GoToS3
C2L3Play

User Manual
Review App & Review Manager
English Version, 10/07/2020

Table of contents

Installing the Unity server

This Windows program allows to manage different projects, to generate and manipulate 3D spaces from building plans and offers tools to collect and manage "Reviews" collections.

1. Parameters and data storage

Each project is saved on the computer's hard drive in a folder with the same name as the project, and with a precise hierarchy. In each project folder you will find:

- A *simulation.xml* file containing project information (project name, session key, floor list). Although it is possible to edit this file, modifying it can lead to a crash of the application the next time the project is loaded and is therefore not recommended.
- A *Floors* folder containing a subfolder for each of the floors created. In each floor’s subfolder, you will find:
  a) Two images: *FloorPlan.png* is a copy of the image file used to generate the floor and *Heatmap.png* is a representation of the density of reviews on the plan with a transparency of 32%. These two images have the same resolution and can be easily overlaid with image processing software.
  b) A *Notes* folder containing the reviews received for this floor. Each review consists of two jpeg images (the original received image and a smaller version whose name ends with *_light.jpg* which is used in the server application) and a JSON file containing the other data associated with the review.

2. Launching the program

- Unzip the zip archive to the location of your choice
- In the *ReviewManager* folder, double-click on *ReviewManager.exe*
- If a window concerning the firewall appears, check the box "private network" and possibly "public network" if the program is to be used on this type of network and then validate.
- In the *ReviewManager* window, select the desired display settings (recommended settings: 1920x1080, Low, Display1) and then click on "Play! »
- To create a shortcut, right-click on *ReviewManager.exe,* then "send to" -> "Desktop (create a shortcut)".

3. Licenses

The program was created on the Unity game engine (version 2018.4). As such, it cannot be sold without a paid Unity license. Its use remains free.

The code of the program is made available in Open Source, under license GPLv3.

In addition to the .NET framework, two plugins and three Open Source scripts were used.

- Plugin: UnitySimpleFileBrowser
  File Browsing Utility for the User Interface
  https://github.com/yasirkula/UnitySimpleFileBrowser
  v1.1 (October 2019)
  MIT License

- Plugin: websocket-sharp
  Necessary for the functioning of the websocket server
  websocket-sharp.dll, compiled from https://github.com/sta/websocket-sharp
  Commit 46f464910621054d6d68df0d5610b5409de9b580 (17 March 2020)
  MIT License

- Script: TextureScale.cs
  Script for resizing textures and images
  http://wiki.unity3d.com/index.php?title=TextureScale&oldid=19637
  October 14, 2016 version
  Published under Creative Commons Attribution Share Alike license (CC BY-SA 3.0)

- Script: TextureFilter.cs
  Library of image processing functions
  https://github.com/andrewkirillov/AForge.NET
  rev 1732
  LGPL v3

- Script: MeshGenerator.cs
  Generation of meshes from a texture
  https://github.com/SebLague/Procedural-Cave-Generation
  Commit 12f0b07d6f273e4dd12db937f939482b03eb69b4 (19 April 2015)
  MIT License

4. Languages

The application was developed exclusively in English. As it is intended for use by an informed person, multilingual support was not considered. A translation would require modifying the interface (Unity project), as well as strings in the code files (C#).

Installing the Android application

The code runs on Android. It has been developed to run normally from Android 5.0 (API 21, Lollipop, 2014) to Android 10.0 (API 29, Q, 2020).

To function properly, the application requires access to an internet connection, as well as access to the camera and the photo gallery.

The display has been designed and built from a Samsung A20e (6''4, 720x1560 resolution) under Android 10. The development was only done for a version of the application in portrait mode (the change of orientation was blocked in the application). It is possible that the display may differ significantly depending on the device used and its dimensions.

1. Parameters and data storage

Photos taken and downloaded can be retrieved by connecting the phone to a computer and browsing the data part of the application: "Phone/Android/data/com.numediart.reviewapp/files".

The downloaded floors are available in the Download folder, sorted by session key. Reviews pictures are available in the Pictures folder, each name corresponding to the Review id. In case the user has used a photo from his gallery for a Review, it will be duplicated in the application folder.

Be careful however not to delete these different files or folders because access to them for the application is governed by a database.

If the user wishes to free up memory space, he is prompted to uninstall the application or "delete data" in the storage settings. Be careful though, this will delete all data, including downloaded images and Reviews that have not yet been sent to the server.

2. Launching the program

There are two compiled versions of the application (APK), a debug version and a release version. The first version allows access to the "Generate Reviews" button in order to check the load the server is able to support. The second version does not have this button. In both cases, the installation process is the same:

- Connect the phone to a computer via the USB port.
- On the phone, a message appears asking if you want to authorize access to the phone data, press "authorize".
- On the computer, open the file explorer (e.g. via the "This PC" icon) and access the phone's folders. Click on the phone icon, then click on "Phone" or an equivalent name. When you get to several folders, choose the "Download" folder and, once inside, copy the APK to it.

- At this point, you can already "eject the phone" and disconnect it from the USB cable.
- On the phone, look for the file explorer. It can sometimes be hidden in subfolders and should have a name like "My Files". Once in this application, look for the file you just transferred (via "recent files" or "install files" for example).
- Press the file to start the installation.
- You may see an alert message indicating that the installation of unknown applications is not allowed, press settings and select the "Authorization from this source" check box. (N.B. once the installation is complete, it is entirely possible to uncheck this option again for your security).
- A message asking if you want to install should appear, press "Install".
- A "Play Protect" warning should appear, press "Install anyway".
- A message asking to "Have the application scanned? "should appear, choose one of the two options.
- Once the installation is complete, return to the application collection view and look for the "Review App" application (or "Review App Debug" for the Debug version).

3. Licenses

The code was developed in Open Source, in GPL v3, the libraries used are based on compatible licenses.

Apart from libraries belonging to Android SDK, the following libraries were used:

- Android Debug Database
  https://github.com/amitshekhariitbhu/Android-Debug-Database
  V1.0.3- Apache 2.0

- GSON
  https://github.com/google/gson
  V2.8.6 - Apache 2.0

- Android Websocket client
  https://github.com/AlexanderShirokih/java-android-websocket-client
  V1.2.2 - Apache 2.0
  (Compiled version of Alexander Shirokih's fork to complement the original Gusavila92 code: https://github.com/gusavila92/java-android-websocket-client as of January 11, 2020, also in Apache 2.0)

The Android Debug Database is only present for debugging purposes and it may be considered to remove it in a final version of the application.

4. Languages

The application was developed exclusively in English. All the texts appearing to the user are gathered in the file "res/values/strings.xml" and it is possible, by translating these terms, to make the application multilingual.

Glossary

1. Review

Also called "Note" in the code part of the server, it is, as a whole and with all its components (photo, location, score, title, explanations, ...), as the user has saved it in the Android application and as it will be sent to the server.

2. Floor

Plan hosted on the server for each project, allowing the localization of each note, thanks to coordinates relative to the image of the plan.

3. Session

Defined by the session key, it allows a project to be uniquely identified, regardless of the username, IP address and port of the server.

Using the Unity server

1. General principle

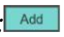
This Windows program allows to manage different projects, to generate and manipulate 3D spaces from building plans and offers tools to collect and manage "Reviews" collections.

2. User interface

The main components of the user interface are buttons and text input fields. They are always rectangular in shape, with a light gray background color for the buttons and white for the text fields. The component that has the focus (when the mouse passes over or a text field that is being modified) takes on a cyan color. An inactive component that cannot be interacted with takes a dark gray color. Text fields are almost always accompanied by a label that specifies the expected content.

Button active :

Active text input fields :

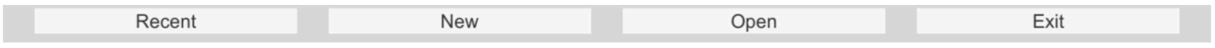
Button with focus :

Button inactive :

Other interface components are also used (cursors on images, check boxes) but their use is specific and will be detailed later in this document.

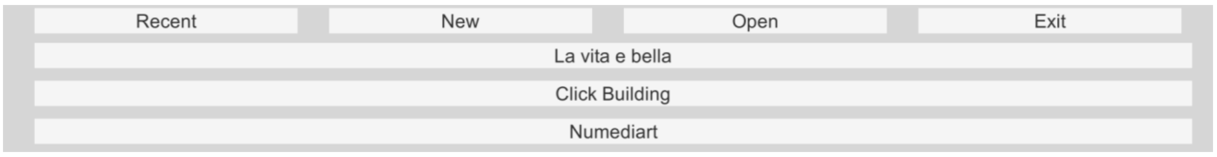
3. Title screen

The first time you start the application, you are taken to the project management screen. It consists of 4 tabs.



a. Recent

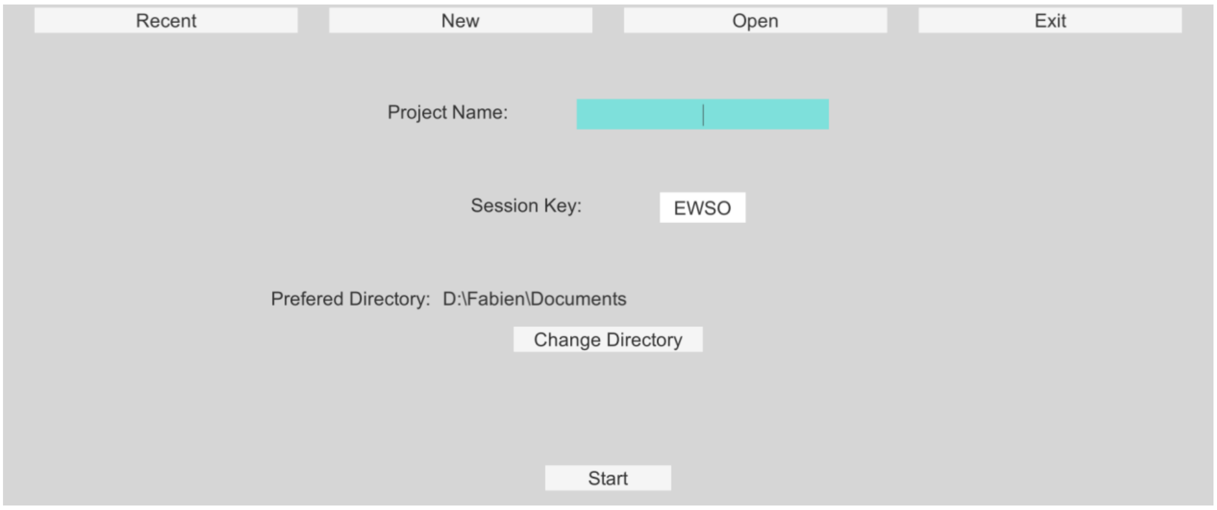
This is the tab displayed when the application is launched. It displays recently loaded projects. To open a recent project, simply click on its name. To remove a project from this list, you must move it or rename it on the hard disk via Windows Explorer and restart the application.



b. New

This tab allows you to create a new project. To do this, you have the following fields:

1. A name must be entered in the *Project Name* text field. It is not possible to give a new project the same name as an already existing project (displayed in the *Recent* tab), in this case an information message will be displayed at the bottom of the screen.
2. Choose a unique session key in the *Session Key* field. This key will be used by the Android application to connect to the computer. The field is initialized with a random string of four uppercase letters. However, you can change it to use lowercase letters or numbers, but the length of the string is limited to four characters.
3. By default, the project will be created in the "My Documents" folder but you can choose another folder and click on the *Change Directory* button.
4. *The Start* button is used to validate the form and to create the project folder on the hard disk.



