"+e);
            return null;
        }
    }
    private String getPostDataString(HashMap<String, String> params) {
        StringBuilder result = new StringBuilder();
        boolean first = true;
        for (Map.Entry<String, String> entry : params.entrySet()) {
            if (first)
                first = false;
            else
                result.append("&");
            try {result.append(URLEncoder.encode(entry.getKey(), "UTF-8"));
            result.append("=");
            result.append(URLEncoder.encode(entry.getValue(), "UTF-8"));
            } catch (Exception e){
                //fdx Log.d(TAG, "Pb with >"+entry.getKey()+":"+entry.getValue()+"<");
            }
        }
        return result.toString();
    }
    public List<Comment> parseCommentsOfRecipe(String json){
        Comment c;
        List<Comment> cs=new ArrayList<>();
        if ((json==null)||(json.equals(""))) return null;
        try {
            JSONArray jarr1=new JSONArray(json);
            for (int i=0; i<jarr1.length(); i++){
                JSONObject obj = jarr1.getJSONObject(i);
                c=parseObjectComment(obj);
                if (c!=null) cs.add(c);
            }
        } catch (Exception e){
            //fdx Log.d(TAG, "Failure in parsing JSON Array Comments
```

```
"+e);
            return null;
        }
        return cs;
    }

    public Comment parseObjectComment(JSONObject obj){
        try {
        UUID uuid=UUID.fromString(obj.getString("id_user"));
        User u=new User(obj.getString("family"), obj.getString("name") );
        u.setId(uuid);
        String s=obj.getString("date_comment");
        Date date=new SimpleDateFormat(MYSQLDATEFORMAT).parse(s);
        return new Comment(obj.getString("comment"),u,date);}
        catch (Exception e){
            //fdx Log.d(TAG, "Failure in parsing JSONObject Comments "+e);
            return null;
        }
    }
    public List<Note> parseNotesOfRecipe(String json){
        Note n;
        List<Note> ns=new ArrayList<>();
        if ((json==null)||(json.equals(""))) return null;
        try {
            JSONArray jarr1=new JSONArray(json);
            for (int i=0; i<jarr1.length(); i++){
                JSONObject obj = jarr1.getJSONObject(i);
                n=parseObjectNote(obj);
                if (n!=null) ns.add(n);
            }
        } catch (Exception e){
            //fdx Log.d(TAG, "Failure in parsing JSON Array Comments "+e);
            return null;
        }
        return ns;
    }

    public Note parseObjectNote(JSONObject obj){
        try {
            // uuid=UUID.fromString(obj.getString("id_recipe"));
            UUID uuid=UUID.fromString(obj.getString("id_user"));
            User u=new User(obj.getString("family"), obj.getString("name") );
            u.setId(uuid);
            String s=obj.getString("date_note");
            Date date=new SimpleDateFormat(MYSQLDATEFORMAT).parse(s);
            return new Note(obj.getInt("note"),u,date);}
        catch (Exception e){
            //fdx Log.d(TAG, "Failure in parsing JSONObject Note : "+e);
            return null;
```

```
        }
    }
    public Recipe parseObjectRecipeStamp(JSONObject obj){
        try {
            //---------------- uuid and users
            UUID uuid = UUID.fromString(obj.getString("id_recipe"));
            Recipe r = new Recipe(uuid);
            uuid = UUID.fromString(obj.getString("id_owner"));
            User u = new User(uuid);
            r.setOwner(u);
            //dates
            Date date;
            String s1 = obj.getString("lastupdate_recipe");
            if (s1==null) return null;
            if ((!s1.equals("null"))&&(s1.length()>5)) {
                date = new SimpleDateFormat(MYSQLDATEFORMAT).parse(s1);
                r.setDate(date);
            } else {return null;}
            s1 = obj.getString("lastupdate_photo");
            if ((!s1.equals("null"))&&(s1.length()>5)) {
                date = new SimpleDateFormat(MYSQLDATEFORMAT).parse(s1);
                r.setDatePhoto(date);
            }
            s1=obj.getString("message");
            if ((s1==null)||(s1.equals("null"))) s1="";
            r.setMessage(s1);
            r.setStatus(StatusRecipe.valueOf(obj.getString("status"));
            s1 = obj.getString("id_from");
            if ((!s1.equals("null"))&&(s1.length()>5)) {
                uuid = UUID.fromString(s1);
                u = new User(uuid);
                r.setOwner(u);
            }
            return r;
        }
        catch (Exception e){
            //fdx Log.d(TAG, "Failure in parsing JSONObject Recette Stamp : "+e);
            return null;
        }
    }

    public List<Recipe> parseStampsTiers(String json){
        Recipe r;
        List<Recipe> rs=new ArrayList<>();
        if ((json==null)||(json.equals(""))) return null;
        try {
            JSONArray jarr1=new JSONArray(json);
            for (int i=0; i<jarr1.length(); i++){
                JSONObject obj = jarr1.getJSONObject(i);
                r=parseObjectRecipeStamp(obj);
```

```
                if (r!=null) rs.add(r);
            }
        } catch (Exception e){
            //fdx Log.d(TAG, "Failure in parsing JSON Array Comments "+e);
            return null;
        }
        return rs;
    }

    public boolean parseObjectRecipe(Recipe r, JSONObject obj, boolean withphoto, boolean full){
        try {
            UUID uuid;
            String s1, s2;
            User u;
            if (full) {
                uuid = UUID.fromString(obj.getString("id_owner"));
                s1 = obj.getString("owner_family");
                s2 = obj.getString("owner_name");
                u = new User(s1, s2);
                u.setId(uuid);
                r.setOwner(u);
                s1=obj.getString("message");
                if ((s1==null)||(s1.equals("null"))) s1="";
                r.setMessage(s1);
                r.setStatus(StatusRecipe.valueOf(obj.getString("status")));
                s1 = obj.getString("id_from");
                if ((!s1.equals("null"))&&(s1.length()>5)&&(s1!=null)) {
                    uuid = UUID.fromString(s1);
                    s1 = obj.getString("from_family");
                    s2 = obj.getString("from_name");
                    u = new User(s1, s2);
                    u.setId(uuid);
                    r.setUserFrom(u);
                }
            }
            //------------ dates
            Date date;
            s1 = obj.getString("lastupdate_recipe");
            if ((!s1.equals("null"))&&(s1.length()>5)&&(s1!=null)) {
                date = new SimpleDateFormat(MYSQLDATEFORMAT).parse(s1);
                r.setDate(date);
            } else {return false;}
            s1 = obj.getString("lastupdate_photo");
            if ((!s1.equals("null"))&&(s1.length()>5)&&(s1!=null)) {
                date = new SimpleDateFormat(MYSQLDATEFORMAT).parse(s1);
                r.setDatePhoto(date);
            }
            //---------------- titre, source x2, nbpers
```

```
            s1 = obj.getString("title");
            URL url;
            if (s1==null) return false;
            if (s1.equals("null")) return false;
            r.setTitle(s1);
            r.setSource(obj.getString("source"));
            s1 = obj.getString("source_url");
            if (!s1.equals("null")&&(s1.length()>5)&&(s1!=null)) {
                try {
                    url = new URL(s1);
                    r.setSource_url(url);
                } catch (MalformedURLException e) {
                    //fdx Log.d(TAG, "getURL from cookbook download failed in parse recipe");
                }
            }
            int nbp=obj.getInt("nb_pers");
            if (nbp==0) return false;
            r.setNbPers(nbp);
            //---------------- Etapes and ing
            DecimalFormat formatter = new DecimalFormat("00");
            for (int i = 0; i < r.getNbStepMax(); i++) {
                s1 = "etape" + formatter.format(i + 1);
                s2=obj.getString(s1);
                if (!s2.equals("null")) {
                    r.setStep(i + 1, s2);}
            }
            for (int i = 0; i < r.getNbIngMax(); i++) {
                s1 = "ing" + formatter.format(i + 1);
                s2=obj.getString(s1);
                if (!s2.equals("null")) {
                    r.setIngredient(i + 1, s2);}
            }
            //----------------- enum
            r.setSeason(RecipeSeason.valueOf(obj.getString("season"));
            r.setDifficulty(RecipeDifficulty.valueOf(obj.getString("difficulty")));

            // --------------- photo
            CookBook cb = CookBook.get(mSession.getContext());
            File f=cb.getPhotoFile(r);
            if (withphoto) {
                s1 = obj.getString("photo");
                if ((!s1.equals("null"))&&(!s1.equals(""))&&(s1!=null)) {
                    Bitmap bm=PictureUtils.getBitmapFromString(s1);
                    if (f.exists()){
                        //fdx Log.d(TAG, "file  " + f.toString()+" ecrasée dans parsing recipe !");
                        f.delete();
                    }
                    try {
                        if(!f.createNewFile()) {
```

```
                            //fdx Log.d(TAG, "Error in creating file "+f.toString());
                        }
                    } catch (IOException e) {
                        //fdx Log.d(TAG, "Error in creating file "+f.toString()+":" +e);
                    }
                    try {
                        FileOutputStream out = new FileOutputStream(f);
                        bm.compress(Bitmap.CompressFormat.JPEG, 100, out);
                        out.flush();
                        out.close();
                    } catch (Exception e) {
                        //fdx Log.d(TAG, "Storing bitmap error  " + e);
                    }

                }
            }

            return true;
        }
        catch (Exception e){
            //fdx Log.d(TAG, "Failure in parsing JSONObject Recette : "+e);
            return false;
        }
    }

}

```

```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  }
67
```

```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;
    }

    public static RecipeSeason NextSeason(RecipeSeason r){
        int i=getIndex(r);
        i=i+1;
        if (i==list.length) i=0;
        return getSeason(i);
    }
}

```

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

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

```
                //fdx Log.d(TAG, "Notes de " + r.getTitle()+" updaté");
                isChanged=true;
                r.updateTS(AsynCallFlag.NEWRATING,false);
                mCookbook.updateRecipe(r);
            }
            if (r.IsMarkedDeleted()){
                if (deleteRecipeFromCB(r)) {
                    //fdx Log.d(TAG, "Recette effacée : " + r.getTitle());
                    mCookbook.removeRecipe(r);
                    mCookbook.deleteImage(r);
                    isChanged=true;
                } else{
                    //fdx Log.d(TAG, "Recette " + r.getTitle()+" non effacée");
                }
            }
        }
        return isChanged;
    }

    @Override
    protected void onPostExecute(Boolean isChanged) {
        super.onPostExecute(isChanged);
        //fdx Log.d(TAG, "Has changed ? " + isChanged);
        if (isChanged)
        Toast.makeText(mContext, mContext.getString(R.string.P1_sync), Toast.LENGTH_SHORT).show();
    }

    private Boolean test204() {
        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();
            return (status == HttpURLConnection.HTTP_NO_CONTENT);
        } catch (Exception e) {
            //fdx Log.d(TAG, "Test 204 : " + e);
            return false;
        }
    }

    private Boolean uploadRecipe(Recipe r) {
        HashMap<String, String> data = new HashMap<>();
        String s1;
        DateFormat dateFormat = new SimpleDateFormat(MYSQLDATEFORMAT)
```

```
141 ;
142         data.put("idrecipe", r.getId().toString());
143         data.put("iduser", r.getOwner().getId().toString());
144         data.put("title", r.getTitle());
145         data.put("source", r.getSource());
146         data.put("sourceurl", r.getSource_url_name());
147         DecimalFormat formatter = new DecimalFormat("00");
148         data.put("nbpers", formatter.format(r.getNbPers()));
149         s1 = dateFormat.format(r.getDate());
150         data.put("date", s1);
151         for (int i = 0; i < r.getNbStepMax(); i++) {
152             s1 = "etape" + formatter.format(i + 1);
153             data.put(s1, r.getStep(i + 1));
154         }
155         for (int i = 0; i < r.getNbIngMax(); i++) {
156             s1 = "ing" + formatter.format(i + 1);
157             data.put(s1, r.getIngredient(i + 1));
158         }
159         data.put("season", r.getSeason().toString());
160         data.put("difficulty", r.getDifficulty().toString());
161         String result = mNetUtils.sendPostRequestJson(mSession.
    getURLPath() + PHPUPDATECREATE, data);
162         if (result==null) return false;
163         if (!result.trim().equals("1")) {
164             //fdx Log.d(TAG, "Retour de "+PHPUPDATECREATE+" =>"+
    result+"<");
165         return false;}
166         return true;
167     }
168
169     private Boolean uploadPhoto(Recipe r) {
170         String s1;
171         File file = mCookbook.getPhotoFile(r);
172         Bitmap bitmap = PictureUtils.getBitmap(file.getPath());
173         if (bitmap == null) {
174             //fdx Log.d(TAG, "Pas de bitmap pour " + r.getTitle());
175             return false;
176         }
177         String uploadImage = PictureUtils.getStringImage(bitmap);
178         HashMap<String, String> data = new HashMap<>();
179         data.put("idrecipe", r.getId().toString());
180         if (uploadImage==null) return false;
181         data.put("image", uploadImage);
182         DateFormat dateFormat = new SimpleDateFormat(MYSQLDATEFORMAT)
    ;
183         s1 = dateFormat.format(r.getDatePhoto());
184         data.put("date", s1);
185         String result = mNetUtils.sendPostRequestJson(mSession.
    getURLPath() + PHPUPUPLOADPHOTO, data);
186         if (result==null) return false;
187         if (!result.trim().equals("1")) {
188             //fdx Log.d(TAG, "Retour de "+ PHPUPUPLOADPHOTO+" = "+
    result);
189             return false;}
```

```
        return true;
    }
    private Boolean deleteRecipeFromCB(Recipe r) {
        HashMap<String, String> data = new HashMap<>();
        data.put("idrecipe", r.getId().toString());
        data.put("iduser", mSession.getUser().getId().toString());
        String result = mNetUtils.sendPostRequestJson(mSession.getURLPath() + PHPDELETERECIPE, data);
        if (result==null) return false;
        if (!result.trim().equals("1")) {
            //fdx Log.d(TAG, "Retour de "+ PHPDELETERECIPE+" = "+result);
            return false;}
        return true;
    }

    private Boolean syncCommentsRecipe(Recipe r) {
        boolean ret=false;
        HashMap<String, String> data = new HashMap<>();
        data.put("idrecipe", r.getId().toString());
        String result = mNetUtils.sendPostRequestJson(mSession.getURLPath() + PHPGETCOMMENTSOFRECIPE, data);
        List<Comment> downloadedComments=mNetUtils.parseCommentsOfRecipe(result);
        if (downloadedComments==null) downloadedComments=new ArrayList<>();
        List<Comment> recipeComments=r.getComments();
        if (recipeComments==null) recipeComments=new ArrayList<>();
        //----------- boucle local
        List<Comment> cloc= new ArrayList<>();
        Comment c,ci;
        if (!recipeComments.isEmpty()){
            for (int i = 0; i < recipeComments.size(); i++){
                ci=recipeComments.get(i);
                c=new Comment(ci.getTxt(),ci.getUser(),ci.getDate());
                if (downloadedComments.isEmpty()) {cloc.add(c);} else {
                if (downloadedComments.indexOf(c)==-1) cloc.add(c);}
            }
        }
        //----------- boucle serveur
        List<Comment> cser= new ArrayList<>();
        if (!downloadedComments.isEmpty()){
            for (int i = 0; i < downloadedComments.size(); i++){
                ci=downloadedComments.get(i);
                c=new Comment(ci.getTxt(),ci.getUser(),ci.getDate());
                if (recipeComments.isEmpty()) {cser.add(c);} else {
                if (recipeComments.indexOf(c)==-1) cser.add(c);}
            }
        }
        if (!cser.isEmpty()){
            for (int i = 0; i < cser.size(); i++){
                r.addComment(cser.get(i));
            }
```

```
            if (cser.size()>0) {
                mCookbook.updateRecipe(r);
                ret=true;
            }
        }
        if (!cloc.isEmpty()) {
            if (!uploadComments(r,cloc)) {ret=false;}
            else {ret=true;}
        }
        return ret;
    }

    private Boolean uploadComments(Recipe r, List<Comment> cs){
        if (cs==null) return false;
        if (cs.size()==0) return true;
        HashMap<String, String> data = new HashMap<>();
        DateFormat dateFormat = new SimpleDateFormat(MYSQLDATEFORMAT);
        String s1;
        boolean b=true;
        for (Comment c:cs) {
            data.clear();
            data.put("idrecipe", r.getId().toString());
            data.put("idfrom", c.getUser().getId().toString());
            data.put("comment", c.getTxt());
            s1 = dateFormat.format(c.getDate());
            data.put("date",s1  );
            String result = mNetUtils.sendPostRequestJson(mSession.getURLPath() + PHPADDCOMMENTTORECIPE, data);
            if (result==null) b=false;
            if (!result.equals("1")) {
                //fdx Log.d(TAG, "Retour de "+ PHPADDCOMMENTTORECIPE+" = "+result);
                b=false;}
        }
        return b;
    }
    private Boolean syncNotesRecipe(Recipe r) {
        boolean ret=false;
        HashMap<String, String> data = new HashMap<>();
        data.put("idrecipe", r.getId().toString());
        String result = mNetUtils.sendPostRequestJson(mSession.getURLPath() + PHPGETNOTESOFRECIPE, data);
        List<Note> downloadedNotes=mNetUtils.parseNotesOfRecipe(result);
        if (downloadedNotes==null) downloadedNotes=new ArrayList<>();
        List<Note> recipeNotes=r.getNotes();
        if (recipeNotes==null) recipeNotes=new ArrayList<>();
        //----------- boucle local
        List<Note> cloc= new ArrayList<>();
        Note c,ci;
        if (!recipeNotes.isEmpty()){
            for (int i = 0; i < recipeNotes.size(); i++){
                ci=recipeNotes.get(i);
```

```
                c=new Note(ci.getNote(),ci.getUser(),ci.getDate());
                if (downloadedNotes.isEmpty()) {
                    cloc.add(c);}
                else {
                    if (downloadedNotes.indexOf(c)==-1) cloc.add(c);}
            }
        }
        //----------- boucle serveur
        List<Note> cser= new ArrayList<>();
        if (!downloadedNotes.isEmpty()){
            for (int i = 0; i < downloadedNotes.size(); i++){
                ci=downloadedNotes.get(i);
                c=new Note(ci.getNote(),ci.getUser(),ci.getDate());
                if (recipeNotes.isEmpty()){
                    cser.add(c);}
                else{
                    if (recipeNotes.indexOf(c)==-1) cser.add(c);}
            }
        }
        if (!cser.isEmpty()){
            for (int i = 0; i < cser.size(); i++){
                r.addNote(cser.get(i));
            }
            if (cser.size()>0) {
                mCookbook.updateRecipe(r);
                ret=true;
            }
        }
        if (!cloc.isEmpty()) {
            if (!uploadNotes(r,cloc)) {ret=false;}
            else {ret=true;}
        }
        return ret;
    }
    private Boolean uploadNotes(Recipe r, List<Note> cs){
        if (cs==null) return false;
        if (cs.size()==0) return true;
        HashMap<String, String> data = new HashMap<>();
        DateFormat dateFormat = new SimpleDateFormat(MYSQLDATEFORMAT);
        DecimalFormat formatter = new DecimalFormat("00");
        String s1;
        boolean b=true;
        for (Note c:cs) {
            data.clear();
            data.put("idrecipe", r.getId().toString());
            data.put("idfrom", c.getUser().getId().toString());
            data.put("note", formatter.format(c.getNote()));
            s1 = dateFormat.format(c.getDate());
            data.put("date",s1  );
            String result = mNetUtils.sendPostRequestJson(mSession.getURLPath() + PHPADDNOTETORECIPE, data);
            if (result==null) b=false;
            if (!result.equals("1")) {
```

```
339                 //fdx Log.d(TAG, "Retour de "+ PHPADDNOTETORECIPE
    +" = "+result);
340                 b=false;}
341         }
342         return b;
343     }
344     private Boolean syncTiersRecipe(User user) {
345         boolean ret=false;
346         HashMap<String, String> data = new HashMap<>();
347         data.put("iduser", user.getId().toString());
348         String result = mNetUtils.sendPostRequestJson(mSession.
    getURLPath() + PHPGETSTAMPSTIERS, data);
349         if (result==null)return ret;
350         List<Recipe> tiersRecipe=mNetUtils.parseStampsTiers(result);
351         if (tiersRecipe==null) return ret;
352         boolean withphoto=false,update=false;
353         Recipe rloc, rnew;
354         for(Recipe r:tiersRecipe) {
355             rloc=mCookbook.getRecipe(r.getId());
356             if (rloc==null) { // case recipe tiers not found locally
     => download
357                 rnew=downloadRecipe(r);
358                 mCookbook.addRecipe(rnew);
359                 //fdx Log.d(TAG, "Recette "+ rnew.getTitle()+"
    downloadée");
360                 ret=true;
361             } else {
362                 if (rloc.IsVisible()){ // case recipe tiers active
    and to be updated if necessary
363                     withphoto=IsAfterAndNonNull(r.getDatePhoto(),
    rloc.getDatePhoto());
364                     update=IsAfterAndNonNull(r.getDate(), rloc.
    getDate());
365                     if (r.IsMessage()){// case recipe tiers active
    and status needs update
366                         if (recipeAccepted(r)){
367                             //fdx Log.d(TAG, "Recette "+ rloc.
    getTitle()+" made visible on server");
368                         }
369                     }
370                     if(update || withphoto){
371                         if (updateRecipe(rloc, withphoto)){
372                             mCookbook.updateRecipe(rloc);
373                             //fdx Log.d(TAG, "Recette "+ rloc.
    getTitle()+" updatée (with photo :"+withphoto+")");
374                             ret=true;
375                         }
376                     }
377                     else continue;
378                 }
379             }
380         }
381         return ret;
382     }
```

```
    private boolean IsAfterAndNonNull(Date ds, Date dl){
        Calendar c=Calendar.getInstance();
        c.set(2000,0,1,0,0, 0);
        Date d0=c.getTime();
        boolean morerecent=(ds.compareTo(dl)==1);
        boolean isnotnull=(ds.compareTo(d0)==1);
        return morerecent && isnotnull;
    }

    private Recipe downloadRecipe(Recipe ref){
        if (ref==null) return null;
        HashMap<String,String> data = new HashMap<>();
        data.put("idrecipe", ref.getId().toString().trim());
        data.put("iduser", mSession.getUser().getId().toString().trim
    ());
        String result = mNetUtils.sendPostRequestJson(mSession.
    getURLPath()+PHPGETRECIPEFROMCBWITHPHOTO,data);
        Recipe r=new Recipe();
        try {
            JSONArray jarr1 = new JSONArray(result);
            if (jarr1.length()!=1){
                //fdx Log.d(TAG, " Number of json objects from
    downloadRecipes =" + jarr1.length()+" !");
                return null;
            } else {
                JSONObject obj = jarr1.getJSONObject(0);
                r = new Recipe(UUID.fromString(obj.getString("
    id_recipe")));
                if (!mNetUtils.parseObjectRecipe(r,obj, true, true))
    return null;
            }
        } catch (Exception e) {
            //fdx Log.d(TAG, " Error parsing CB in downloadRecipe" +
    e);
        }
        return r;
    }
    private Boolean updateRecipe(Recipe ref, boolean withphoto){
        if (ref==null) return null;
        HashMap<String,String> data = new HashMap<>();
        data.put("idrecipe", ref.getId().toString().trim());
        data.put("withphoto", (withphoto)?"1":"0");
        String result = mNetUtils.sendPostRequestJson(mSession.
    getURLPath()+ PHPGETRECIPEFROMCB,data);
        Recipe r=new Recipe();
        try {
            JSONArray jarr1 = new JSONArray(result);
            if (jarr1.length()!=1){
                //fdx Log.d(TAG, " Number of json objects from
    downloadRecipes =" + jarr1.length()+" !");
                return null;
            } else {
                JSONObject obj = jarr1.getJSONObject(0);
                if (!mNetUtils.parseObjectRecipe(ref,obj, withphoto,
```