Project Name:

Session Key: EWSO

Prefered Directory: D:\Fabien\Documents

c. Open

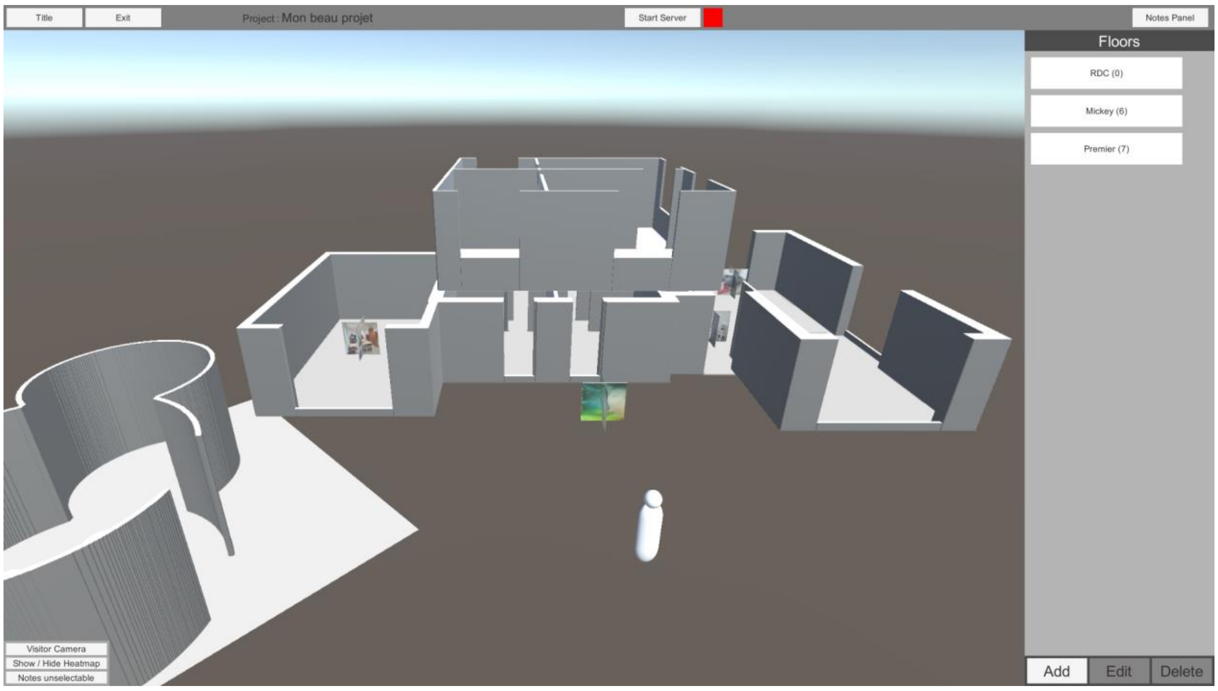
This button opens a dialog window that allows you to navigate to the *simulation.xml* file of the project you want to open.

d. Exit

This button closes the application.

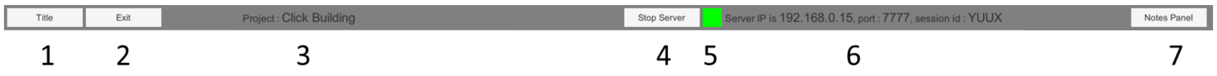
4. Project screen

This screen consists of 3 interaction areas.



a. Menu bar

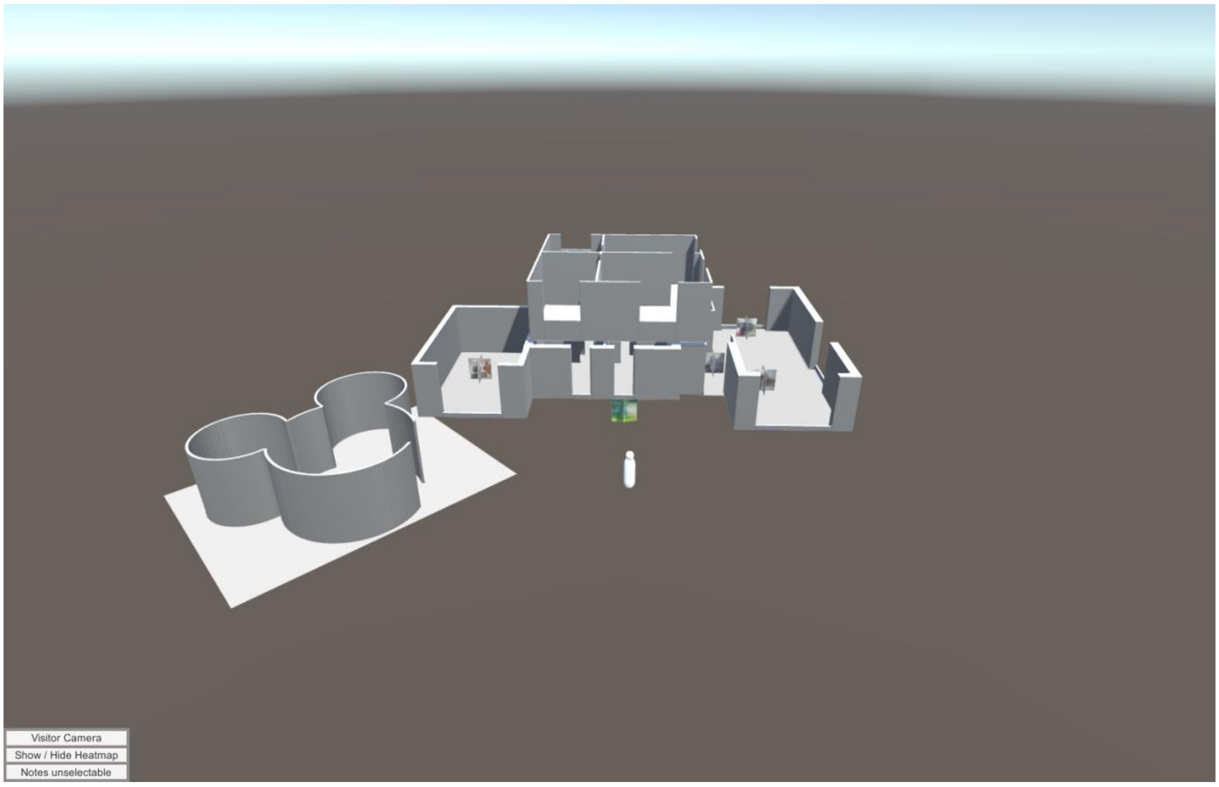
The menu bar is located at the top of the application window. It runs from left to right:



1. *Title Button: Back to Home screen*
2. *Exit button: closes the application (also possible by pressing the escape key on the keyboard)*
3. *Project: the name of the currently open project*
4. *A Start/Stop Server button: Start/stop the websocket server. The server can only communicate with the Android application and thus transmits floor plans and receives reviews when it is started.*
5. *A red light when the server is stopped, green when it is started.*
6. *Server information (IP address, port, session key), to be entered on the Android application home screen to connect to the server and send reviews.*
7. *Notes Panel button: Scrolls up / down the notes panel. See section Panel Notes for more information.*

b. 3D view

The 3D view takes up most of the screen. It allows you to display the 3D model of the building that was generated from the plans of each floor, to move around in this model and to display the images of the reviews where they were located on the plan.



3D view control buttons

At the bottom left of the 3D view there are 3 buttons.

- *Visitor / Omniscient Camera: allows to switch between omniscient camera control and visitor camera control (see next sub-section for more details).*
- *Show / Hide Heatmap: choose between displaying the ground as defined by the plan image and displaying the density map of the reviews.*
- *Notes unselectable / selectable: choose whether a click on one of the notes in the 3D view will open the details panel on this note. The currently pointed note is highlighted in yellow.*

Camera Interactions

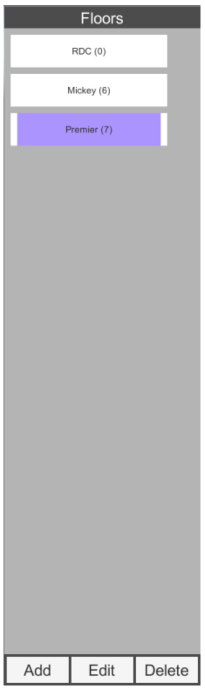
There are 2 cameras in the 3D view.

1) The omniscient camera (selected by default when opening a project) is represented by a white sphere in 3D space and gives control over the 3D view with the mouse.
   - The left mouse click controls the left / right and forward / reverse travel.
   - Right click and hold controls the left / right and up / down translation.
   - The thumbwheel allows you to zoom in or zoom out
   - Click and hold on the thumbwheel to control camera rotation.
2) The visitor camera is represented by a white cylinder surmounted by a white sphere (visible on the screenshot above) and allows you to move in 3D space thanks to the keyboard keys.
   - z' or up arrow: forward
   - s' or down arrow: backward
   - q' or left arrow: rotate to the left

- from' or 'right arrow: rotate to the right
- Shift' or 'PageUp': go up
- Control' or 'PageDown': go down

c. Side panel

Floors



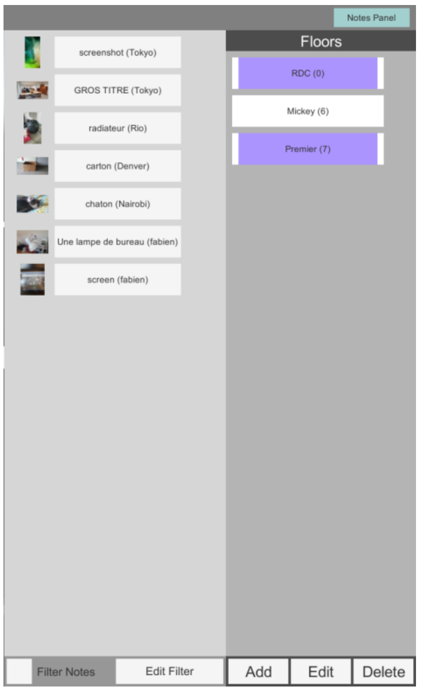
This panel displays a list of the floors created in the project, with for each one the name that was assigned to it when it was created, as well as a number assigned automatically by the program. Each floor can be selected by clicking on it (only one floor selected at a time, except when the *Notes Panel* is deployed). A selected floor is colored with purple.

Below the list of floors there are three buttons.

Add: Opens the floor creation screen. See section Add Floor.
Edit: Unfolds the floor editing panel. See the following of this section for more details (inactive if no floor is selected).
Delete: Opens a dialog window to confirm that you want to delete this floor (inactive if no floor is selected).

WARNING: Deleting a floor deletes its directory on the hard disk, which includes ALL the reviews associated with it.

Panel Notes



Clicking the *Notes Panel* button on the menu bar allows you to expand or collapse this panel. It displays the reviews of all the currently selected floors, with a thumbnail of the photo, the title given to it and the author in brackets.

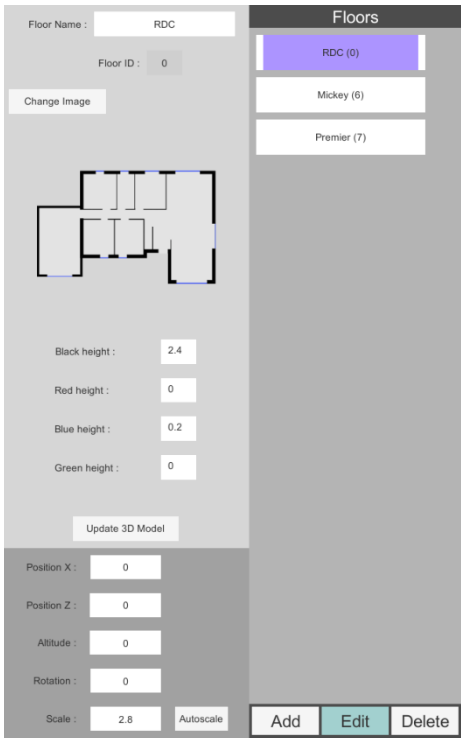
To select multiple floors, hold down the Control key on the keyboard and click on the floors you wish to select. Clicking on a selected floor again deselects it.

It is possible to display a review and all related information by clicking on its title. See section Note Details for more details.

At the bottom of the panel, there is a check box and a button.
Filter Notes: If the box is checked, the filter is applied and only the reviews that match this filter are displayed in the list and in the 3D view. If the box is unchecked, all reviews of the selected floors are displayed.
Edit Filter: Opens the edit screen for filtering reviews. See also section 5.d

Edit Panel

When a floor is selected, clicking the *Edit* button expands or collapses that panel. If the *Notes Panel* was open, *Edit Panel* replaces it and vice versa.