```
428 false)) {
429                     //fdx Log.d(TAG, " Null recipe from JSOn object
    " + result);
430                     return false;
431                 }
432             }
433         } catch (Exception e) {
434             //fdx Log.d(TAG, " Error parsing CB in downloadRecipe" +
    e);
435             return false;
436         }
437         return true;
438     }
439
440     private Boolean recipeAccepted(Recipe r){
441         if (r==null) return null;
442         HashMap<String,String> data = new HashMap<>();
443         data.put("idrecipe", r.getId().toString().trim());
444         data.put("iduser", mSession.getUser().getId().toString());
445         String result = mNetUtils.sendPostRequestJson(mSession.
    getURLPath()+ PHPACCEPTRECIPE,data);
446         if (!result.equals("1")) {
447             //fdx Log.d(TAG, "Retour de "+ PHPACCEPTRECIPE+" = "+
    result);
448             return false;}
449         return true;
450     }
451 }
452
```

```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 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";
        }
    }
}
```

```
1 package com.fdx.cookbook;
2
3 import android.content.Intent;
4 import android.graphics.drawable.AnimationDrawable;
5 import android.os.AsyncTask;
6 import android.os.Bundle;
7 import android.support.v4.content.ContextCompat;
8 import android.support.v7.app.AppCompatActivity;
9 import android.text.Editable;
10 import android.text.TextWatcher;
11 import android.text.method.HideReturnsTransformationMethod;
12 import android.text.method.PasswordTransformationMethod;
13 import android.view.View;
14 import android.view.WindowManager;
15 import android.widget.Button;
16 import android.widget.EditText;
17 import android.widget.ImageView;
18 import android.widget.ProgressBar;
19 import android.widget.TextView;
20
21 import org.json.JSONArray;
22 import org.json.JSONObject;
23
24 import java.text.SimpleDateFormat;
25 import java.util.ArrayList;
26 import java.util.Date;
27 import java.util.HashMap;
28 import java.util.UUID;
29 import java.util.regex.Pattern;
30
31 public class SplashActivity extends AppCompatActivity {
32     private TextView mMoto;
33     private String mFamilyEntered;
34     private String mMemberEntered;
35     private String mPwdEntered;
36     private String mPwdRead;
37     private TextView mEnterFamilyLbl;
38     private EditText mEnterFamily;
39     private TextView mEnterMemberLbl;
40     private EditText mEnterMember;
41     private TextView mEnterPwdLbl;
42     private EditText mEnterPwd;
43     private TextView mEnterMessage;
44     private Button mNewSession;
45     private Button mNewMember;
46     private Button mNewFamily;
47     private ProgressBar mProgressBar;
48     private ImageView mNetAnim;
49     private AnimationDrawable frameNetAnim;
50     private ArrayList<User> memberFamily;
51     private ArrayList<Recipe> mRecipes;
52     private SessionInfo mSession;
53     private NetworkUtils mNetUtils;
54     private User mUser;
```

```
    private TestConnection tc;
    private int mState;
    private static final String TAG = "CB_SplashActivity";
    private final static int NEW_FAMILY=1;
    private final static int NEW_MEMBER=2;
    private final static int NEW_SESSION=3;
    private final static int NEW_PWD=3;
    private final static Integer MINMAX[][]={{8,45},{1,25},{3,25}}; // min max pour family, member, pwd strings
    private static final String REGEX_FAMILY="[-_!?\\w\\p{javaLowerCase}\\p{javaUpperCase}()\\p{Space}]*";
    private static final String REGEX_MEMBER="[-_\\w\\p{javaLowerCase}\\p{javaUpperCase}]*";
    private static final String REGEX_PWD="[-_!?\\w\\p{javaLowerCase}\\p{javaUpperCase}()]*";
    private static final String REGEX[]={REGEX_FAMILY, REGEX_MEMBER, REGEX_PWD};
    private static final String PHPREQFAMILYCOMP="getfamilycomposition.php";
    private static final String PHPREQNEWMEMBER="createnewmember.php";
    private static final String PHPREQNEWFAMILY="createnewfamily.php";
    private static final String PHPREQUPLOADPHOTO="uploadphotointorecipe.php";
    private static final String PHPREQGETCB="getcookbook.php";
    private static final String PHPREQGETNOTES="getnotes.php";
    private static final String PHPREQGETCOMMENTS="getcomments.php";
    private static final String MYSQLDATEFORMAT="yyyy-MM-dd HH:mm:ss";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        mSession= SessionInfo.get(getApplicationContext());
        if ((!mSession.IsEmpty())&&(!mSession.IsReqNewSession())){
            Intent intent=new Intent(getApplicationContext(), RecipeListActivity.class);
            startActivity(intent);
            finish();
        }
        mNetUtils=new NetworkUtils(getApplicationContext());
        memberFamily=new ArrayList<>();
        mRecipes=new ArrayList<>();
        tc = new TestConnection();
        tc.setTestConnexion(getApplicationContext());
        tc.testGo();

        getWindow().setFlags(
                WindowManager.LayoutParams.FLAG_FULLSCREEN,
                WindowManager.LayoutParams.FLAG_FULLSCREEN);
        setContentView(R.layout.activity_splash);
        mMoto=(TextView) findViewById(R.id.splash_moto);
```

```
        mMoto.setText(R.string.P0_moto);
        if (mSession.IsReqNewSession()){
            mFamilyEntered=mSession.getUser().getFamily();
            mMemberEntered=mSession.getUser().getName();
            mPwdEntered="";
        } else {
            mFamilyEntered=getString(R.string.P0HF);
            mMemberEntered=getString(R.string.P0HM);
            mPwdEntered=getString(R.string.P0HP);
        }
        mProgressBar=(ProgressBar) findViewById(R.id.splash_progBar);
        mProgressBar.setProgress(0);
        mEnterMessage=(TextView) findViewById(R.id.splash_edit_message);
        mEnterMessage.setText("");
        mEnterFamilyLbl=(TextView) findViewById(R.id.splash_edit_family_lbl);
        mEnterFamily=(EditText) findViewById(R.id.splash_edit_family);
        mEnterMemberLbl=(TextView) findViewById(R.id.splash_edit_member_lbl);
        mEnterMember=(EditText) findViewById(R.id.splash_edit_member);
        mEnterPwdLbl=(TextView) findViewById(R.id.splash_edit_pwd_lbl);
        mEnterPwd=(EditText) findViewById(R.id.splash_edit_pwd);
        mNewSession=(Button) findViewById(R.id.splash_button_session);
        mNewMember=(Button) findViewById(R.id.splash_button_member);
        mNewFamily=(Button) findViewById(R.id.splash_button_family);
        updateTop();
        mNetAnim=(ImageView) findViewById(R.id.net_anim);
        mNetAnim.setBackgroundResource(R.drawable.network_animation);
        mNetAnim.setVisibility(View.INVISIBLE);
        frameNetAnim = (AnimationDrawable) mNetAnim.getBackground();
        updateDisplayOnAsync(false);

        mEnterFamily.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int start, int count, int after) {}

            @Override
            public void onTextChanged(CharSequence s, int start, int before, int count) {
                String z=s.toString();
                mFamilyEntered=z;
                Integer d= getColorOnCondition(z,NEW_FAMILY);
                mEnterFamily.setBackgroundColor(ContextCompat.getColor(getApplicationContext(), d));
                buttonEnabled((d==R.color.bg_enter_val));
            }

            @Override
```

```
            public void afterTextChanged(Editable s) {}
        });
        mEnterMember.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int start, int count, int after) {}

            @Override
            public void onTextChanged(CharSequence s, int start, int before, int count) {
                String z=s.toString();
                mMemberEntered=z;
                Integer d= getColorOnCondition(z,NEW_MEMBER);
                mEnterMember.setBackgroundColor(ContextCompat.getColor(getApplicationContext(), d));
                buttonEnabled((d==R.color.bg_enter_val));
            }

            @Override
            public void afterTextChanged(Editable s) {}
        });
        mEnterPwd.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int start, int count, int after) {}

            @Override
            public void onTextChanged(CharSequence s, int start, int before, int count) {
                String z=s.toString();
                mPwdEntered=z;
                Integer d= getColorOnCondition(z,NEW_PWD);
                mEnterPwd.setBackgroundColor(ContextCompat.getColor(getApplicationContext(), d));
                buttonEnabled((d==R.color.bg_enter_val));
            }

            @Override
            public void afterTextChanged(Editable s) {}
        });
        mNewSession.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(testConnectStatus()){
                    mState=NEW_SESSION;
                    updateDisplayOnAsync(true);
                    goAsyncTasks();
                }
                tc.testGo();
            }
        });
        mNewMember.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
```

```
                if(testConnectStatus()){
                    mState=NEW_MEMBER;
                    updateDisplayOnAsync(true);
                    goAsyncTasks();
                }
                tc.testGo();
            }
        });
        mNewFamily.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(testConnectStatus()){
                    mState=NEW_FAMILY;
                    updateDisplayOnAsync(true);
                    goAsyncTasks();
                }
                tc.testGo();
            }
        });
    }

    public void ShowHidePass(View view){
        if(view.getId()==R.id.show_pass_btn){
            if(mEnterPwd.getTransformationMethod().equals(PasswordTransformationMethod.getInstance())){
                ((ImageView)(view)).setImageResource(R.drawable.ic_pwd_no_vis);
                mEnterPwd.setTransformationMethod(HideReturnsTransformationMethod.getInstance());
            }
            else{
                ((ImageView)(view)).setImageResource(R.drawable.ic_pwd_vis);
                mEnterPwd.setTransformationMethod(PasswordTransformationMethod.getInstance());

            }
        }
    }
    private void updateTop(){
        mEnterFamilyLbl.setText(R.string.P0LF);
        mEnterFamily.setText(mFamilyEntered);
        mEnterMemberLbl.setText(R.string.P0LM);
        mEnterMember.setText(mMemberEntered);
        mEnterPwdLbl.setText(R.string.P0LP);
        mEnterPwd.setText(mPwdEntered);
        mEnterMessage.setText("");
        mNewSession.setText(R.string.P0BP);
        mNewMember.setText(R.string.P0BM);
        mNewFamily.setText(R.string.P0BF);
    }

    private void updateDisplayOnAsync(Boolean b){
        int indNet,indClick;
```

```
239         if (b){
240             indNet=View.VISIBLE;
241             indClick=View.INVISIBLE;
242             frameNetAnim.start();
243             mProgressBar.setProgress(1);
244         }
245         else {
246             indNet=View.INVISIBLE;
247             indClick=View.VISIBLE;
248             frameNetAnim.stop();
249             mProgressBar.setProgress(0);
250         }
251         mNetAnim.setVisibility(indNet);
252         mProgressBar.setVisibility(indNet);
253         mNewFamily.setVisibility(indClick);
254         mNewMember.setVisibility(indClick);
255         mNewSession.setVisibility(indClick);
256     }
257
258     private Boolean IsLenOK(String s, int min, int max){
259         int l=s.length();
260         return ((l>min)&&(l<max));
261     }
262
263     private Integer getColorOnCondition(String z, Integer state){
264         state=state-1;
265         int retDraw =0;
266         int TRUE=R.color.bg_enter_val;
267         int FALSE=R.color.light_red;
268         retDraw=((Pattern.matches(REGEX[state],z)&&(IsLenOK(z,MINMAX[
    state][0],MINMAX[state][1])))?
269                 TRUE : FALSE);
270         Integer err_mess[]={R.string.P0ERUF,R.string.P0RUM,R.string.
    P0RUP};
271         if (retDraw==FALSE){
272         mEnterMessage.setText(getResources().getString(err_mess[state
    ],MINMAX[state][0],MINMAX[state][1]));}
273         else {mEnterMessage.setText("");}
274         return retDraw;
275     }
276
277     private Boolean testConnectStatus(){
278         mEnterFamily.setBackgroundColor(ContextCompat.getColor(
    getApplicationContext(), R.color.bg_enter_val));
279         mEnterMember.setBackgroundColor(ContextCompat.getColor(
    getApplicationContext(), R.color.bg_enter_val));
280         mEnterPwd.setBackgroundColor(ContextCompat.getColor(
    getApplicationContext(), R.color.bg_enter_val));
281         Boolean ret=true;
282         String message="";
283         if(!mSession.IsConnected()){
284             ret=false;
285             message += getResources().getString(R.string.P0ER_nocon)+
    "\n";
```

```
        }
        mEnterMessage.setText(message);
        return ret;
    }

    private void buttonEnabled(Boolean b){
        mNewSession.setEnabled(b);
        mNewMember.setEnabled(b);
        mNewFamily.setEnabled(b);
    }

    /**
*******************************************************************
*******************
    *                            ASYNC                                                    *
*******************************************************************
*********************/

    private void goAsyncTasks() {

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

            @Override
            protected void onPreExecute() {
                super.onPreExecute();
            }

            @Override
            protected void onPostExecute(Boolean b) {
                super.onPostExecute(b);
                updateDisplayOnAsync(false);
                if (b) {
                    mSession.setStoredUser(mUser);
                    Intent intent=new Intent(getApplicationContext(),
 RecipeListActivity.class);
                    startActivity(intent);
                }
                return;
            }

            @Override
            protected Boolean doInBackground(Void... voids) {
                CookBook cb=CookBook.get(getApplicationContext());
                String s=getFamilyComp();
                mProgressBar.setProgress(2);
                if (s==null){
                    mEnterMessage.setText(R.string.POER_com);
                    return false; } else {
                    memberFamily=parseJsonFamily(s); }
                mProgressBar.setProgress(5);
                Integer cas=goFamilyGet();
                if (cas==0) { return false;}
```

```
                if ((cas==NEW_FAMILY)||(cas==NEW_MEMBER)){
                    if (createMember(cas)){
                        mProgressBar.setProgress(10);
                        mEnterMessage.setText(getString(R.string.POOKM));
                    } else {
                        mEnterMessage.setText(getString(R.string.POERM_new));
                        return false;}
                }
                cb.clearCookBook();
                s=downloadCB();
                if (s==null){
                    //fdx Log(TAG, "download cookbook failed");
                    return false;
                }
                mProgressBar.setProgress(50);
                mRecipes=parseJsonCB(s);
                if (!downloadNotesAndParse()){
                    //fdx Log(TAG, "Failure in reading and parsing Notes");
                    return false;
                }
                mProgressBar.setProgress(70);
                if (!downloadCommentsAndParse()){
                    //fdx Log(TAG, "Failure in reading and parsing Comments");
                    return false;
                }
                mProgressBar.setProgress(70);
                cb.fillCookBook(mRecipes);
                mEnterMessage.setText("Number of recipes : "+mRecipes.size());
                mProgressBar.setProgress(90);
                return true;
            }
        }
        GoAsyncTasks getAsync = new GoAsyncTasks();
        getAsync.execute();
    }

    private String getFamilyComp(){
        HashMap<String,String> data = new HashMap<>();
        data.put("family", mFamilyEntered.trim());
        String result = mNetUtils.sendPostRequestJson(mSession.getURLPath()+PHPREQFAMILYCOMP,data);
        return result;
    }

    private Boolean createMember(int flag){
        String link=mSession.getURLPath();
        if (flag==NEW_MEMBER) {
            link=link+PHPREQNEWMEMBER;
        } else {
```

```
            if (flag==NEW_FAMILY) {
                link=link+PHPREQNEWFAMILY;
            } else {
                //fdx Log(TAG, "Function createmember flag bad value");
                return false;
            }
        }
        HashMap<String,String> data = new HashMap<>();
        data.put("family", mFamilyEntered.trim());
        data.put("pwd", mPwdEntered.trim());
        data.put("name", mMemberEntered.trim());
        mUser=new User(mFamilyEntered.trim(),mMemberEntered.trim());
        data.put("iduser", mUser.getId().toString().trim());
        String result = mNetUtils.sendPostRequestJson(link,data);
        if (result.trim().equals("1")) {
            //fdx Log(TAG, "Function createmember succeed");
            return true;
        } else {
            //fdx Log(TAG, "Function createmember failed");
            return false;
        }
    }

    private String downloadCB(){
        HashMap<String,String> data = new HashMap<>();
        data.put("iduser", mUser.getId().toString().trim());
        data.put("pwd", mPwdEntered.trim());
        String result = mNetUtils.sendPostRequestJson(mSession.getURLPath()+PHPREQGETCB,data);
        return result;
    }

    private Boolean downloadNotesAndParse(){
        HashMap<String,String> data = new HashMap<>();
        data.put("iduser", mUser.getId().toString().trim());
        String json = mNetUtils.sendPostRequestJson(mSession.getURLPath()+PHPREQGETNOTES,data);
        //User u;
        //String s;
        UUID uuid;
        //Date date;
        Note n;
        try {
            JSONArray jarr1=new JSONArray(json);
            for (int i=0; i<jarr1.length(); i++){
                JSONObject obj = jarr1.getJSONObject(i);
                uuid=UUID.fromString(obj.getString("id_recipe"));
                n=mNetUtils.parseObjectNote(obj);
                for(Recipe r:mRecipes){
                    if (r.getId().equals(uuid)){
                        r.addNote(n);
                        break;
                    }
```

```
                }
            }
        } catch (Exception e){
                //fdx Log(TAG, "Failure in parsing JSON Notes "+e);
                return false;
            }
        return true;
    }

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

    public ArrayList<User> parseJsonFamily(String json){
        ArrayList<User> ret=new ArrayList<>();
        User u;
        String name,dateString;
        UUID uuid;
        Date date;
        mPwdRead="";
        try {
            JSONArray jarr1=new JSONArray(json);
            for (int i=0; i<jarr1.length(); i++){
                JSONObject obj = jarr1.getJSONObject(i);
                name=obj.getString("name");
                uuid=UUID.fromString(obj.getString("id_user"));
                u=new User(mFamilyEntered, name );
                u.setId(uuid);
```

```
                dateString=obj.getString("last_sync");
                date=new SimpleDateFormat(MYSQLDATEFORMAT).parse(dateString);
                u.setDate(date);
                ret.add(u);
                mPwdRead=obj.getString("pass");
            }
        }
        catch (Exception e){
            //fdx Log(TAG, "Failure in parsing JSON FamilyGet" +e);
        }
        return ret;
    }
    public ArrayList<Recipe> parseJsonCB(String json){
        ArrayList<Recipe> result=new ArrayList<>();
        Recipe r;
        try {
            JSONArray jarr1 = new JSONArray(json);
            for (int j = 0; j < jarr1.length(); j++) {
                JSONObject obj = jarr1.getJSONObject(j);
                r = new Recipe(UUID.fromString(obj.getString("id_recipe")));
                if (mNetUtils.parseObjectRecipe(r,obj, true, true)) result.add(r);
            }
        } catch (Exception e) {
            //fdx Log(TAG, " Error parsing CB" + e);
        }
        return result;
    }

    private Integer goFamilyGet(){
        switch(mState){
            case NEW_SESSION : {
                if (memberFamily.size()==0) {
                    mEnterMessage.setText(R.string.P0ERM_nofam);
                    mEnterFamily.setBackgroundColor(ContextCompat.getColor(getApplicationContext(), R.color.light_red)); }
                else {
                    mEnterMessage.setText(R.string.P0OKF);
                    User u= new User("","");
                    for(User user:memberFamily){
                        if (user.getName().equals(mMemberEntered)){u=user;}
                    }
                    if (u.getName().equals("")) {
                        mEnterMessage.setText(R.string.P0ERF_nomem);
                        mEnterMember.setBackgroundColor(ContextCompat.getColor(getApplicationContext(), R.color.light_red));
                    } else {
                        if (!mPwdRead.equals(mPwdEntered.trim())){
                            mEnterMessage.setText(R.string.P0ERP_wrong);
                            mEnterPwd.setBackgroundColor(
```

```
ContextCompat.getColor(getApplicationContext(), R.color.light_red));
                        } else {
                            mUser=u;
                        mEnterMessage.setText(getString(R.string.P0OKP, u.getName()));
                        return NEW_SESSION; }
                    }
                }
                break;
            }
            case NEW_FAMILY : {
                if (memberFamily.size()==0) {
                    mEnterMessage.setText(R.string.P0OK_done);
                    return NEW_FAMILY;
                }
                else {
                    mEnterMessage.setText(R.string.P0ERF_notnew);
                    mEnterFamily.setBackgroundColor(ContextCompat.getColor(getApplicationContext(), R.color.light_red));
                }
                break;
            }
            case NEW_MEMBER : {
                if (memberFamily.size()==0) {
                    mEnterMessage.setText(R.string.P0ERM_nofam);
                    mEnterFamily.setBackgroundColor(ContextCompat.getColor(getApplicationContext(), R.color.light_red));
                }
                else {
                    // family found
                    User u= new User("","");
                    for(User user:memberFamily){
                        if (user.getName().equals(mMemberEntered)){u=user;}
                    }
                    if (u.getName().equals("")) {
                        if (!mPwdRead.equals(mPwdEntered.trim())){
                            mEnterMessage.setText(R.string.P0ERP_wrong);
                            mEnterPwd.setBackgroundColor(ContextCompat.getColor(getApplicationContext(), R.color.light_red));
                        } else {
                        mEnterMessage.setText(R.string.P0OKM);
                        return NEW_MEMBER;}
                    } else {
                        mEnterMessage.setText(R.string.P0ERM_notnew);
                        mEnterMember.setBackgroundColor(ContextCompat.getColor(getApplicationContext(), R.color.light_red));
                    }
                }
                break;
            }
            default : {
                //fdx Log(TAG, "Switch case on mState anomaly : "+
```

```
mState);
                }
            }
            return 0;
        }
}

```

```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;
// ******************************************************************
//  to use :
//        TestConnection t;
//        t = new TestConnection();
//        t.setTestConnexion(getApplicationContext());
//        t.testGo(); start async => result in Boolean  mSession.IsConnected()
//*************************************************************

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

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

    public void testGo(){

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

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

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

```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.COMMENTS+", "+
                RecipeDbSchema.RecipeTable.Cols.STATUS+", "+
                RecipeDbSchema.RecipeTable.Cols.NOTES+", "+
                RecipeDbSchema.RecipeTable.Cols.MESSAGE+", "+
                RecipeDbSchema.RecipeTable.Cols.MESSAGE_FROM+", "+
                RecipeDbSchema.RecipeTable.Cols.TS_RECIPE+", "+
                RecipeDbSchema.RecipeTable.Cols.TS_PHOTO+", "+
                RecipeDbSchema.RecipeTable.Cols.TS_COMMENT+", "+
                RecipeDbSchema.RecipeTable.Cols.TS_NOTE+
                ")"
        );

    }
```

```

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

    }
}

```

```java
package com.fdx.cookbook;

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

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

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

```

```java
package com.fdx.cookbook;

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

import java.util.UUID;

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

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

```
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
```

```
1 package com.fdx.cookbook;
2
3 import android.app.Activity;
4 import android.content.Intent;
5 import android.content.pm.PackageManager;
6 import android.content.pm.ResolveInfo;
7 import android.graphics.Bitmap;
8 import android.net.Uri;
9 import android.os.Bundle;
10 import android.provider.MediaStore;
11 import android.support.v4.app.Fragment;
12 import android.support.v4.content.FileProvider;
13 import android.text.Editable;
14 import android.text.TextWatcher;
15 import android.view.LayoutInflater;
16 import android.view.Menu;
17 import android.view.MenuInflater;
18 import android.view.MenuItem;
19 import android.view.View;
20 import android.view.ViewGroup;
21 import android.widget.AdapterView;
22 import android.widget.ArrayAdapter;
23 import android.widget.EditText;
24 import android.widget.ImageButton;
25 import android.widget.ImageView;
26 import android.widget.ScrollView;
27 import android.widget.Spinner;
28 import android.widget.TextView;
29
30 import java.io.File;
31 import java.io.FileOutputStream;
32 import java.io.IOException;
33 import java.net.MalformedURLException;
34 import java.net.URL;
35 import java.util.Date;
36 import java.util.List;
37 import java.util.UUID;
38
39 public class RecipeEditFragment extends Fragment {
40     // Constantes
41     private static final String ARG_RECIPE_ID="recipe_id";
42     private static final String DIALOG_DATE="DialogDate";
43     private static final int REQUEST_DATE=0;
44     private static final int REQUEST_PHOTO= 2;
45     private static final String TAG = "CB_RecipeEditFragment";
46     private static final String FPROVIDER="com.fdx.cookbook.
   fileprovider";
47     private static final String UUIDNULL="00000000-0000-0000-0000-
   000000000000";
48     // Variables internes
49     private Recipe mRecipe;
50     private Recipe mRecipeInit;
51     private UUID mRecipeId;
52     private File mPhotoFile;
```