Floor Name : RDC
Floor ID : 0
Black height : 2.4
Red height : 0
Blue height : 0.2
Green height : 0
Position X : 0
Position Z : 0
Altitude : 0
Rotation : 0
Scale : 2.8

This panel is divided into two parts.

The top one allows you to modify the values given when creating the panel. To take into account the modifications made on this part, click on the *Update 3D Model* button, otherwise these modifications will be lost when the panel is closed.

Floor Name: Text input field to change the floor name
Floor ID: (cannot be changed) identifier of the floor for the program
Change Image: Button that opens a dialog window to change the image of the floor plan.

WARNING: if the new image has a different width/height ratio than the previous one, the reviews will not be in the right place. Also, if the image is modified while android applications are connected to the server, they will not have the right version of the map.

Black / Red / Blue / Green height: Height of walls and other elements in the 3D model depending on the color in the floor image
Update 3D Model: Validate changes, save project data and regenerate the 3D model of the floor

The second part contains the controls for changing the position and scale of the 3D model. Changes made on this part are immediately taken into account in the 3D view and saved.

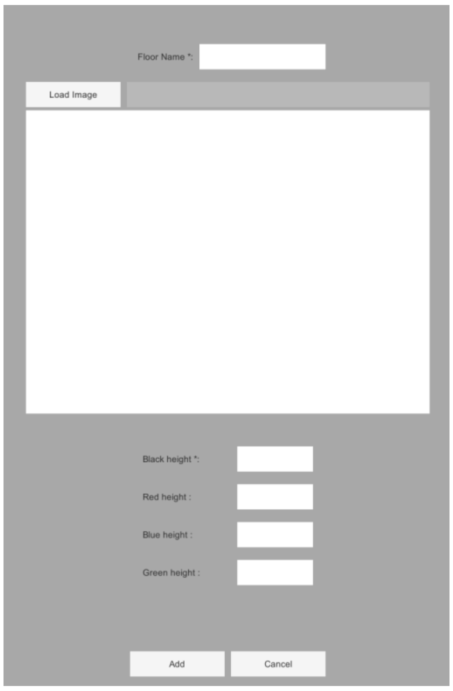
X / Z position: Translation of the 3D model on the left/right or front/rear axis with respect to the origin. At its creation, the model is centered in X = 0, Z = 0.
Altitude: Height at which the base of the 3D model must be located.
Rotation: Rotate the model around its center, value in degrees
Scale and Autoscale button: Multiplying scale factor to be applied to the X and Z dimensions. For more simplicity, a utility allows to calculate this factor automatically. It opens by clicking on the Autoscale button. See section 5.b for details.

5. Special Screens

In addition to this Project screen, some controls bring up task-specific screens. In addition to the confirmation screens, which are self-explanatory, the operation of some screens is described below.

a. Add Floor

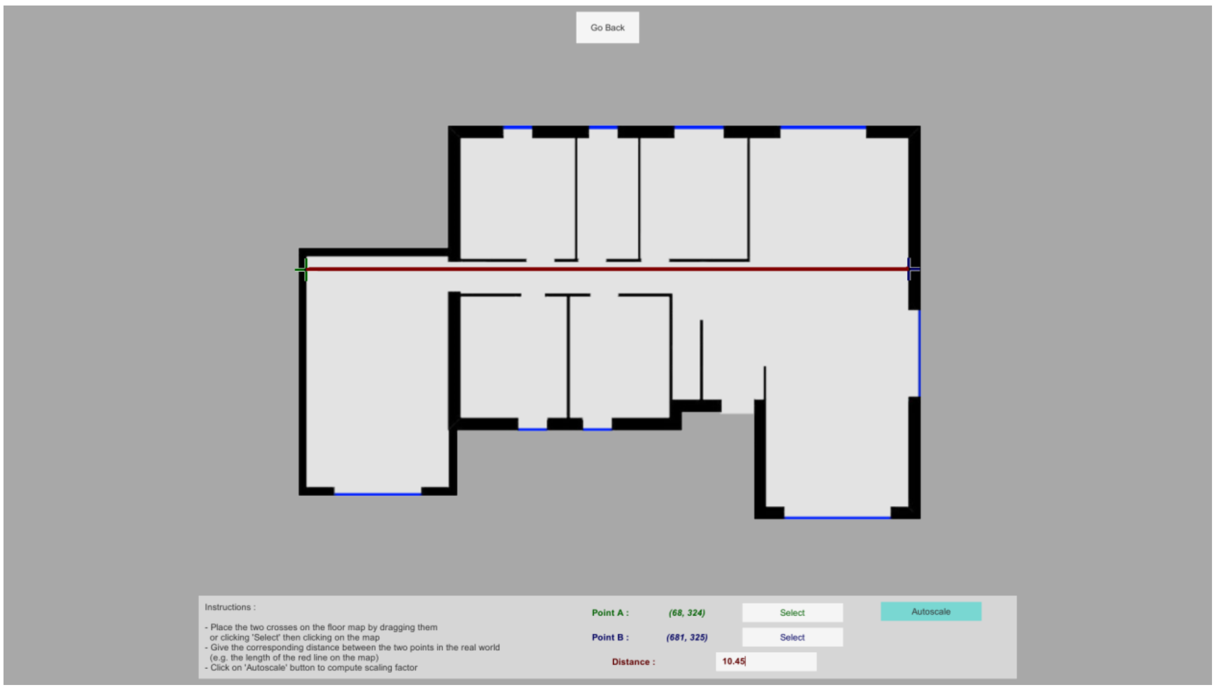
On this form, you will find the same information to give as on the first part of the Edit *Panel*.



Floor Name: *Text input field to give a name to the new floor*
Load Image: *Button that opens a dialog window to choose the floor image. For best practices on how to create this image, see section 6.a*
Black / Red / Blue / Green height: *Height of walls and other elements in the 3D model depending on the colour in the floor image*
Add: *Validate the changes, save the project data and generate the 3D model of the floor*
Cancel: *Cancel the changes and return to the Project screen.*

b. Floor Autoscale

This tool is used to calculate the scale ratio to be applied to the 3D model of the floor so that its representation in the 3D space is "natural".



Instructions :

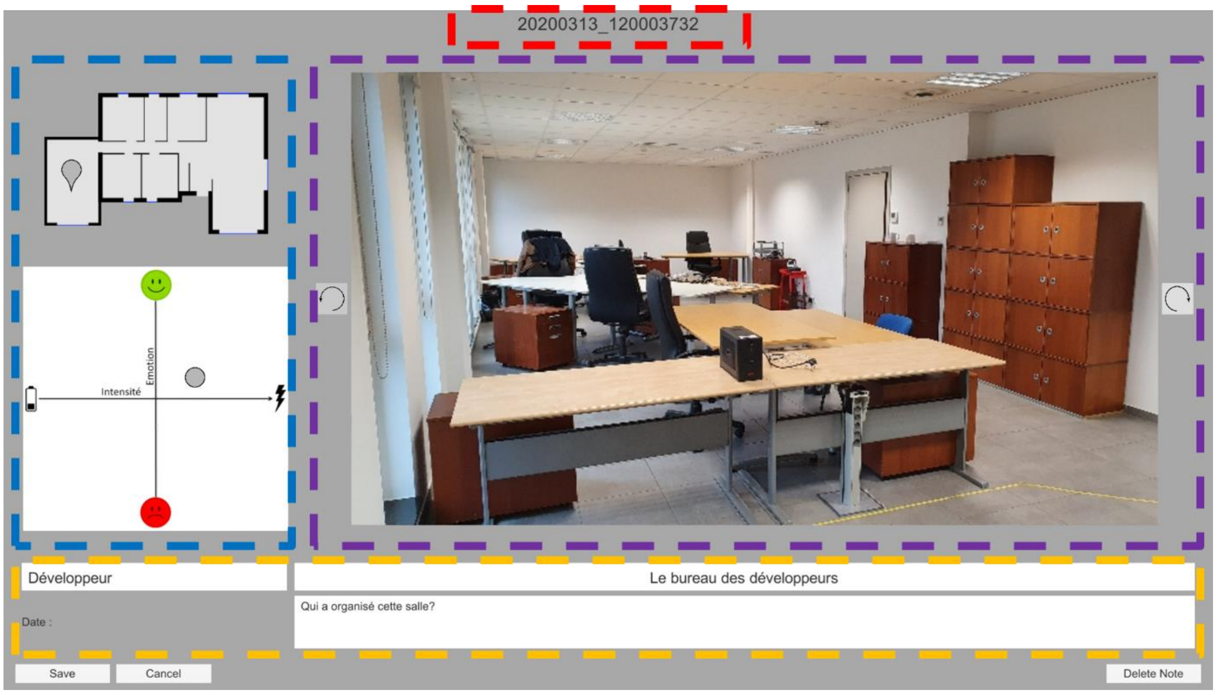
- Place the two crosses on the floor map by dragging them
  or clicking 'Select' then clicking on the map
- Give the corresponding distance between the two points in the real world
  (e.g. the length of the red line on the map)
- Click on 'Autoscale' button to compute scaling factor

Point A : (68, 324)
Point B : (681, 325)
Distance : 10.45

Go Back: *Undo the changes and return to the Project screen*
Instructions: *reminder of the instructions for using this tool*
Point A / B: *Position of points A and B in pixels on the plan image, relative to the bottom left corner. The sliders can be moved by clicking on them and dragging the mouse to the desired position or by using the Select buttons.*
Select: *Transform the mouse cursor into a cross of the color of the point to be selected. Click on the image to place this point.*
Distance: *Spacing between points in real space - length of the red line (in meters for example).*
Autoscale: *Calculate the scale factor and return to the Project screen.*

c. Note Details

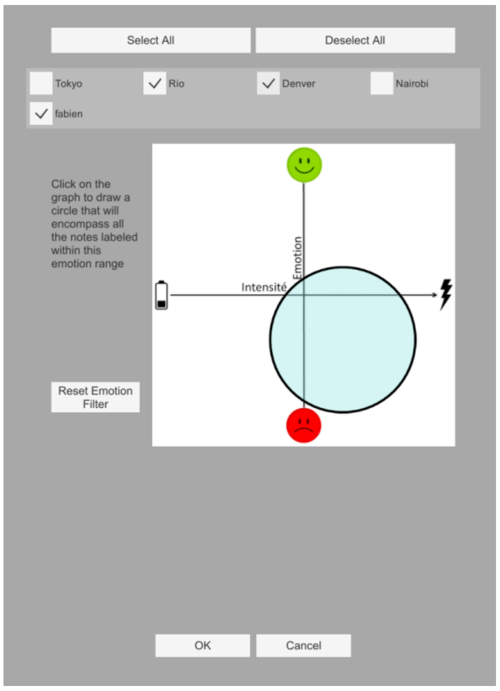
This screen shows the information associated with a review and allows you to modify it. The functions of the different fields on this screen are described below.



***In red**: Name of the JSON file associated with the review. This is the date the review was received by the server.*
***In blue**: Representation of the location of the review in relation to the floor plan and representation of the emotion cursor on the valence/intensity diagram. These two sliders can be moved by clicking on them and dragging them to the desired location.*
***In purple**: Review image presentation area. This image is in low definition to spare the computer memory. The original version is stored on the hard disk. Two buttons with arrows allow you to rotate the image by 90 degrees if necessary.*
***In orange**: The text fields associated with the review.*
At the top left of the area is the Author field.
At the top right, the Title field.
At the bottom left, the Date field (creation date of the note on the Android application), which cannot be modified.
In the bottom right corner, the descriptive text of the review.
Other buttons:
***Save:** Save the changes and return to the Project screen.*
***Cancel:** Ignore the changes and return to the Project screen.*
***Delete:** Delete the review and the associated image.*

d. Note Filter Edition

This screen allows you to customize the filter applied to reviews in the *Notes Panel* if the *Filter Notes check* box is checked.



Select All
Deselect All
Tokyo
Rio
Denver
Nairobi
fabien
Click on the graph to draw a circle that will encompass all the notes labeled within this emotion range
Emotion
Intensité
Reset Emotion Filter
OK
Cancel

The upper part allows to filter according to the authors.