```
    private EditText mTitleField;
    private EditText mSourceField;
    private EditText mSourceUrl;
    private EditText mNbPersField;
    //    private Button mDateButton;
    private TextView[] mSTextView;
    private EditText mNewStepField;
    private ImageButton mNewStepEnter;
    private ImageButton mNewStepBack;
    private int mStepNb;
    private TextView[] mSTextIngView;
    private EditText mNewIngField;
    private ImageButton mNewIngEnter;
    private ImageButton mNewIngBack;
    private int mIngNb;
    private Spinner mSeasonSpinner;
    private Spinner mDifficultySpinner;
    private ScrollView mScroll;
    private ImageView mPhotoView;

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

    @Override
    public void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        mRecipe=new Recipe();
        mRecipeInit=new Recipe();
        mRecipeId=(UUID) getArguments().getSerializable(ARG_RECIPE_ID);
        setHasOptionsMenu(true);
        SessionInfo session=SessionInfo.get(getActivity());
        if (!IsRecipeNew(mRecipeId)) {
            mRecipe=CookBook.get(getActivity()).getRecipe(mRecipeId);
            mRecipeInit=CookBook.get(getActivity()).getRecipe(mRecipeId);
            mPhotoFile=CookBook.get(getActivity()).getPhotoFile(mRecipe);
        } else {
            mRecipe.setOwner(session.getUser());
            mRecipeInit.setOwner(session.getUser());
            mRecipeInit.setId(mRecipe.getId());
        }
        mStepNb=mRecipe.getNbStep();
        mIngNb=mRecipe.getNbIng();
    }

    @Override
    public void onPause(){
```

```
        super.onPause();
        //mRecipe.updateTS(AsynCallFlag.NEWRECIPE, !mRecipe.hasNotChangedSince(mRecipeInit));
        //CookBook.get(getActivity()).updateRecipe(mRecipe);
    }

    @Override
    public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
        super.onCreateOptionsMenu(menu, inflater);
        inflater.inflate(R.menu.fragment_recipe, menu);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.recipe_menu_done:
                if (mRecipe.getTitle().length()>0){
                    if (!mRecipe.hasNotChangedSince(mRecipeInit)){
                    mRecipe.updateTS(AsynCallFlag.NEWRECIPE, true);}
                    if (IsRecipeNew(mRecipeId)){
                        CookBook.get(getActivity()).addRecipe(mRecipe);
                    } else {
                        CookBook.get(getActivity()).updateRecipe(mRecipe);
                    }
                }
                getActivity().onBackPressed();
                return true;
            case R.id.recipe_menu_delete:
                if (!IsRecipeNew(mRecipeId)){
                    CookBook.get(getActivity()).markRecipeToDelete(mRecipe);
                }
                getActivity().onBackPressed();
                return true;
            default:
                return super.onOptionsItemSelected(item);
        }
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState){
        final View v=inflater.inflate(R.layout.fragment_recipe_edit, container, false);
        mScroll=(ScrollView) v.findViewById(R.id.fragment_recipe_scroll);
        PackageManager packageManager=getActivity().getPackageManager();
            final Intent captureImage=new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
            boolean canTakePhoto=(mPhotoFile != null) &&
```

```
                    (captureImage.resolveActivity(packageManager)!= null);
        mPhotoView=(ImageView) v.findViewById(R.id.recipe_photo);
        mPhotoView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Uri uri= FileProvider.getUriForFile(getActivity(),
                        FPROVIDER, mPhotoFile);
                captureImage.putExtra(MediaStore.EXTRA_OUTPUT, uri);
                List<ResolveInfo> cameraActivities=getActivity()
                        .getPackageManager().queryIntentActivities(captureImage,
                                PackageManager.MATCH_DEFAULT_ONLY);
                for(ResolveInfo activity : cameraActivities){
                    getActivity().grantUriPermission(activity.activityInfo.packageName,
                            uri, Intent.FLAG_GRANT_WRITE_URI_PERMISSION);
                }
                startActivityForResult(captureImage, REQUEST_PHOTO);
            }
        });
        updatePhotoView();
        mTitleField= (EditText) v.findViewById(R.id.recipe_title);
        mTitleField.setText(mRecipe.getTitle());
        mTitleField.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int start, int count, int after) {
                // blank
            }

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

            @Override
            public void afterTextChanged(Editable s) {
                //blank
            }
        });

        mSourceField= (EditText) v.findViewById(R.id.recipe_source);
        mSourceField.setText(mRecipe.getSource());
        mSourceField.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int start, int count, int after) {
                // blank
            }

            @Override
```

```
            public void onTextChanged(CharSequence s, int start, int before, int count) {
                mRecipe.setSource(s.toString());
            }

            @Override
            public void afterTextChanged(Editable s) {

            }
        });

        mSourceUrl=(EditText) v.findViewById(R.id.recipe_source_url);
        mSourceUrl.setText(mRecipe.getSource_url().toString());
        mSourceUrl.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int start, int count, int after) {

            }

            @Override
            public void onTextChanged(CharSequence s, int start, int before, int count) {
                if (s.toString().equals("")) {} else {
                    try {
                        URL url=new URL(s.toString());
                        mRecipe.setSource_url(url);
                    } catch (MalformedURLException e) {
                        //fdx Log(TAG, "onTextChanged de mSource_url >" +s.toString()+"< Failed >"+ e);
                    }
                    }
            }

            @Override
            public void afterTextChanged(Editable s) {

            }
        });

        mNbPersField= (EditText) v.findViewById(R.id.recipe_nbpers);
        mNbPersField.setText(mRecipe.getNbPers()+"");
        mNbPersField.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int start, int count, int after) {

            }

            @Override
            public void onTextChanged(CharSequence s, int start, int before, int count) {
                if (s.toString().equals("")) {} else {
                    int nb_entered = Integer.parseInt(s.toString());
```

```
                    if ((nb_entered > 0) && (nb_entered < 13)) {
                        mRecipe.setNbPers(nb_entered);
                    }
                }
            }

            @Override
            public void afterTextChanged(Editable s) {

            }
        });



        mSeasonSpinner= (Spinner) v.findViewById(R.id.recipe_season);
        ArrayAdapter<CharSequence> adapterSeason = ArrayAdapter.createFromResource(getContext(),
                R.array.recipe_season_array, android.R.layout.simple_spinner_item);
        mSeasonSpinner.setAdapter(adapterSeason);
        mSeasonSpinner.setSelection(RecipeSeason.getIndex(mRecipe.getSeason()));
        mSeasonSpinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
                mRecipe.setSeason(RecipeSeason.getSeason(position));
            }

            @Override
            public void onNothingSelected(AdapterView<?> parent) {

            }
        });

        mDifficultySpinner= (Spinner) v.findViewById(R.id.recipe_difficulty);
        ArrayAdapter<CharSequence> adapterDifficulty = ArrayAdapter.createFromResource(getContext(),
                R.array.recipe_difficulty_array, android.R.layout.simple_spinner_item);
        mDifficultySpinner.setAdapter(adapterDifficulty);
        mDifficultySpinner.setSelection(RecipeDifficulty.getIndex(mRecipe.getDifficulty()));
        mDifficultySpinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
                mRecipe.setDifficulty(RecipeDifficulty.getDifficulty(position));
            }

```

```
            @Override
            public void onNothingSelected(AdapterView<?> parent) {

            }
        });

        final int[] rID= {R.id.recipe_S1,R.id.recipe_S2,R.id.recipe_S3,R.id.recipe_S4,
                R.id.recipe_S5,R.id.recipe_S6,R.id.recipe_S7,R.id.recipe_S8,R.id.recipe_S9};
        mSTextView= new TextView[mRecipe.getNbStepMax()];
        for(int i=0;i<mRecipe.getNbStepMax();i++) {
            mSTextView[i] = (TextView) v.findViewById(rID[i]);
        }
        mNewStepEnter=(ImageButton) v.findViewById(R.id.recipe_step_enter);
        mNewStepBack=(ImageButton) v.findViewById(R.id.recipe_step_back);
        updateListStep(v);

        mNewStepField= (EditText) v.findViewById(R.id.recipe_new_step);
        mNewStepField.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int start, int count, int after) {
                // blank
            }

            @Override
            public void onTextChanged(CharSequence s, int start, int before, int count) {
                mRecipe.setStep(mStepNb+1, s.toString());
            }
            @Override
            public void afterTextChanged(Editable s) {

            }
        });

        mNewStepEnter.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
//                Toast.makeText(getActivity(), mRecipe.getStep(mStepNb+1)+"/"+mStepNb,
//                        Toast.LENGTH_SHORT).show();
                mNewStepField.setVisibility(updateListStep(v));
                mNewStepField.getText().clear();
            }
        });

        mNewStepBack.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
```

```
                mRecipe.setStep(mStepNb, "");
                mRecipe.setStep(mStepNb+1, "");
                mNewStepField.setVisibility(updateListStep(v));
                mNewStepField.getText().clear();
            }
        });

        final int[] rIngID= {R.id.recipe_I01,R.id.recipe_I02,R.id.recipe_I03,R.id.recipe_I04,
                R.id.recipe_I05,R.id.recipe_I06,R.id.recipe_I07,R.id.recipe_I08,
                R.id.recipe_I09,R.id.recipe_I10,R.id.recipe_I11,R.id.recipe_I12,
                R.id.recipe_I13,R.id.recipe_I14,R.id.recipe_I15};
        mSTextIngView= new TextView[mRecipe.getNbIngMax()];
        for(int i=0;i<mRecipe.getNbIngMax();i++) {
            mSTextIngView[i] = (TextView) v.findViewById(rIngID[i]);
        }
        mNewIngEnter=(ImageButton) v.findViewById(R.id.recipe_ing_enter);
        mNewIngBack=(ImageButton) v.findViewById(R.id.recipe_ing_back);
        updateListIng(v);

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

            }

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

            @Override
            public void afterTextChanged(Editable s) {

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

```
                mRecipe.setIngredient(mIngNb+1, "");
                mNewIngField.setVisibility(updateListIng(v));
                mNewIngField.getText().clear();
            }
        });

        return v;
    }


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

        if (requestCode==REQUEST_PHOTO){
            Uri uri=FileProvider.getUriForFile(getActivity(),
                    FPROVIDER, mPhotoFile);
            getActivity().revokeUriPermission(uri, Intent.FLAG_GRANT_WRITE_URI_PERMISSION);
            if (!ResizePhoto(mRecipe)){
                // pb
            }
            mRecipe.setDatePhoto(new Date());
            mRecipe.updateTS(AsynCallFlag.NEWPHOTO,true);
            updatePhotoView();
        }
    }

    private String getRecipeReport(){
        String report="";
        Integer iplus=0;
        report =getString(R.string.P2RE_title, mRecipe.getTitle())+"\n";
        report +=getString(R.string.P2RE_user,mRecipe.getOwner().getNameComplete())+"\n";

         if(!mRecipe.getSource_url_name().equals("")){
            report += getString(R.string.P2RE_url, mRecipe.getSource_url_name())+"\n";
        }
        for(int i=0;i<mRecipe.getNbIng();i++){
            report += getString(R.string.P2RE_ing, mRecipe.getIngredient(i+1))+"\n";
        }
        for(int i=0;i<mRecipe.getNbStep();i++){
            iplus=i+1;
            report += getString(R.string.P2RE_step, iplus+"",
                    mRecipe.getStep(i+1))+"\n";
        }
        report +=getString(R.string.P2RE_end);
        return report;
```

```
    }

    private void updatePhotoView(){
        if (mPhotoFile==null || !mPhotoFile.exists()){
            mPhotoView.setImageDrawable(getResources().getDrawable(R.drawable.ic_recipe_camera));
        } else {
            Bitmap bitmap=PictureUtils.getScaledBitmap(mPhotoFile.getPath(), getActivity());
            mPhotoView.setImageBitmap(bitmap);
        }
    }

    private int updateListStep(View v) {
        int imax=mRecipe.getNbStepMax(), iplus;
        String[] display=new String[imax];
        mStepNb = mRecipe.getNbStep();
        int gone=View.GONE;
        int visible=View.VISIBLE;
        mSTextView[0].setText("1...");
        mSTextView[0].setVisibility(visible);
        for(int i=0;i<imax;i++){
            iplus=i+1; display[i]=iplus+". "+mRecipe.getStep(i+1);
            if (mStepNb>0){mSTextView[i].setText(display[i]);}
            if (i>=0){mSTextView[i].setVisibility((mStepNb>i)?visible:gone);}
        }
        mNewStepBack.setVisibility((mStepNb==0)? gone:visible);
        mNewStepEnter.setVisibility((mStepNb==imax)? gone:visible);
        return ((mStepNb==imax)?gone:visible);
    }

    private int updateListIng(View v) {
        int imax=mRecipe.getNbIngMax(), iplus;
        String[] display=new String[imax];
        mIngNb = mRecipe.getNbIng();
        int gone=View.GONE;
        int visible=View.VISIBLE;
        mSTextIngView[0].setText("-...");
        mSTextIngView[0].setVisibility(visible);
        for(int i=0;i<imax;i++){
            iplus=i+1; display[i]="-"+mRecipe.getIngredient(i+1);
            if (mIngNb>0){mSTextIngView[i].setText(display[i]);}
            if (i>=0){mSTextIngView[i].setVisibility((mIngNb>i)?visible:gone);}
        }
        mNewIngBack.setVisibility((mIngNb==0)? gone:visible);
        mNewIngEnter.setVisibility((mIngNb==imax)? gone:visible);
        return ((mIngNb==imax)?gone:visible);
    }

    private Boolean IsRecipeNew(UUID uuid){
        return uuid.toString().equals(UUIDNULL);
    }
```

```java

    private Boolean ResizePhoto(Recipe r) {
        File f = CookBook.get(getActivity()).getPhotoFile(r);
        if (f==null || !f.exists()){
            //fdx Log(TAG, "No file from Cookbook for this recipe :" + r.getTitle()+" Error CB004");
            return false;
        } else {
            Bitmap bitmap=PictureUtils.getScaledBitmap(mPhotoFile.getPath(), getActivity(), 600);
            f.delete();
            try {
                if(!f.createNewFile()) {
                    //fdx Log(TAG, "File not deleted ! Error CB001");
                    return false;
                }
            } catch (IOException e) {
                //fdx Log(TAG, "Error in creating new file : "+e+" Error CB002");
                return false;
            }
            try {

                FileOutputStream fOut = new FileOutputStream(f);
                bitmap.compress(Bitmap.CompressFormat.JPEG, 100, fOut);
                fOut.flush();
                fOut.close();
                return true;
            } catch (IOException e) {
                //fdx Log(TAG, "Error in reducing and saving file "+" Error CB003");
                return false;
            }
        }
    }
}