Select All: *select all authors*
Deselect All: *do not select any author*
List of authors with checkboxes: *specifically select one or more authors. If no author is selected, the filtering on the authors does not apply and the result is the same as if they were all selected.*

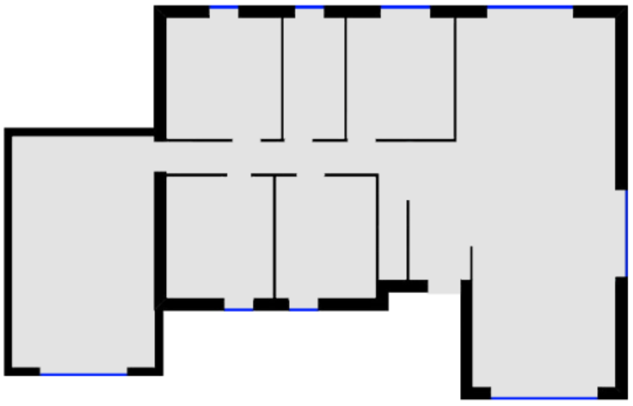
The central part allows you to filter according to the emotion associated with the review. To do this, click in the center of the emotion area to be selected and then hold down the mouse button while dragging to adjust the size of the circle. All the reviews whose emotion cursor is in this circle will be kept. To not use this filter, click on the *Reset Emotion Filter* button.

OK: *validate the filter and return to the Project screen.*
Cancel: *Ignore the changes made to the filter and return to the Project screen.*

6. Helpers

a. Create / modify a floor generation image

Here is an example of a suitable plan for 3D model generation. This process is called extrusion.



The image must be in PNG or JPEG format. The PNG format is preferable because it allows a finer management of the transparency (alpha channel) and does not introduce compression artifacts when saving.

The image resolution should not be too high to limit memory usage and the computation time needed to generate the 3D model. Typically, the largest dimension of the image should always be less than 1200 pixels.

The background of the image must be white or light grey and without inscriptions, it corresponds to areas that will not be extruded. It is possible to divert the building and leave the exterior transparent (transparency must be applied on white - #00FFFFFF). This option is practical for the upper levels and makes it possible to create floors that align perfectly.

Up to 4 different extrusion heights can be created using color coding. Walls are in plain black #0000. The other usable colors are red #FF0000, blue #0000FF and green #00FF00. They can be used for example to represent windows (in blue in the example image) or fixed furniture. Extruded parts of the image are not editable in the 3D view.

To convert an architect's plan into a usable image for the program, it will be necessary to use image processing software and work with layers.

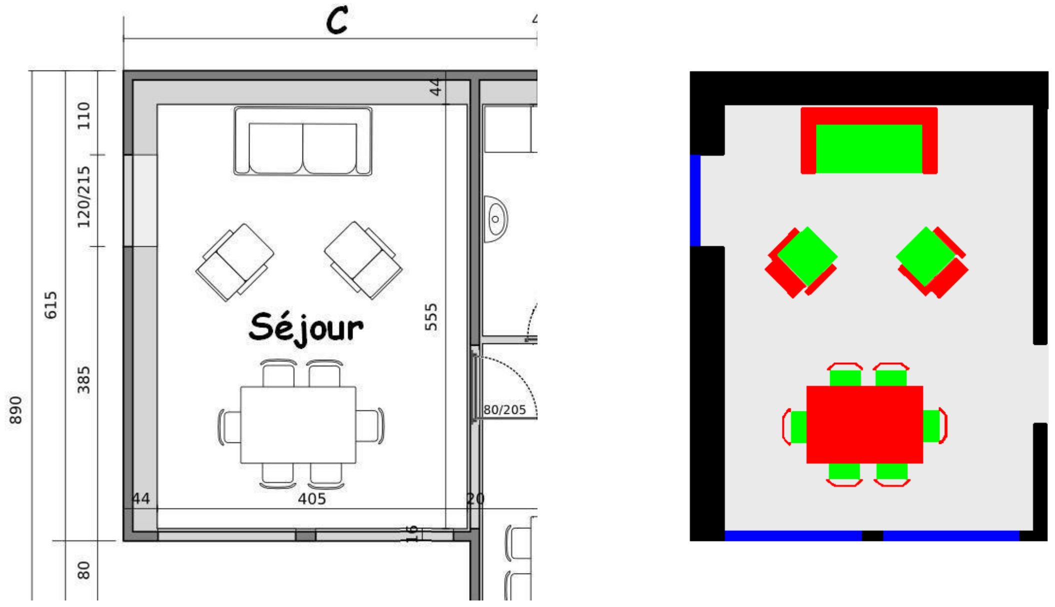
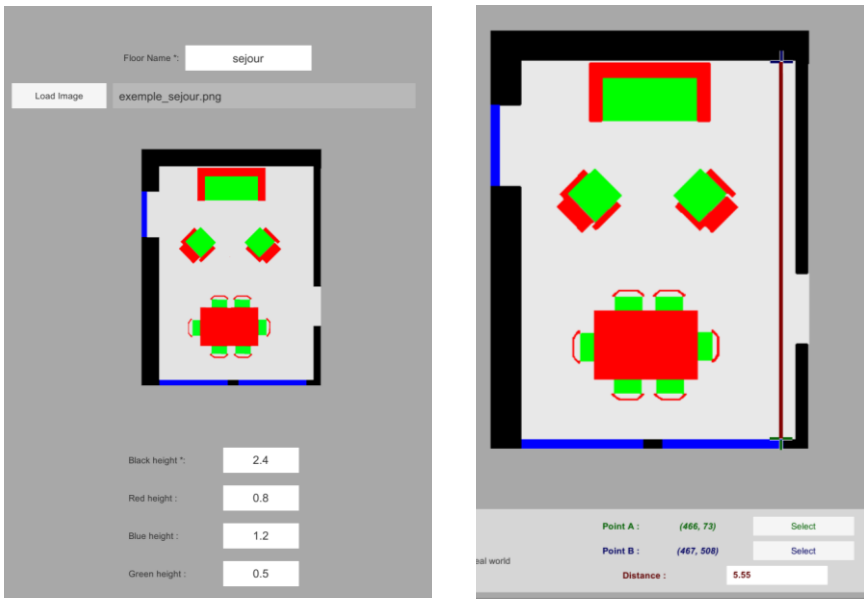
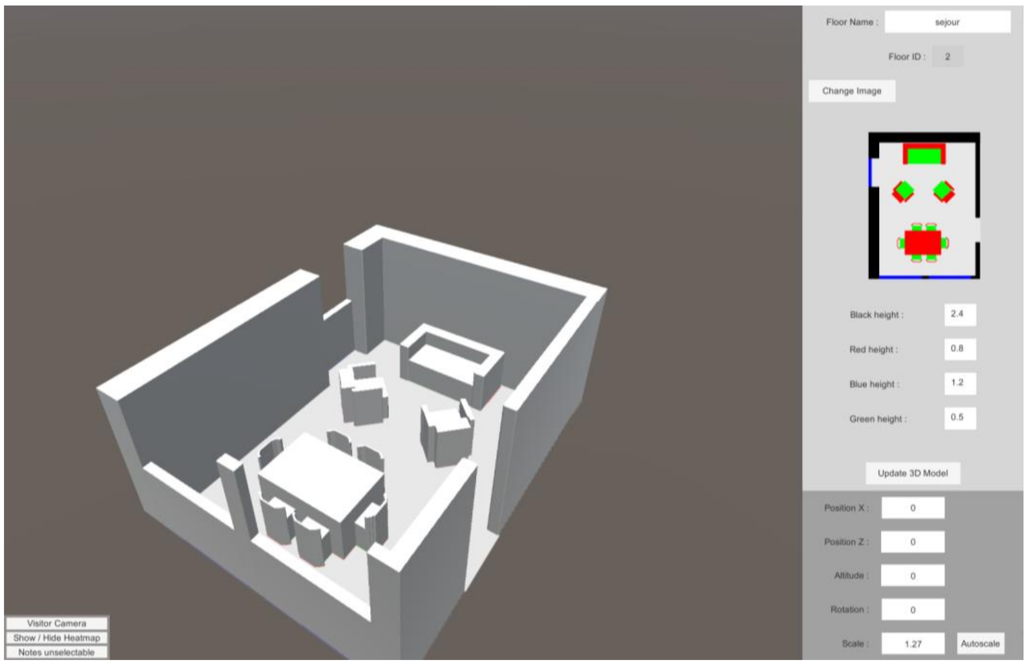


Figure 1 - Example of a plan adapted for extrusion with conservation of fixed furniture



Floor Name *: sejour
exemple_sejour.png
Black height *: 2.4
Red height : 0.8
Blue height : 1.2
Green height : 0.5
Point A : (466, 73)
Point B : (467, 508)
Distance : 5.55

Figure 2 - Creating the floor and scaling with the Autoscale tool



Floor Name : sejour
Floor ID : 2
Black height : 2.4
Red height : 0.8
Blue height : 1.2
Green height : 0.5
Position X : 0
Position Z : 0
Altitude : 0
Rotation : 0
Scale : 1.27

Figure 3 - Result in the 3D view

b. Form navigation

In all forms, it is possible to move to the next field by using the Tab key. The color of the field that has the focus is cyan.

In some forms, fields are required to be filled in. They are usually marked with an asterisk *. If one of these fields is not filled in and the user tries to validate the form, an information message appears, and the field(s) concerned turn(s) red.

c. Length units

The unit of length chosen to define the scale of the model or the height of the walls is not important in itself (it can be metres, centimetres, feet, cubits or other), but it must always be the same on the project. If it is not, the 3D models will be distorted. It is advisable to use meters.

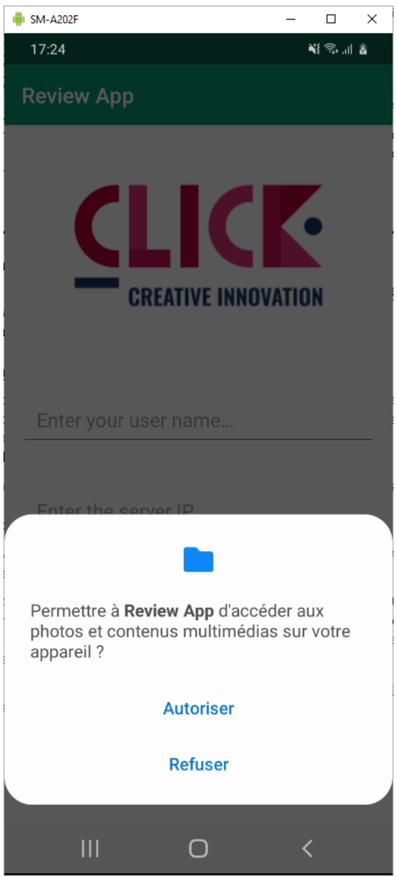
Using the Android application

1. General Principle

This application allows you to connect to a server in order to send it localized Reviews. The user can create one Review at a time, read back the Reviews he has already saved and upload them to the server.

2. Loading screen

The first time you start the application, you will be taken to the loading screen. On the loading screen, you are asked to allow access to photos and multimedia content. This is to be able to save photos but also to be able to use the photos in the gallery to create "Reviews".



Permettre à **Review App** d'accéder aux photos et contenus multimédias sur votre appareil ?

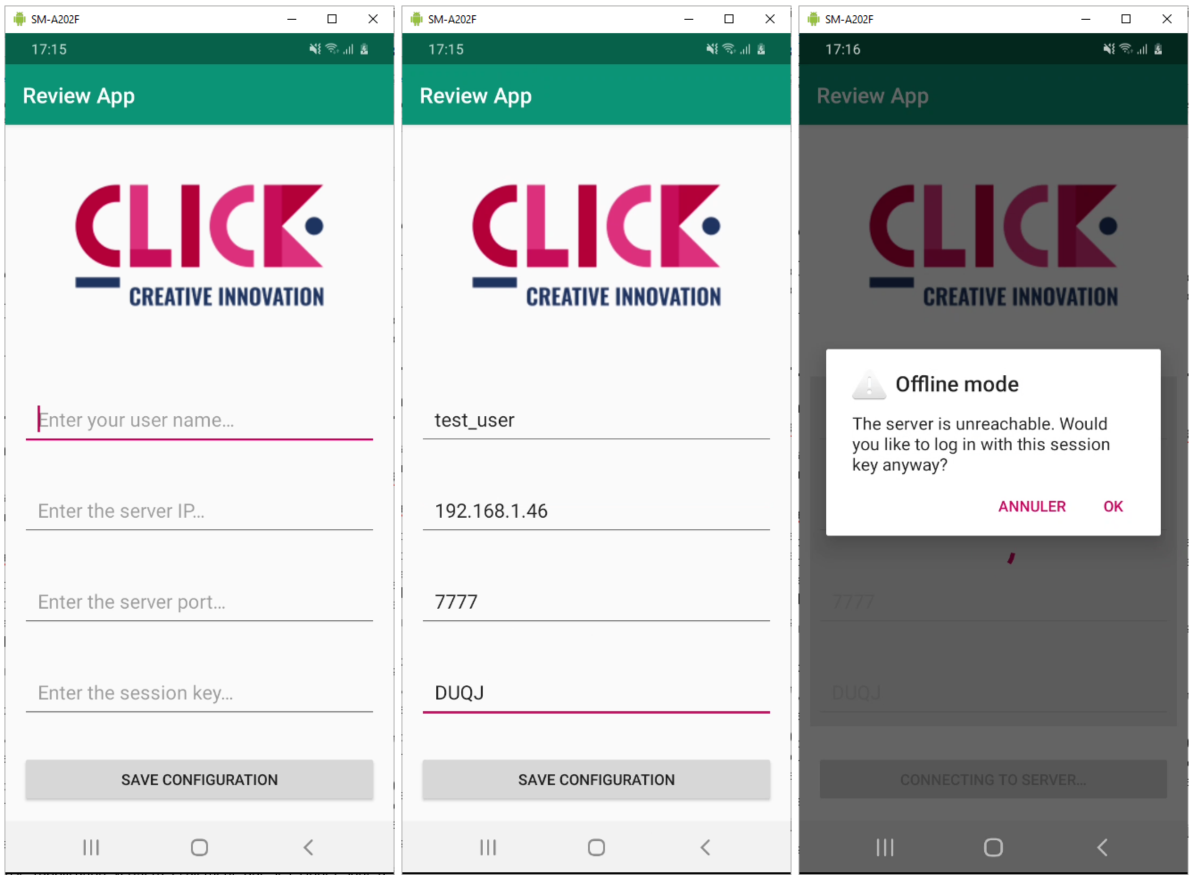
Figure 4 - First start-up

On this first screen, you are asked to enter a username, the IP address of the server and the port it uses. Finally, you must enter the session key linked to the project (see the "server" part).

In case of an error in the entered parameters, a message will be returned to the user. If the application manages to contact the server, it will switch to the "Main screen".

It is also possible to use the application in "offline" mode. This mode is designed to be able to enter an already existing session (for which the "Floors" have already been downloaded) and continue to create Reviews, even if the server is currently out of reach of the application.

At each log in, the application will also check that the Floors are up to date and re-download new versions.



test_user

192.168.1.46

7777

DUQJ

The server is unreachable. Would you like to log in with this session key anyway?

Figure 5 - Loading Screens

3. Main screen

When arriving on the Main screen, the application will first update the Floors (or download them for the first time). Once this is done, you will be able to perform various actions.

When you first connect, the "Upload reviews" button will not be available, but it will become available when Reviews have been saved and can be sent to the server. Similarly, the "See all reviews" button will take you to a screen that will summarize all Reviews saved for the session but is currently empty. Finally, the "Add a review" button allows you to create your first review. If you were previously saving a Review, you can finish editing it using the same button.

It is possible to log out of this session by using the back button. You will then be asked to confirm your disconnection.

N.B.: The top right button "Generate Reviews" is only available when the variable DEBUG is used in the code. It is only used to automatically generate Reviews per packet, for test purposes.

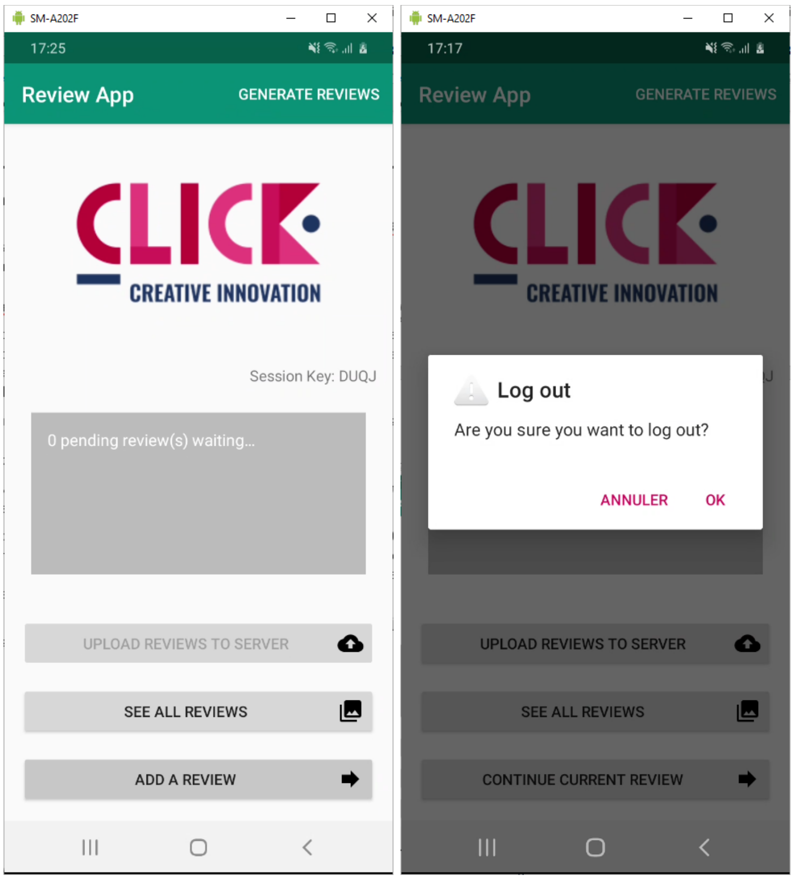


Figure 6 - Main screen

4. Add a review

By clicking on the "Add a review" or "Continue current review" button, you will always go through the same screens in the same order. At each step, the different information is saved but the Review will only be validated and can only be sent by validating each step, which will eventually bring you back to the Main screen.

a. Location on a map

First you get to the screen that allows you to locate the Review. By simply clicking on the map, you can position a point indicating where you are. By "pinching" with two fingers, you can zoom in and out of the map in question, as well as move it by dragging a single finger.

In case there are several Floors available, you can navigate through them with the directional arrows, to choose the map on which you want to locate the Review.

Once correctly positioned, you can click on "Take a picture" to proceed on next screen.

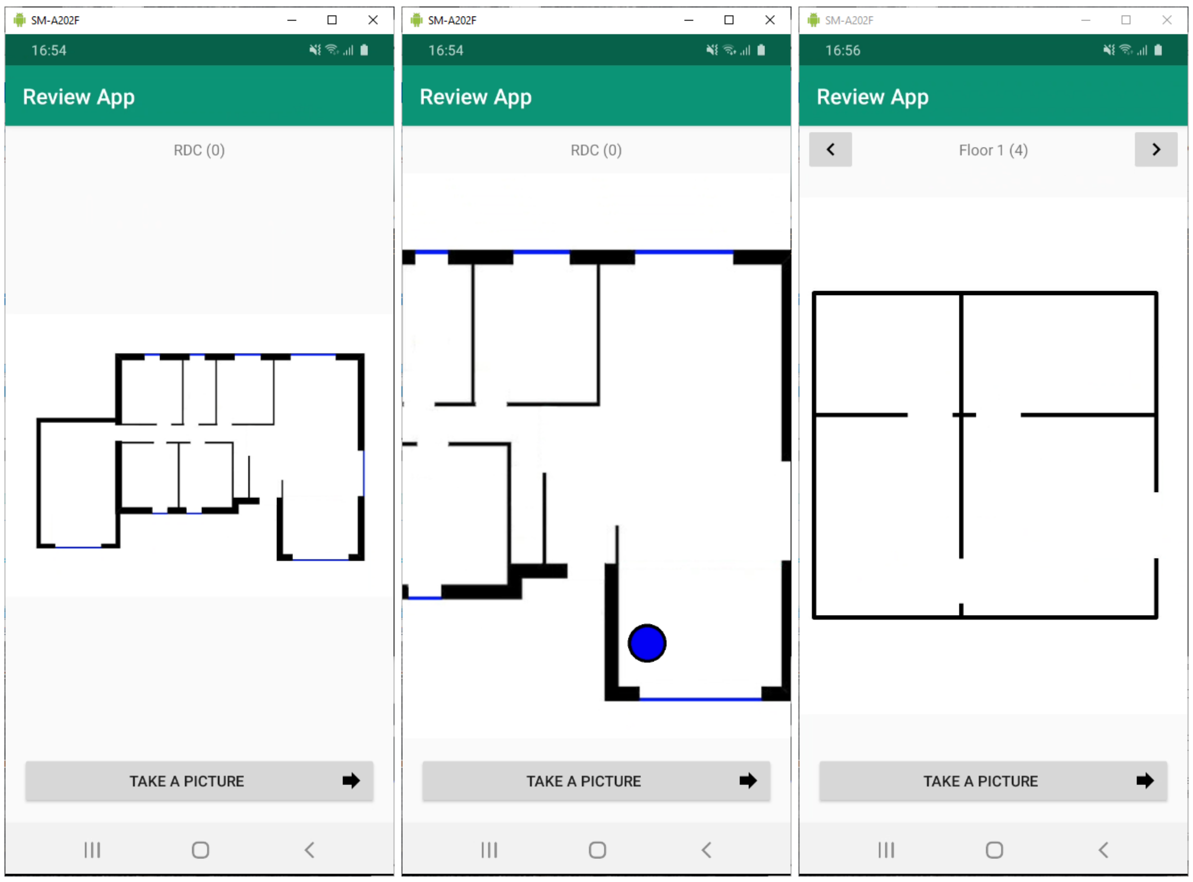


Figure 7 - Locate Screen

b. Photo

This part opens the camera directly to you. Once your photo has been taken, you will be asked to confirm it and then you will be taken to the management screen. At this stage, you can select another photo from the gallery or take another picture.

When the camera opens and if you do not want to take a picture (for example because the previous picture was better), the back button will send you to the management screen. This will eventually allow you to select a photo from your gallery. A second backspace will take you back to the location screen.

Before you can go any further, it will in any case be necessary to provide a photo to the application so that it can save it.

The "Set your mood about it" button will send you to the next screen.