```

```java
package com.fdx.cookbook;

import android.support.v4.app.Fragment;

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

```

```
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.Bundle;
8 import android.support.annotation.NonNull;
9 import android.support.v4.app.Fragment;
10 import android.support.v4.app.FragmentManager;
11 import android.support.v7.app.AppCompatActivity;
12 import android.support.v7.widget.LinearLayoutManager;
13 import android.support.v7.widget.RecyclerView;
14 import android.view.LayoutInflater;
15 import android.view.Menu;
16 import android.view.MenuInflater;
17 import android.view.MenuItem;
18 import android.view.View;
19 import android.view.ViewGroup;
20 import android.widget.ImageView;
21 import android.widget.RatingBar;
22 import android.widget.TextView;
23 import android.widget.Toast;
24
25 import java.io.File;
26 import java.text.DecimalFormat;
27 import java.util.ArrayList;
28 import java.util.Collections;
29 import java.util.List;
30 import java.util.UUID;
31
32 //-----------------------------------------------------------------------
   ------------------------------
33 // print 16/07/2021
34 // Saved GitHUb V2.0
35 //-----------------------------------------------------------------------
   --------------------------
36 public class RecipeListFragment extends Fragment {
37     private RecyclerView mRecipeRecyclerView;
38     private RecipeAdapter mAdapter;
39     private SessionInfo mSession;
40     private Recipe mRecipeInit;
41     private MenuItem mMessageItem;
42     private static final String SAVED_SORT_STATUS="sort";
43     private ListMask mListMask;
44     private static final String TAG = "CB_RecipeListFragment";
45     private static final String DIALOG_RATE = "DialogRate";
46     private static final int REQUEST_RATE = 0;
47     private static final String UUIDNULL="00000000-0000-0000-0000-
   000000000000";
48
49     @Override
50     public void onCreate(Bundle savedInstanceState) {
51         super.onCreate(savedInstanceState);
```

```
        setHasOptionsMenu(true);
        mSession= SessionInfo.get(getActivity());
        mRecipeInit=new Recipe();
        AppCompatActivity activity = (AppCompatActivity) getActivity();
        User u=mSession.getUser();
        activity.getSupportActionBar().setSubtitle(getString(R.string.P2U_txt,u.getName(),u.getFamily()));
        AsyncCallClass instanceAsync = new AsyncCallClass(getContext());
        instanceAsync.execute();
        mListMask=new ListMask(mSession.getUser());
    }

    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (resultCode != Activity.RESULT_OK) {
            return;
        }
        //todo P3 enlever request rate reliquat
        if (requestCode == REQUEST_RATE) {
            Integer rate = (Integer) data
                    .getSerializableExtra(RatePickerFragment.EXTRA_RATE);
            updateUI();
        }
    }

    @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()));
        if (savedInstanceState!=null){
            mListMask.updateFromString(savedInstanceState.getString(SAVED_SORT_STATUS));
        }
        updateUI();
        return view;
    }

    @Override
    public void onResume(){
        super.onResume();
        updateUI();
    }

    @Override
```

```
    public void onSaveInstanceState(Bundle outState) {
        super.onSaveInstanceState(outState);
        outState.putString(SAVED_SORT_STATUS, mListMask.convertToString());
    }

    @Override
    public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
        super.onCreateOptionsMenu(menu, inflater);
        inflater.inflate(R.menu.fragment_recipe_list, menu);
        mMessageItem = menu.findItem(R.id.new_mail);
        CookBook cookbook=CookBook.get(getActivity());
        mMessageItem.setVisible(cookbook.isThereMail());
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item){
        switch (item.getItemId()){
            case R.id.new_recipe:
                Intent intent= RecipeActivity.newIntent(getActivity(), UUID.fromString( UUIDNULL));
                startActivity(intent);
                return true;
            case R.id.list_logout:
                mSession.setReqNewSession(true);
                Intent intent2=new Intent(getActivity().getApplicationContext(), SplashActivity.class);
                startActivity(intent2);
                return true;
            case R.id.list_sync:
                AsyncCallClass instanceAsync = new AsyncCallClass(getContext());
                instanceAsync.execute();
                return true;
            case R.id.new_mail:
                CookBook cookbook=CookBook.get(getActivity());
                if (cookbook.isThereMail()) {
                Intent intent3= RecipeMailDisplayActivity.newIntent(getActivity());
                startActivity(intent3); } else {
                mMessageItem.setVisible(cookbook.isThereMail());}
                return true;
            case R.id.list_filter:
                mListMask.reset();
                updateUI();
                return true;
            case R.id.feedback:
                Intent intent3 = new Intent(Intent.ACTION_SENDTO);
                intent3.putExtra(Intent.EXTRA_SUBJECT, "Feedback on application");
                intent3.putExtra(Intent.EXTRA_TEXT, "Your comment");
                intent3.setData(Uri.parse("mailto:cookbookfamily.founder@gmail.com"));
```

```
                intent3.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                startActivity(intent3);
                return true;
            default:
                return super.onOptionsItemSelected(item);
        }
    }

    private void updateUI() {
        CookBook cookbook=CookBook.get(getActivity());
        List<Recipe> recipes_in=cookbook.getRecipes();
        List<Recipe> recipes=new ArrayList<>();
        recipes.addAll(recipes_in);
        for(Recipe r:recipes_in){
            if (!mListMask.isRecipeSelected(r)){
                recipes.remove(r);
            }
        }
        if (recipes.isEmpty()){
            Toast.makeText(getContext(), getString(R.string.TEY), Toast.LENGTH_LONG).show();
        } else {
            Collections.sort(recipes,
                      (r1, r2) -> {
                            return (r1.getTitle().compareTo(r2.getTitle()));
                      });

            if (mListMask.isNoteSorted()) {
                Collections.sort(recipes,
                        (r1, r2) -> {
                            return ((int) (r2.getNoteAvg() - r1.getNoteAvg()));
                        });
            }
        }
        if (mAdapter==null){
            mAdapter=new RecipeAdapter(recipes);
            mRecipeRecyclerView.setAdapter(mAdapter);
        } else {
            mAdapter.setRecipes(recipes);
            mAdapter.notifyDataSetChanged();
        }
    }

    private class RecipeHolder extends RecyclerView.ViewHolder
                                implements View.OnClickListener {
        private TextView mTitleTextView;
        private TextView mSourceTextView;
        private TextView mNoteTextView;
        private TextView mDifficulty;
        private ImageView mEditIcon;
        private ImageView mPhotoView;
        private ImageView mSunIcon;
```

```
        private ImageView mIceIcon;
        private File mPhotoFile;
        private Recipe mRecipe;
        private RatingBar mRating;
        public RecipeHolder(LayoutInflater inflater,ViewGroup parent) {
            super(inflater.inflate(R.layout.list_item_recipe, parent, false));
            itemView.setOnClickListener(this);
            mTitleTextView= (TextView) itemView.findViewById(R.id.recipe_title);
            mSourceTextView= (TextView) itemView.findViewById(R.id.recipe_source);
            mNoteTextView= (TextView) itemView.findViewById(R.id.recipe_note);
            mEditIcon=(ImageView) itemView.findViewById(R.id.recipe_edit);
            mSunIcon=(ImageView) itemView.findViewById(R.id.recipe_img_sun);
            mIceIcon=(ImageView) itemView.findViewById(R.id.recipe_img_ice);
            mDifficulty=(TextView) itemView.findViewById(R.id.recipe_difficulty);
            mPhotoView=(ImageView) itemView.findViewById(R.id.recipe_photo);
            mPhotoFile=CookBook.get(getActivity()).getPhotoFile(mRecipe);
            mRating=(RatingBar) itemView.findViewById(R.id.recipe_list_ratingBar);
            mTitleTextView.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    Intent intent=RecipeDisplayActivity.newIntent(getActivity(),mRecipe.getId());
                    startActivity(intent);
                }
            });
            mNoteTextView.setOnClickListener(new View.OnClickListener(){
                @Override
                public void onClick(View v){
                    Toast.makeText(getContext(), getString(mListMask.toggleSortNote()),
                            Toast.LENGTH_SHORT ).show();
                    updateUI();
                }
            });
            mSourceTextView.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    Toast.makeText(getContext(), getString(mListMask.toggleUser(mRecipe.getOwner()),
```

```
                                mRecipe.getOwner().getName()),Toast.LENGTH_SHORT ).show();
                        updateUI();
                    }
                });
                mSunIcon.setOnClickListener(new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        Toast.makeText(getContext(), getString(mListMask.toggleSun()),Toast.LENGTH_SHORT ).show();
                        updateUI();
                    }
                });
                mIceIcon.setOnClickListener(new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        Toast.makeText(getContext(), getString(mListMask.toggleWinter()),
                                Toast.LENGTH_SHORT ).show();
                        updateUI();
                    }
                });
                mDifficulty.setOnClickListener(new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        Toast.makeText(getContext(),
                                getString(mListMask.toggleDifficulty(mRecipe.getDifficulty())),
                                Toast.LENGTH_SHORT ).show();
                        updateUI();
                    }
                });
                mEditIcon.setOnClickListener(new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        Intent intent= RecipeActivity.newIntent(getActivity(),mRecipe.getId());
                        startActivity(intent);
                    }
                });
                mPhotoView.setOnClickListener(new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        FragmentManager manager = getFragmentManager();
                        RatePickerFragment dialog = RatePickerFragment.newInstance(mRecipe.getId());
                        dialog.setTargetFragment(RecipeListFragment.this, REQUEST_RATE);
                        dialog.show(manager, DIALOG_RATE);
                    }
                });
            }
```

```
        public void bind(Recipe recipe){
            mRecipe=recipe;
            mTitleTextView.setText(mRecipe.getTitle());
            mSourceTextView.setText(getString(R.string.P1_de, mRecipe.getOwner().getName()));
            //mSourceTextView.setText(mRecipe.getFlag()); // for debug
            mRating.setRating((float) mRecipe.getNoteAvg());
            DecimalFormat df = new DecimalFormat("#.#");
            mNoteTextView.setText(df.format(mRecipe.getNoteAvg())+"/5");
            mPhotoFile=CookBook.get(getActivity()).getPhotoFile(mRecipe);
            int idff= RecipeDifficulty.getIndex(mRecipe.getDifficulty());
            String[] stringArray = getResources().getStringArray(R.array.recipe_difficulty_array);
            mDifficulty.setText(stringArray[idff]);
            if (mPhotoFile==null || !mPhotoFile.exists()){
                mPhotoView.setImageDrawable(getResources().getDrawable(R.drawable.ic_recipe_see));
            } else {
                Bitmap bitmap=PictureUtils.getScaledBitmap(mPhotoFile.getPath(), getActivity());
                mPhotoView.setImageBitmap(bitmap);
            }
            mIceIcon.setImageResource((mRecipe.IsWinter()) ? R.drawable.ic_recipe_ice : R.drawable.ic_recipe_ice_disabled);
            mSunIcon.setImageResource((mRecipe.IsSummer()) ? R.drawable.ic_recipe_sun : R.drawable.ic_recipe_sun_disabled);
            mEditIcon.setVisibility((mRecipe.getOwner().getId().equals(mSession.getUser().getId())) ? View.VISIBLE : View.GONE);
        }

        @Override
        public void onClick(View v){

        }
    }

    private class RecipeAdapter extends RecyclerView.Adapter<RecipeHolder> {
        private List<Recipe> mRecipes;
        public RecipeAdapter(List<Recipe> recipes){
            mRecipes=recipes;
        }

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