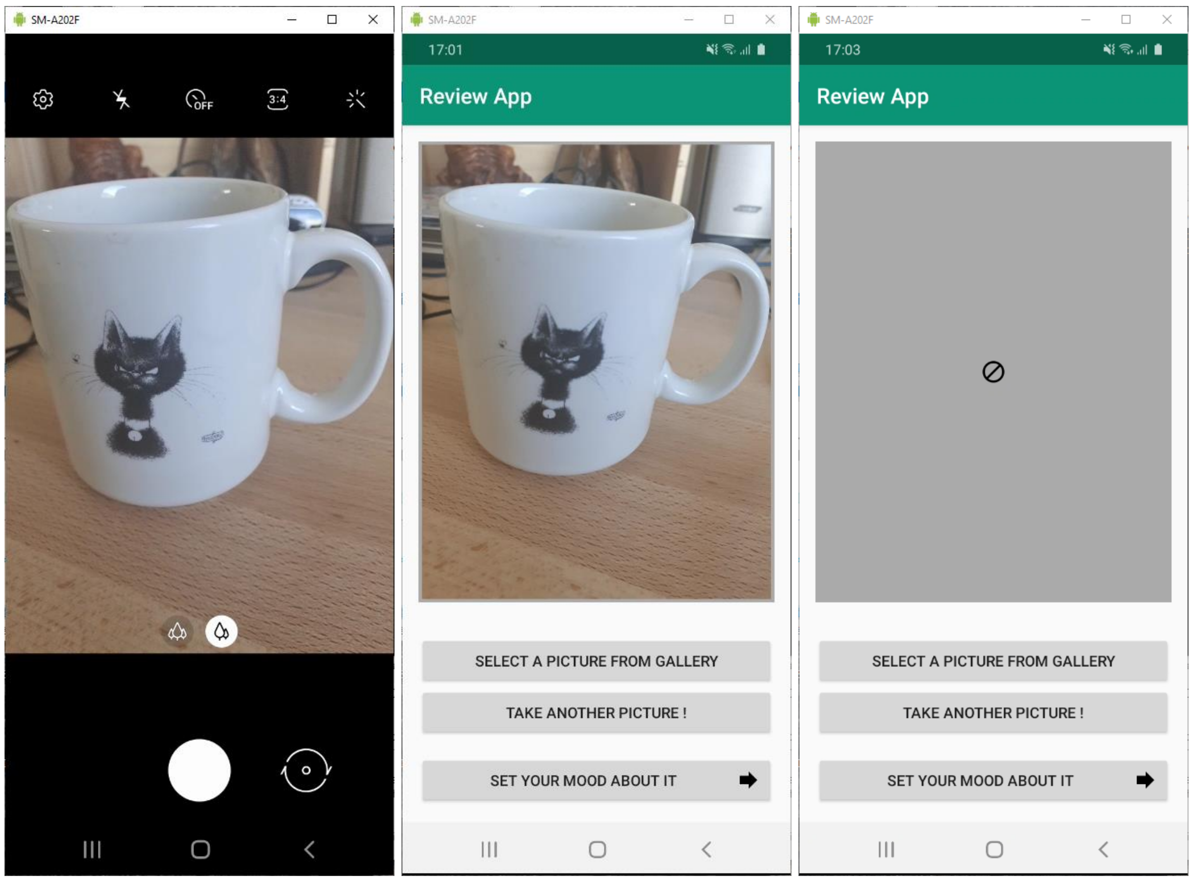


Figure 8 - Picture screens

c. Emotion scoring

In this screen, you will be asked to indicate your mood in relation to the photo taken. A simple click will allow you to position it on the graph. The "Give more details" button will take you to the last information to be entered.

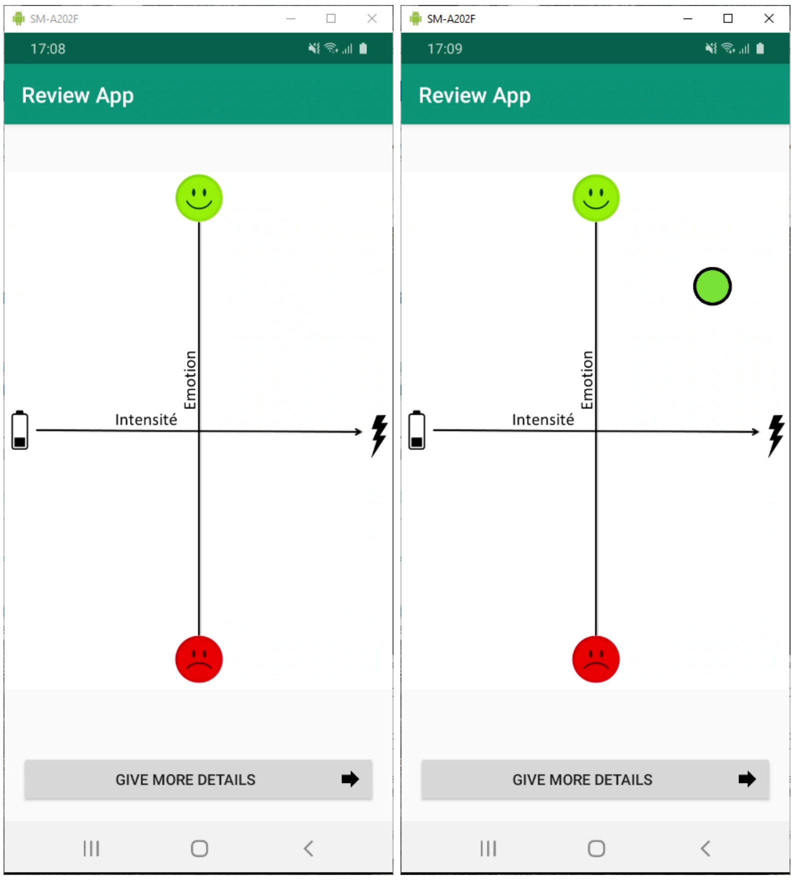


Figure 9 - (Emotion) Score screen

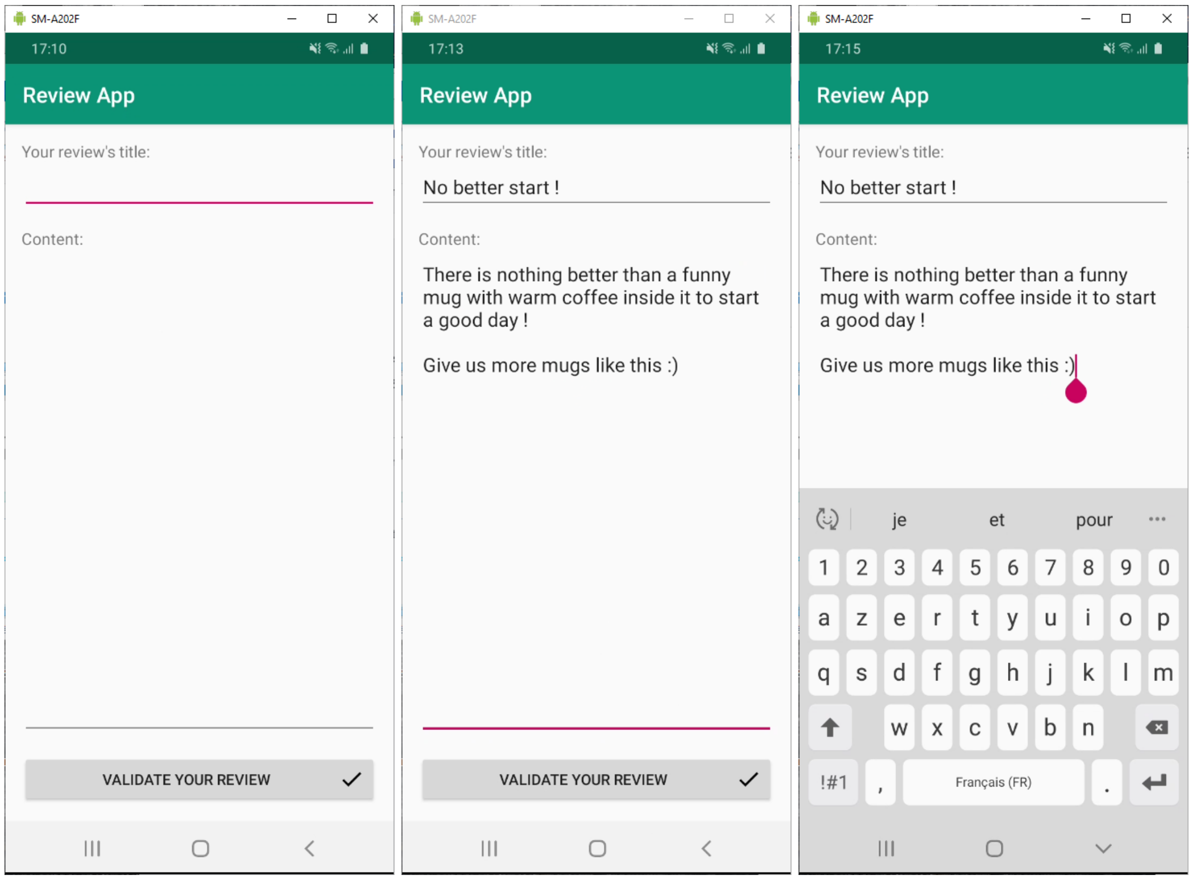
d. Additional notes

Here you can give your Review a title and additional details. Keep the title concise as it will be displayed in the summaries (application and server side) and will allow you to quickly identify each note.

Once you have completed your Review, you can validate it by clicking on the button at the bottom of the page. When you return to the main screen, a message will tell you that your Review has been saved and is ready to be sent to the server.

Please note that once you press this button, you will not be able to edit your Review.

N.B.: When you are in the details edition, the button to close the keyboard also corresponds to the one making backspaces. In order to avoid the unfortunate double-clicking and thus losing all the content you have written, the title and details will be saved even if you go backwards!



Your review's title:

No better start !

Content:

There is nothing better than a funny mug with warm coffee inside it to start a good day !

Give us more mugs like this :)

Figure 10 - Notes screen

5. List of reviews

To be able to find the Reviews that the user has written, it is possible to display a list of them. Back in the Main screen, press "See all reviews". The application will bring you to the list of Reviews saved for the current session (the session key is renamed in the Main screen).

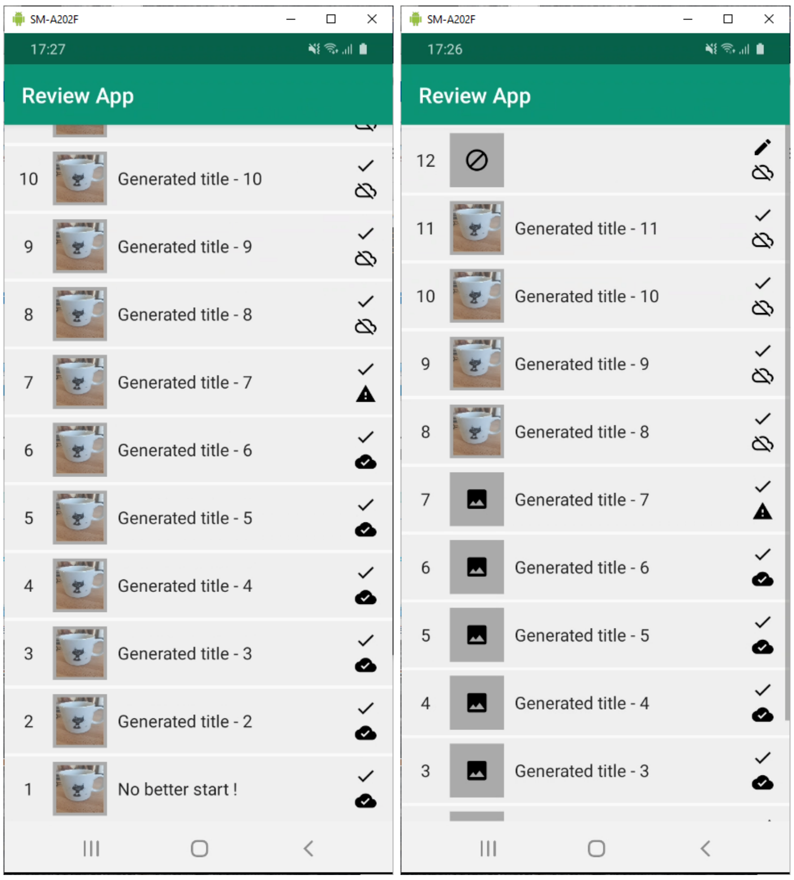
By swiping you can travel through all registered Reviews. As the display of the image is delayed, you may have to wait a little while before the corresponding icon appears. The "Image" icon tells you that the image has not yet been loaded, while a "Forbidden" icon tells you that there is no image for the Review in question.

From left to right, the different information are: a number corresponding to the ID of the Review in the local database, the photo taken for the Review, its title and then two icons.

The upper icon (a pencil or a large V) indicates the local status of the Review (the pencil indicating that it is still in editing, the large V indicating that it has been validated by the user).

The lower icon (a barred cloud, an alert symbol or a cloud with a V inside) indicates the status of the Review in relation to the server. The strikethrough cloud means that the Review has not yet been uploaded to the server. Conversely, the cloud with a V indicates that the Review has been saved on the server.

Finally, the alert symbol informs you that there was an error sending this Review to the server. On the next upload attempt, the application will only attempt to resend this review after sending all the reviews marked with a crossed-out cloud.



10 Generated title - 10
9 Generated title - 9
8 Generated title - 8
7 Generated title - 7
6 Generated title - 6
5 Generated title - 5
4 Generated title - 4
3 Generated title - 3
2 Generated title - 2
1 No better start !

12
11 Generated title - 11
10 Generated title - 10
9 Generated title - 9
8 Generated title - 8
7 Generated title - 7
6 Generated title - 6
5 Generated title - 5
4 Generated title - 4
3 Generated title - 3

Figure 11 - Reviews List

6. Details of a review

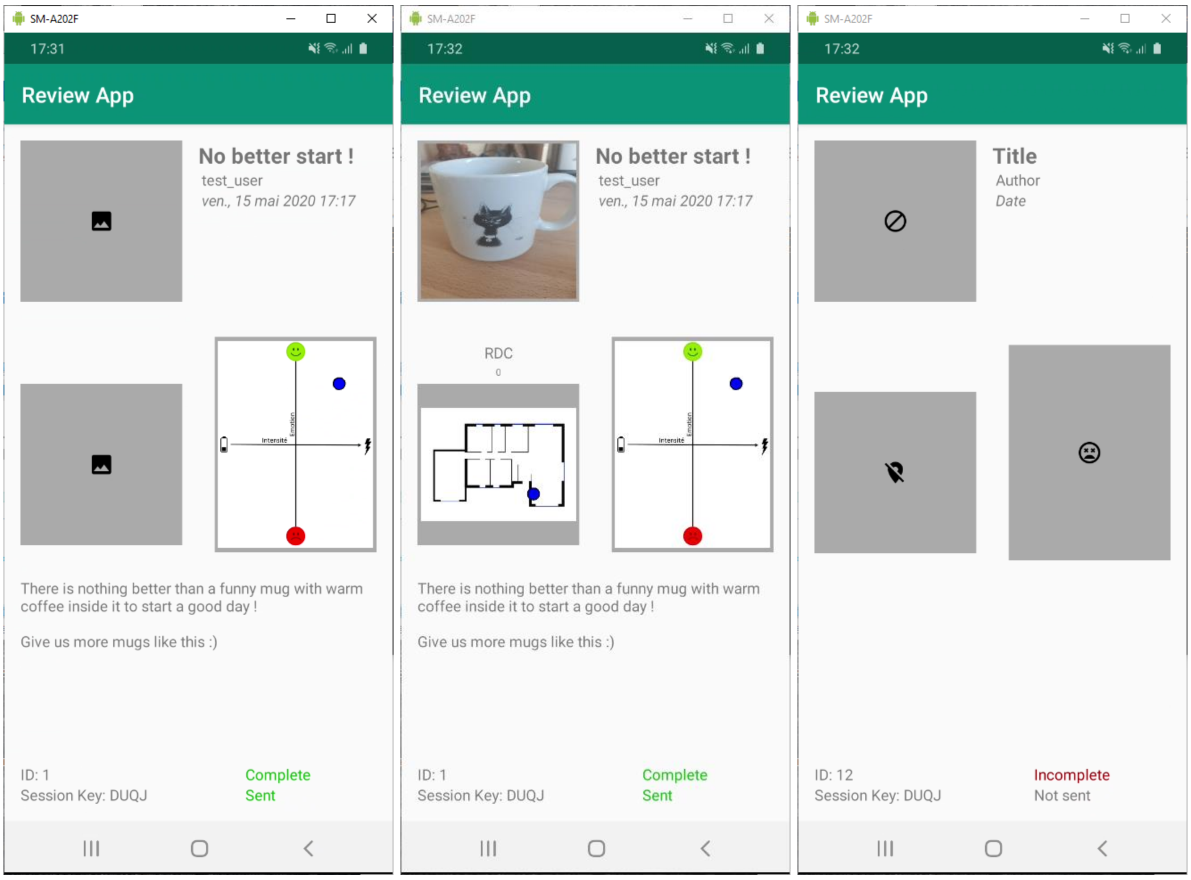
When you click on a Review in the list, you switch to its detailed view. As with the list view, it may take a few seconds for the image and location to be displayed. This screen only allows you to view Reviews but not to change them, regardless of their status.

In this display, by moving from left to right and from bottom to top, you can find this information:

- The recorded photo
- The title
- The name of the user
- The registration date
- The name of the Floor
- Floor ID

- The image of the Floor with a blue dot representing the location of the Review
- The emotion graph with a blue dot representing the emotion for the Review
- Additional annotated details
- The Review ID in the local database
- The session key used for the Review
- An indicator (Complete/Incomplete) to tell if the Review has been validated locally
- An indicator (Not sent/Sent/Error while sent) to tell if the Review has been uploaded to the server

Finally, the third screenshot below shows you the status of the display when no information has been entered and the entire Review has not yet been completed.



No better start !
test_user
ven., 15 mai 2020 17:17

There is nothing better than a funny mug with warm coffee inside it to start a good day !

Give us more mugs like this :)

ID: 1
Session Key: DUQJ
Complete
Sent

No better start !
test_user
ven., 15 mai 2020 17:17

There is nothing better than a funny mug with warm coffee inside it to start a good day !

Give us more mugs like this :)

ID: 1
Session Key: DUQJ
Complete
Sent

Title
Author
Date

ID: 12
Session Key: DUQJ
Incomplete
Not sent

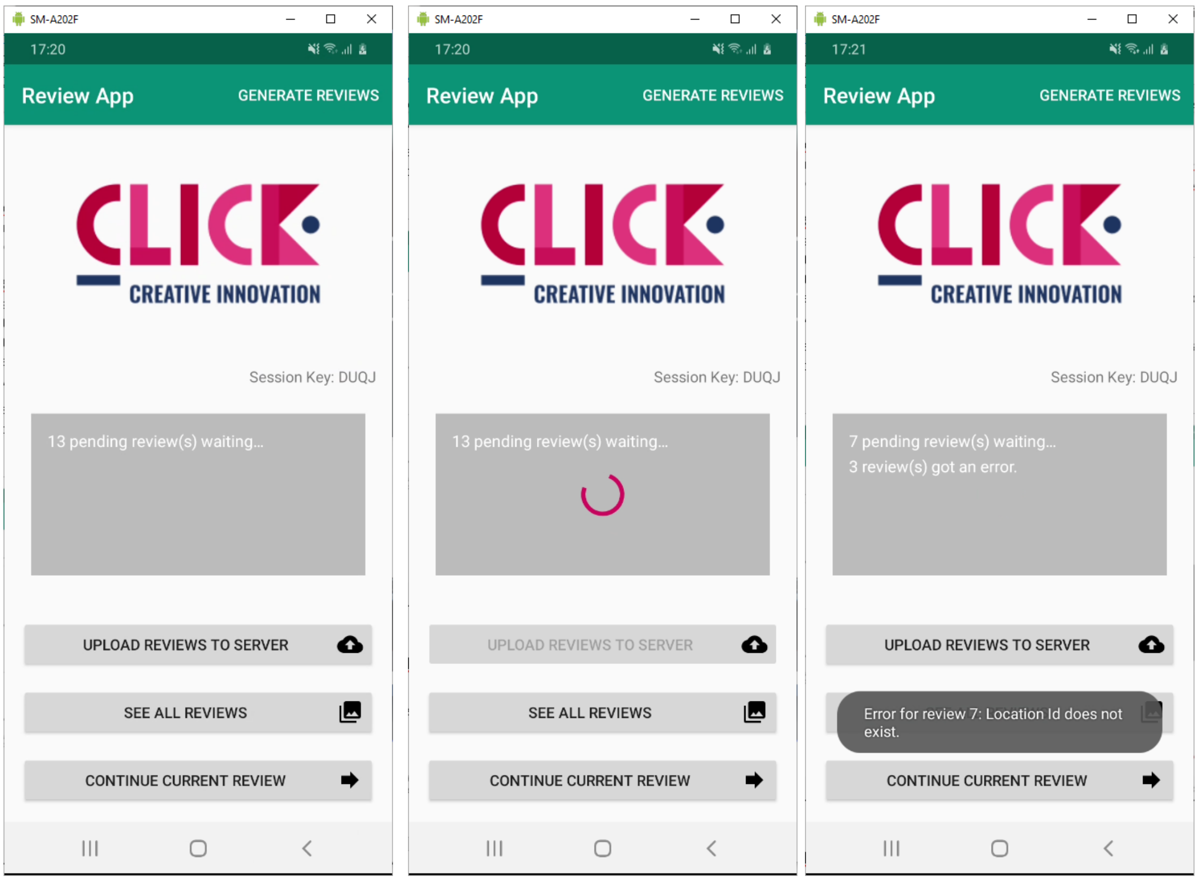
Figure 12 - Detailed Review

7. Upload review

By clicking on the "Upload Reviews to server" button, you will start the connection with the server and start uploading Reviews to it. When the upload is running, a progress bar appears in the middle of the screen and the Upload button becomes unavailable.

The number of "pending reviews" is progressively updated. Similarly, if any Reviews go into error, the display will be updated. The upload will be stopped if the user leaves this screen (in extenso: if the user disconnects, creates a new review, displays the list of reviews).

When the application encounters an error for a Review, the upload stops and an error message is displayed. It is then possible for the user to restart the upload. In the third image below, the application informs that the user is trying to upload a Review for which the Floor no longer exists (the administrator has deleted it on the server).



13 pending review(s) waiting...

7 pending review(s) waiting...
3 review(s) got an error.

Error for review 7: Location Id does not exist.

Figure 13 - Uploading reviews

8. Potential Errors

The application has been designed to be robust against network losses, voluntary or involuntary outages of the user and administrator. In all these cases, the Review being uploaded is not noted in error and only an information message is displayed to the user.

On the other hand, if the server returns an error with respect to the Review, it is noted as an error. It is not removed from the application and the application will continue to try to return it once all the Reviews that are not in error have been sent.

Apart from development errors (for example, a bad data structure sent) and resource errors on the server (for example, no more disk space available), the most likely error to occur is the case that appears on the image above. If the administrator deletes a Floor when Reviews have not yet been sent by users, they will remain blocked on the application, without being able to be sent. It will then be impossible for the administrator to retrieve them from the server.

Organization of the Unity code

This section, which is complementary to the user manual, is intended to explain the general operation of the server program and to indicate in which classes the important data structures are located, as well as the functions for manipulating them.

1. Interface

The interface consists of 2 Unity scenes, *TitleScene* and *VisualizerScene*.

For each of these scenes, we will present the GameObjects tree structure.

The names given to the scripts take the name of the GameObject to which they are attached, followed by the suffix *Script*. This is also the case for scripts associated with Prefabs.

a. TitleScene

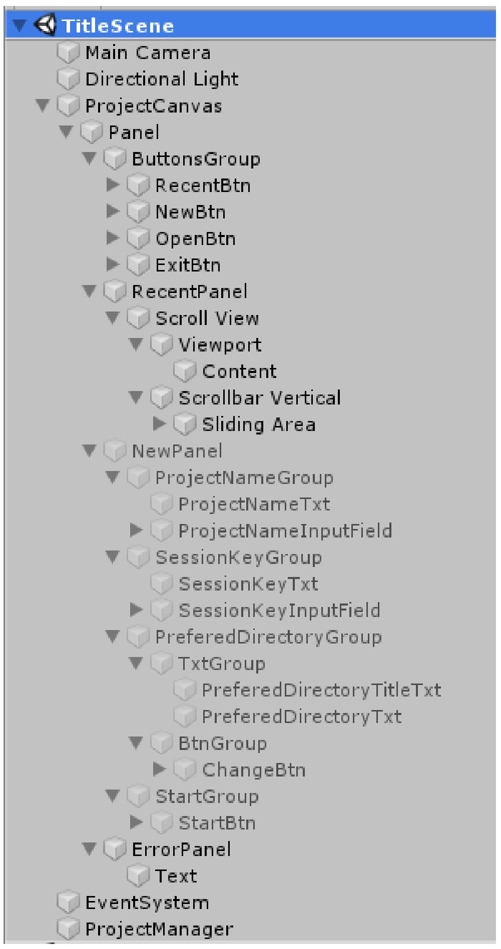
This scene corresponds to the first screen the user sees when the program is launched.