```
        }

        @Override
        public void onBindViewHolder(@NonNull 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;
        }
    }
}

```

```java
package com.fdx.cookbook;

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

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

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

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

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

```
            ing[i]=getString(getColumnIndex(RecipeTable.Cols.ING[i]));
            r.setIngredient(i+1, ing[i]);
        }
        String season = getString(getColumnIndex(RecipeTable.Cols.SEASON));
        r.setSeason(RecipeSeason.valueOf(season));
        String difficulty = getString(getColumnIndex(RecipeTable.Cols.DIFFICULTY));
        r.setDifficulty(RecipeDifficulty.valueOf(difficulty));
        String comments = getString(getColumnIndex(RecipeTable.Cols.COMMENTS));
        r.getCommentsDeserialised(comments);
        String status = getString(getColumnIndex(RecipeTable.Cols.STATUS));
        r.setStatus(StatusRecipe.valueOf(status));
        String notes = getString(getColumnIndex(RecipeTable.Cols.NOTES));
        r.getNotesDeserialised(notes);
        String message = getString(getColumnIndex(RecipeTable.Cols.MESSAGE));
        r.setMessage(message);
        String fromString = getString(getColumnIndex(RecipeTable.Cols.MESSAGE_FROM));
        r.getFromDeserialized(fromString);
        int tsrecipe = getInt(getColumnIndex(RecipeTable.Cols.TS_RECIPE));
        r.updateTS(AsynCallFlag.NEWRECIPE, (tsrecipe==1));
        int tsphoto = getInt(getColumnIndex(RecipeTable.Cols.TS_PHOTO));
        r.updateTS(AsynCallFlag.NEWPHOTO, (tsphoto==1));
        int tscomment = getInt(getColumnIndex(RecipeTable.Cols.TS_COMMENT));
        r.updateTS(AsynCallFlag.NEWCOMMENT, (tscomment==1));
        int tsnote = getInt(getColumnIndex(RecipeTable.Cols.TS_NOTE));
        r.updateTS(AsynCallFlag.NEWRATING, (tsnote==1));
        return r;
    }
}
```

```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);
    }
}

```

```
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 static final int REQUEST_PHOTO= 2;
39     private Recipe mRecipe;
40     private File mPhotoFile;
41     private int mStepNb;
42     private int mIngNb;
43     private String newcomment;
44     private String DEFAULT_URL="https://wwww.cookbookfamily.com";
45     private SessionInfo mSession;
46     private ImageView mDPhotoView;
47     private TextView mDTitleText;
48     private RatingBar mDRatingBar;
49     private TextView mDRatingBarText;
50     private TextView mDAuthorText;
51     private TextView mDDifficulty;
52     private ImageView mSunIcon;
53     private ImageView mIceIcon;
54     private TextView mDSourceText;
```

```
    private TextView mDSourceUrl;
    private TextView mDIngTitle;
    private TextView mDComTitle;
    private String mToFamily;
    private String mToMember;
    private String mToMessage;
    private TextView[] mDStepText;
    private TextView[] mDIngText;
    private TextView[] mDComText;
    private EditText mDNexComment;
    private ImageView mDEnterComment;
    private ScrollView mScroll;
    //private Snackbar snack;
    private final static Integer MINMAX[][]={{8,45},{1,25},{3,25}}; // min max pour family, member, pwd strings
    private static final String REGEX_FAMILY="[-_!?\\w\\p{javaLowerCase}\\p{javaUpperCase}()\\p{Space}]*";
    private static final String REGEX_MEMBER="[-_\\w\\p{javaLowerCase}\\p{javaUpperCase}]*";

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

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        mRecipe=new Recipe();
        UUID recipeId=(UUID) getArguments().getSerializable(ARG_RECIPE_ID);
        mRecipe=CookBook.get(getActivity()).getRecipe(recipeId);
        mPhotoFile=CookBook.get(getActivity()).getPhotoFile(mRecipe);
        mSession= SessionInfo.get(getActivity());
        setHasOptionsMenu(true);
        mStepNb=mRecipe.getNbStep();
        mIngNb=mRecipe.getNbIng();
        mToFamily="";
        mToMember="";
    }
    @Override
    public void onPause(){
        super.onPause();
        CookBook.get(getActivity()).updateRecipe(mRecipe);
    }

    @Override
    public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
        super.onCreateOptionsMenu(menu, inflater);
        inflater.inflate(R.menu.fragment_recipe_display, menu);
```

```
        MenuItem menuItem = menu.findItem(R.id.recipe_menu_edit);
        if(!mRecipe.getOwner().getId().equals(mSession.getUser().getId())){
            //getActivity().invalidateOptionsMenu();
            menuItem.setVisible(false);
        }
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.recipe_menu_report:
                Intent i=new Intent(Intent.ACTION_SEND);
                i.setType("text/plain");
                i.putExtra(Intent.EXTRA_TEXT, getRecipeReport());
                i.putExtra(Intent.EXTRA_SUBJECT,getString(R.string.P2MU_send));
                startActivity(i);
                return true;
            case R.id.recipe_menu_edit:
                Intent intent= RecipeActivity.newIntent(getActivity(),mRecipe.getId());
                startActivity(intent);
                return true;
            case R.id.recipe_mail:
                LinearLayout layout = new LinearLayout(getContext());
                layout.setOrientation(LinearLayout.VERTICAL);
                final EditText familyBox = new EditText(getContext());
                familyBox.setHint(getString(R.string.P0HF));
                layout.addView(familyBox);
                final EditText memberBox = new EditText(getContext);
                memberBox.setHint(getString(R.string.P0HM));
                layout.addView(memberBox);
                final EditText messageBox = new EditText(getContext());
                messageBox.setHint(getString(R.string.P4M_mes));
                layout.addView(messageBox);
                AlertDialog.Builder builder = new AlertDialog.Builder(getContext());
                builder.setTitle(getString(R.string.P4M_title));
                builder.setView(layout);
                builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        mToFamily = familyBox.getText().toString();
                        mToMember = memberBox.getText().toString();
                        mToMessage = messageBox.getText().toString();
                        Boolean b=(Pattern.matches(REGEX_FAMILY,mToFamily));
                        b=b&&(Pattern.matches(REGEX_MEMBER,mToMember)
```

```
);
                        b=b&&(IsLenOK(mToFamily,MINMAX[0][0],MINMAX[0][1]));
                        b=b&&(IsLenOK(mToMember,MINMAX[1][0],MINMAX[1][1]));
                        //todo test comment versus pattern message
                        if (b) {
                        sendMailAsync sendmail = new sendMailAsync();
                        sendmail.execute();
                        } else {
                            //snack.setText(R.string.mail_send_fail_error);
                            //snack.show();
                            //mDSourceText.setText(getString(R.string.mail_send_fail_error));
                            //mDSourceText.setTextColor(getResources().getColor(R.color.light_red));
                            Toast.makeText(getActivity(),getString(R.string.P4M_err), Toast.LENGTH_SHORT).show();
                        }
                    }
                });
                builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        dialog.cancel();
                    }
                });
                builder.show();
                return true;
            case R.id.recipe_menu_delete:
                CookBook.get(getActivity()).markRecipeToDelete(mRecipe);
                getActivity().onBackPressed();
            default:
                return super.onOptionsItemSelected(item);
        }
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState){
        final View v=inflater.inflate(R.layout.fragment_display_recipe, container, false);
       //snack=Snackbar.make(v, "Message is deleted", Snackbar.LENGTH_LONG);
        mScroll=(ScrollView) v.findViewById(R.id.fragment_recipe_scroll);
        mDTitleText =(TextView) v.findViewById(R.id.recipe_display_title);
        mDPhotoView=(ImageView) v.findViewById(R.id.recipe_display_photo);
```

```
        updatePhotoView();
        mDRatingBar=(RatingBar) v.findViewById(R.id.recipe_display_ratingBar);
        mDRatingBarText=(TextView) v.findViewById(R.id.recipe_display_ratingBar_txt);
        mDDifficulty=(TextView) v.findViewById(R.id.recipe_display_difficulty);
        mSunIcon=(ImageView) v.findViewById(R.id.recipe_img_sun) ;
        mIceIcon=(ImageView) v.findViewById(R.id.recipe_img_ice) ;
        mDAuthorText=(TextView) v.findViewById(R.id.recipe_display_author);
        mDSourceText=(TextView) v.findViewById(R.id.recipe_display_source);
        mDSourceUrl=(TextView) v.findViewById(R.id.recipe_display_source_url);
        mDIngTitle=(TextView) v.findViewById(R.id.recipe_display_title_ing);
        mDComTitle=(TextView)v.findViewById(R.id.recipe_display_comment_title);
        final int[] rIngID= {R.id.recipe_display_I01,R.id.recipe_display_I02,R.id.recipe_display_I03,
                R.id.recipe_display_I04,R.id.recipe_display_I05,R.id.recipe_display_I06,
                R.id.recipe_display_I07,R.id.recipe_display_I08,R.id.recipe_display_I09,
                R.id.recipe_display_I10,R.id.recipe_display_I11,R.id.recipe_display_I12,
                R.id.recipe_display_I13,R.id.recipe_display_I14,R.id.recipe_display_I15};
        mDIngText= new TextView[mRecipe.getNbIngMax()];
        for(int i=0;i<mRecipe.getNbIngMax();i++) {
            mDIngText[i] = (TextView) v.findViewById(rIngID[i]);
        }
        final int[] rID= {R.id.recipe_display_S1,R.id.recipe_display_S2,R.id.recipe_display_S3,
                R.id.recipe_display_S4,R.id.recipe_display_S5,R.id.recipe_display_S6,
                R.id.recipe_display_S7,R.id.recipe_display_S8,R.id.recipe_display_S9};
        mDStepText=new TextView[mRecipe.getNbStepMax()];
        for(int i=0;i<mRecipe.getNbStepMax();i++) {
            mDStepText[i] = (TextView) v.findViewById(rID[i]);
        }
        final int[] rComID= {R.id.recipe_display_C01,R.id.recipe_display_C02,R.id.recipe_display_C03,
                R.id.recipe_display_C04,R.id.recipe_display_C05,R.id.recipe_display_C06,
                R.id.recipe_display_C07,R.id.recipe_display_C08,R.id.recipe_display_C09,
                R.id.recipe_display_C10,R.id.recipe_display_C11,R.id.recipe_display_C12,
                R.id.recipe_display_C13,R.id.recipe_display_C14,R.id.recipe_display_C15,
                R.id.recipe_display_C16,R.id.recipe_display_C17,R.id.
```

```
recipe_display_C18,
                R.id.recipe_display_C19,R.id.recipe_display_C20};
        mDComText=new TextView[mRecipe.getNbComMax()];
        for(int i=0;i<mRecipe.getNbComMax();i++) {
            mDComText[i] = (TextView) v.findViewById(rComID[i]);
        }
        updateUI();
        mDNexComment=(EditText) v.findViewById(R.id.recipe_display_enter_comment);
        mDNexComment.setText("");
        mDNexComment.setHint(getString(R.string.P2C_hint));
        //todo test comment versus pattern
        mDNexComment.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int start, int count, int after) {

            }

            @Override
            public void onTextChanged(CharSequence s, int start, int before, int count) {
                newcomment=s.toString();
            }

            @Override
            public void afterTextChanged(Editable s) {

            }
        });
        mDEnterComment=(ImageView) v.findViewById(R.id.recipe_img_enter);
        mDEnterComment.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                mRecipe.addComment(new Comment(newcomment, mSession.getUser()));
                mRecipe.updateTS(AsynCallFlag.NEWCOMMENT,true);
                mDNexComment.setText("...");
                updateUI();
            }
        });
        return v;
    }
    private void updatePhotoView(){
        if (mPhotoFile==null || !mPhotoFile.exists()){
            mDPhotoView.setImageDrawable(getResources().getDrawable(R.drawable.ic_recipe_camera));
        } else {
            Bitmap bitmap=PictureUtils.getScaledBitmap(mPhotoFile.getPath(), getActivity());
            mDPhotoView.setImageBitmap(bitmap);
        }
```

```
    }
    private void updateUI(){
        String s;
        User u=mRecipe.getOwner();
        int ingMax=mRecipe.getNbIngMax();
        int stepMax=mRecipe.getNbStepMax();
        int comMax=mRecipe.getNbComMax();
        int NbCom=mRecipe.getComments().size();
        int gone=View.GONE;
        int visible=View.VISIBLE;
        DecimalFormat df = new DecimalFormat("#.#");
        mDTitleText.setText(mRecipe.getTitle());
        mDRatingBar.setRating((float) mRecipe.getNoteAvg());
        mDRatingBarText.setText(df.format(mRecipe.getNoteAvg())+"  (" + mRecipe.getNotes().size()+")");
        s=getString(R.string.P2U_txt,u.getName(),u.getFamily());
        int idiff= RecipeDifficulty.getIndex(mRecipe.getDifficulty());
        String[] stringArrayDiff = getResources().getStringArray(R.array.recipe_difficulty_array);
        mDDifficulty.setText(stringArrayDiff[idiff]);
        mSunIcon.setImageResource((mRecipe.IsSummer()) ? R.drawable.ic_recipe_sun : R.drawable.ic_recipe_sun_disabled);
        mIceIcon.setImageResource((mRecipe.IsWinter()) ? R.drawable.ic_recipe_ice : R.drawable.ic_recipe_ice_disabled);
        mDAuthorText.setText(s);
        mDSourceText.setText(mRecipe.getSource());
        if (mRecipe.getSource_url_name().equals(DEFAULT_URL)) {
            mDSourceUrl.setText(""); }
        else {mDSourceUrl.setText(mRecipe.getSource_url_name());}
        s=getString(R.string.P2IT, ""+mRecipe.getNbPers());
        mDIngTitle.setText(s);
        for(int i=0;i<ingMax;i++){
            if (mIngNb>0){mDIngText[i].setText("- "+mRecipe.getIngredient(i+1));}
            if (i>=0){mDIngText[i].setVisibility((mIngNb>i)? visible:gone);}
        }
        for(int i=0;i<stepMax;i++){
            if (mStepNb>0){mDStepText[i].setText((i+1)+". "+mRecipe.getStep(i+1));}
            if (i>=0){mDStepText[i].setVisibility((mStepNb>i)? visible:gone);}
        }
        s=getString(R.string.P2CT, ""+mRecipe.getComments().size());
        mDComTitle.setText(s);
        for(int i=0;i<comMax;i++){
            if (NbCom>0){
                if (i<NbCom) {mDComText[i].setText("- "+mRecipe.getComment(NbCom-i-1).toTxt());}
                else {mDComText[i].setText("");}
            }
            if (i>=0){mDComText[i].setVisibility((NbCom>i)? visible:gone);}
```

```
        }
    }
    private Boolean IsLenOK(String s, int min, int max){
        int l=s.length();
        return ((l>min)&&(l<max));
    }
    private String getRecipeReport(){
        String report;
        Integer iplus;
        report =getString(R.string.P2RE_title, mRecipe.getTitle())+"\n";
        report +=getString(R.string.P2RE_user,mRecipe.getOwner().getNameComplete())+"\n";
        if(!mRecipe.getSource().equals("")){
            report +=  getString(R.string.P2RE_src, mRecipe.getSource())+"\n";
        }
        if(!mRecipe.getSource_url_name().equals("")){
            report +=  getString(R.string.P2RE_url, mRecipe.getSource_url_name())+"\n";
        }
        if (mRecipe.getNbIng()>0){
            report +=getString(R.string.P1R_ing)+"\n";
            for(int i=0;i<mRecipe.getNbIng();i++){
                report += getString(R.string.P2RE_ing, mRecipe.getIngredient(i+1))+"\n";
            }
        }
        if (mRecipe.getNbStep()>0) {
            report +=getString(R.string.P1R_step)+"\n";
            for (int i = 0; i < mRecipe.getNbStep(); i++) {
                iplus = i + 1;
                report += getString(R.string.P2RE_step, iplus + "",
                        mRecipe.getStep(i + 1)) + "\n";
            }
        }
        report +=getString(R.string.P2RE_end);
        return report;
    }

    /**************************************************************************************
     *                                ASYNC                                               *
     **************************************************************************************/


    class sendMailAsync extends AsyncTask<Void, Void, Boolean> {
        private static final String PHP204 = "return204.php";
        private static final String MYSQLDATEFORMAT="yyyy-MM-dd HH:mm:ss";
```

```
        private static final String PHPSENDMAIL = "sendmail.php";
        @Override
        protected void onPreExecute() {
            super.onPreExecute();
            //snack.setText(R.string.mail_send_start);
            //snack.show();
            //mDSourceText.setText(getString(R.string.mail_send_start));
            //mDSourceText.setTextColor(getResources().getColor(R.color.light_red));
            Toast.makeText(getActivity(),getString(R.string.P4_start), Toast.LENGTH_SHORT).show();
        }

        @Override
        protected void onPostExecute(Boolean b) {
            super.onPostExecute(b);
            if (b) {
                //snack.setText(getString(R.string.mail_send_success,mToMember,mToFamily));
                //mDSourceText.setText(getString(R.string.mail_send_success,mToMember,mToFamily));
                //mDSourceText.setTextColor(getResources().getColor(R.color.light_green));
                Toast.makeText(getActivity(), getString(R.string.P4_OK,mToMember,mToFamily), Toast.LENGTH_LONG).show();
            }
            else {
                //snack.setText(getString(R.string.mail_send_fail));
                //mDSourceText.setText(getString(R.string.mail_send_fail));
                Toast.makeText(getActivity(), getString(R.string.P4_fail), Toast.LENGTH_LONG).show();
            }
            //snack.show();
            return;
        }

        @Override
        protected Boolean doInBackground(Void... voids) {
            NetworkUtils mNetUtils = new NetworkUtils(getContext());
            if (!test204()) {
                return false;
            }
            HashMap<String, String> data = new HashMap<>();
            data.put("idrecipe", mRecipe.getId().toString());
            if ((mToFamily==null)||(mToFamily.length()<5)) return false;
            data.put("family",mToFamily.trim());
            if ((mToMember==null)||(mToMember.length()<5)) return false;
            data.put("membre", mToMember.trim());
            data.put("idfrom", mSession.getUser().getId().toString());
```

```
            if (mToMessage==null) return false;
            data.put("message", mToMessage.trim());
            String result = mNetUtils.sendPostRequestJson(mSession.getURLPath() + PHPSENDMAIL, data);
            if (result==null) return false;
            if (!result.trim().equals("1")) {
                //fdx Log(TAG, "Retour de "+PHPSENDMAIL+" =>"+result+"<");
                return false;}
            return true;
        }

        private Boolean test204() {
            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();
                return (status == HttpURLConnection.HTTP_NO_CONTENT);
            } catch (Exception e) {
                //fdx Log(TAG, "Test 204 : " + e);
                return false;
            }
        }

    }

}
```

```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);
        //intent.putExtra(EXTRA_RECIPE_ID, recipeId);
        return intent;
    }

    @Override
    protected Fragment createFragment() {
        //UUID recipeId=(UUID) getIntent().getSerializableExtra(EXTRA_RECIPE_ID);
        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() {
        //Bundle args=new Bundle();
        //args.putSerializable(ARG_RECIPE_ID, recipeId);
        RecipeMailDisplayFragment fragment=new RecipeMailDisplayFragment();
        //fragment.setArguments(args);
        return fragment;
    }

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

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

```java

    @Override
    public void onResume(){
        super.onResume();
        updateUI();
    }

    private void updateUI() {
        CookBook cookbook=CookBook.get(getActivity());
        List<Recipe> recipes_in=cookbook.getRecipes();
        List<Recipe> recipes=new ArrayList<>();
        for(Recipe r:recipes_in){
            if (r.IsMessage()) recipes.add(r);
        }
        if (recipes.isEmpty()){
            Toast.makeText(getContext(), getString(R.string.P4M0), Toast.LENGTH_LONG ).show();
        }
        if (mAdapter==null){
            mAdapter=new RecipeMailDisplayFragment.RecipeAdapter(recipes);
            mRecipeRecyclerView.setAdapter(mAdapter);
        } else {
            mAdapter.setRecipes(recipes);
            mAdapter.notifyDataSetChanged();
        }
    }
// -----------------------------------RECIPE HOLDER ------------------------
    private class RecipeHolder extends RecyclerView.ViewHolder
            implements View.OnClickListener {
        private TextView mTitleTextView;
        private TextView mMessage;
        private TextView mFrom;
        private ImageView mPhotoView;
        private ImageView mDelete;
        private ImageView mAdd;
        private File mPhotoFile;
        private Recipe mRecipe;

        public RecipeHolder(LayoutInflater inflater,ViewGroup parent) {
            super(inflater.inflate(R.layout.list_item_mail_display, parent, false));
            itemView.setOnClickListener(this);
            mTitleTextView= (TextView) itemView.findViewById(R.id.recipe_MD_title);
            mMessage=(TextView) itemView.findViewById(R.id.mail_display_message);
            mFrom=(TextView) itemView.findViewById(R.id.mail_display_author);
            mDelete=(ImageView) itemView.findViewById(R.id.mail_display_delete);
            mAdd=(ImageView) itemView.findViewById(R.id.
```

```
mail_display_add);
            mPhotoView=(ImageView) itemView.findViewById(R.id.recipe_MD_photo);
            mPhotoFile=CookBook.get(getActivity()).getPhotoFile(mRecipe);

            mDelete.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    mRecipe.setStatus(StatusRecipe.Deleted);
                    CookBook cookbook=CookBook.get(getActivity());
                    cookbook.updateRecipe(mRecipe);
                    updateUI();
                }
            });
            mAdd.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    mRecipe.setStatus(StatusRecipe.Visible);
                    CookBook cookbook=CookBook.get(getActivity());
                    cookbook.updateRecipe(mRecipe);
                    updateUI();
                }
            });

        }
        public void bind(Recipe recipe){
            mRecipe=recipe;
            mTitleTextView.setText(mRecipe.getTitle());
            mMessage.setText(mRecipe.getMessage());
            mFrom.setText(mRecipe.getUserFrom().getNameComplete());
            mPhotoFile=CookBook.get(getActivity()).getPhotoFile(mRecipe);
            if (mPhotoFile==null || !mPhotoFile.exists()){
                mPhotoView.setImageDrawable(getResources().getDrawable(R.drawable.ic_recipe_see));
            } else {
                Bitmap bitmap=PictureUtils.getScaledBitmap(mPhotoFile.getPath(), getActivity());
                mPhotoView.setImageBitmap(bitmap);
            }
        }
    @Override
    public void onClick(View v){    }
    }
    // -----------------------------------ADAPTER------------------------
    private class RecipeAdapter extends RecyclerView.Adapter<RecipeMailDisplayFragment.RecipeHolder> {
        private List<Recipe> mRecipes;
        public RecipeAdapter(List<Recipe> recipes){
            mRecipes=recipes;
        }

```

```
        @NonNull
        @Override
        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;
        }
    }

}

```