The GameObject *ProjectCanvas* contains the entire user interface for project management, with the business logic implemented in the *ProjectCanvasScript*.

The GameObject NewPanel is initially disabled. It will only become active if the user clicks on the *New* button (callback in *ProjectCanvasScript*).

The GameObject *ErrorPanel* contains a text field that can be used to tell the user if an error occurred or if a prohibited action was performed, for example creating a project with a name that already exists in the list of recent projects.

In this scene the GameObject *ProjectManager* (singleton) is created, which is not destroyed when switching to the *VisualizerScene*. The *ProjectManagerScript is* attached to it. This script is the most important script of the program.

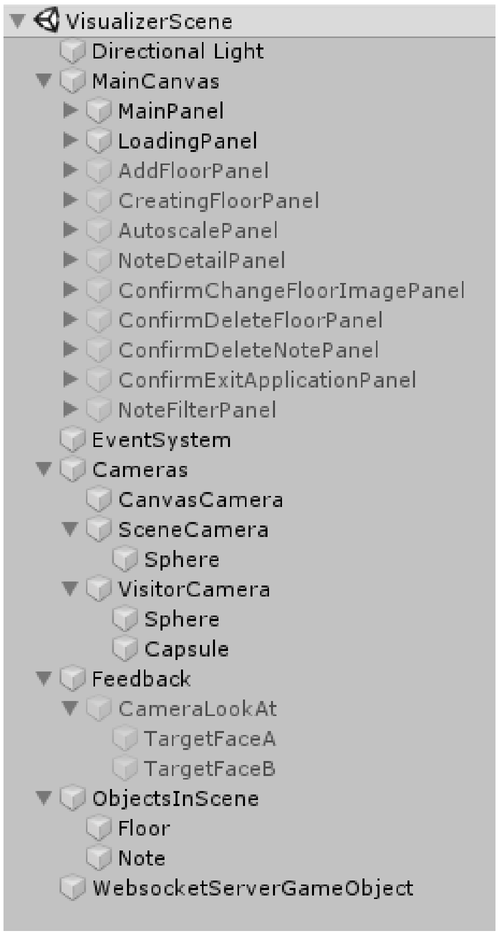


```
▼ TitleScene
    Main Camera
    Directional Light
  ▼ ProjectCanvas
    ▼ Panel
      ▼ ButtonsGroup
        ▶ RecentBtn
        ▶ NewBtn
        ▶ OpenBtn
        ▶ ExitBtn
      ▼ RecentPanel
        ▼ Scroll View
          ▼ Viewport
              Content
          ▼ Scrollbar Vertical
            ▶ Sliding Area
      ▼ NewPanel
        ▼ ProjectNameGroup
            ProjectNameTxt
          ▶ ProjectNameInputField
        ▼ SessionKeyGroup
            SessionKeyTxt
          ▶ SessionKeyInputField
        ▼ PreferedDirectoryGroup
          ▼ TxtGroup
              PreferedDirectoryTitleTxt
              PreferedDirectoryTxt
          ▼ BtnGroup
            ▶ ChangeBtn
        ▼ StartGroup
          ▶ StartBtn
      ▼ ErrorPanel
          Text
    EventSystem
    ProjectManager
```

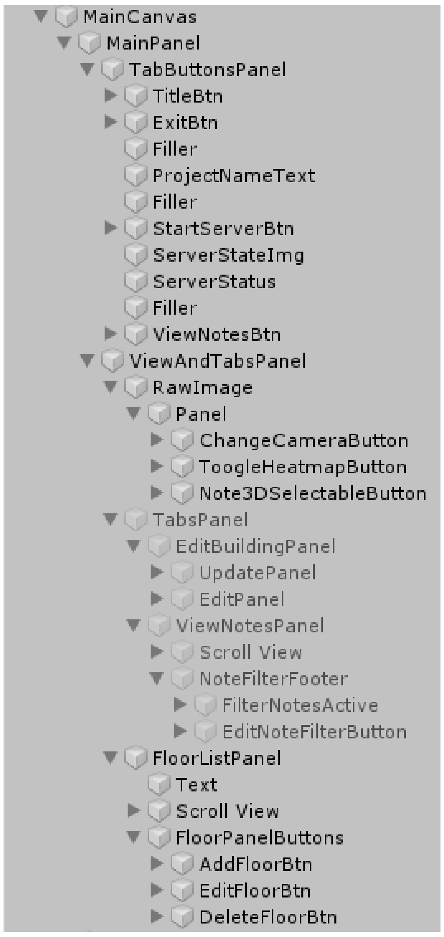
b. VisualizerScene

This scene contains the core of the program, except for the *ProjectManager, which was* created previously and not destroyed when loading the *VisualizerScene*. When working in the Unity Editor, the program has to be started from the first scene and not from the *VisualizerScene*.

The GameObject *MainCanvas* contains the entire GUI, spread over Panel-type GameObjects.



```
▼ VisualizerScene
    Directional Light
  ▼ MainCanvas
    ▶ MainPanel
    ▶ LoadingPanel
    ▶ AddFloorPanel
    ▶ CreatingFloorPanel
    ▶ AutoscalePanel
    ▶ NoteDetailPanel
    ▶ ConfirmChangeFloorImagePanel
    ▶ ConfirmDeleteFloorPanel
    ▶ ConfirmDeleteNotePanel
    ▶ ConfirmExitApplicationPanel
    ▶ NoteFilterPanel
    EventSystem
  ▼ Cameras
      CanvasCamera
    ▼ SceneCamera
        Sphere
    ▼ VisitorCamera
        Sphere
        Capsule
  ▼ Feedback
    ▼ CameraLookAt
        TargetFaceA
        TargetFaceB
  ▼ ObjectsInScene
      Floor
      Note
    WebsocketServerGameObject
```



```
▼ MainCanvas
  ▼ MainPanel
    ▼ TabButtonsPanel
      ▶ TitleBtn
      ▶ ExitBtn
        Filler
        ProjectNameText
        Filler
      ▶ StartServerBtn
        ServerStateImg
        ServerStatus
        Filler
      ▶ ViewNotesBtn
    ▼ ViewAndTabsPanel
      ▼ RawImage
        ▼ Panel
          ▶ ChangeCameraButton
          ▶ ToogleHeatmapButton
          ▶ Note3DSelectableButton
      ▼ TabsPanel
        ▼ EditBuildingPanel
          ▶ UpdatePanel
          ▶ EditPanel
        ▼ ViewNotesPanel
          ▶ Scroll View
          ▼ NoteFilterFooter
            ▶ FilterNotesActive
            ▶ EditNoteFilterButton
      ▼ FloorListPanel
          Text
        ▶ Scroll View
        ▼ FloorPanelButtons
          ▶ AddFloorBtn
          ▶ EditFloorBtn
          ▶ DeleteFloorBtn
```

Of these, the *MainPanel* (deployed in the image on the right) is divided into sub-panels. The *TabButtonPanel is* the menu bar at the top of the window. The *ViewAndTabsPanel* contains :

- The 3D view and its control buttons (GameObject *RawImage*)
- The *TabsPanel* that can be deployed or folded to display the floor editor or the list of reviews (instantiation of *NoteItemListPrefab*)
- The *FloorListPanel* which contains the display of the floors (instantiation of *FloorItemListPrefab*).

The other Panels are GUI overlays that come to the foreground when activated by one of the scripts. Only *LoadingPanel* is enabled. This is necessary because the *ProjectManagerScript* will look for it when loading the *VisualizerScene to* display information about loading 3D models and reviews.

The GameObject *Cameras* contains the cameras used to calculate the display rendering. The *CanvasCamera* corresponds to what is displayed on the screen, the other two are used to explore the 3D world and are represented by simple 3D models (sphere or capsule).

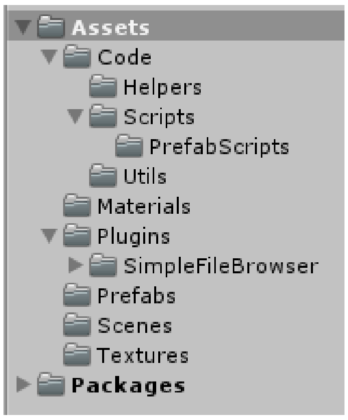
The GameObject *Feedback* allows you to display a crosshair on the screen. It is disabled by default.

The GameObject *ObjectsInScene* has the container in which the floor meshes will be placed when loaded (instantiation of *FloorPrefab*), each floor containing its associated GameObject 3D reviews (instantiation of *NotePrefab*).

Finally, the *WebsocketServerGameObject* is a container for the script implementing the Websocket server.

2. Resource files

a. Assets folder



The Assets folder contains all resource files of the program.

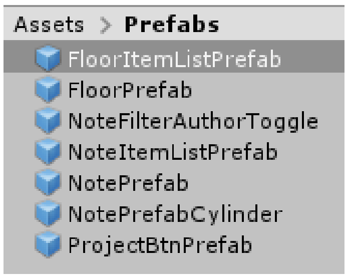
The Code file is subdivided into three files. The *Helper folder* contains the *GuiHelper* class with static functions used in various scripts and the *SimulationHelper* file contains the definition of the data structures manipulated by the program (more details in the section 3.a). The *Scripts* folder contains all scripts attached to GamesObjects, as well as the prefab scripts in the *PrefabScripts* subfolder. The *Utils folder* contains classes whose purpose is to simplify data manipulation, mainly for image processing.

The *Plugins* folder contains a compiled websocket-sharp DLL library. It is provided here to ensure the project works but should always be replaced by the latest available version. The *SimpleFileBrowser* folder contains the code of the plugin of the same name used to propose file browsing windows to the user. It is also recommended to replace it with the latest available version.

The contents of the *Prefabs* file are presented in the following sub-section.

The *Textures* folder contains all the images used in the program, especially for the GUI, which are not pictures associated with the reviews.

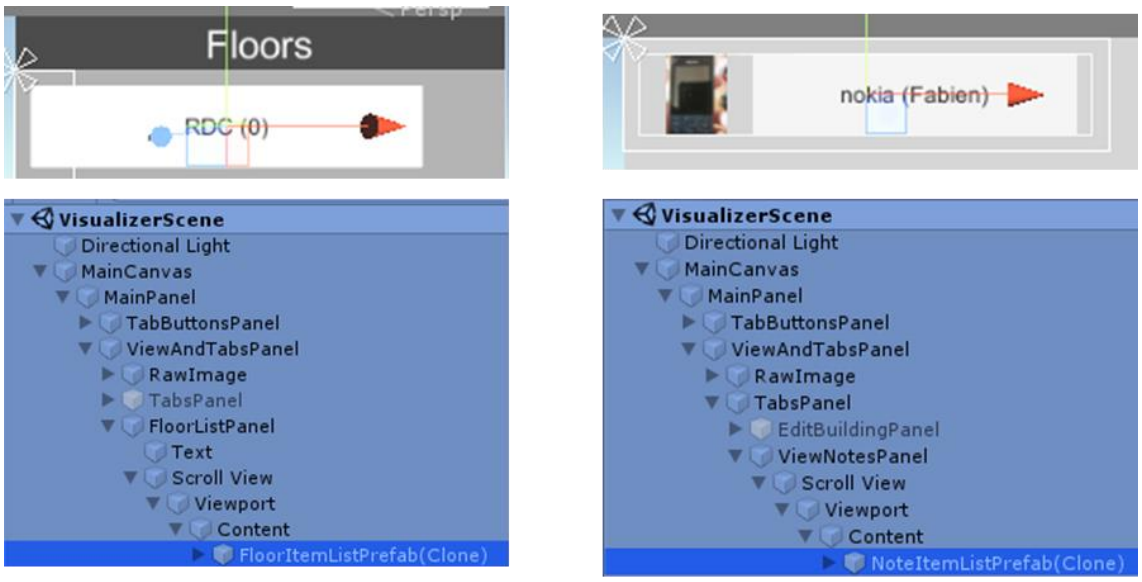
b. Prefabs



Prefabs are more complex GameObjects than other basic GameObjects that can be instantiated more easily from scripts.

- FloorItemListPrefab: 2D GameObject used in the GUI (*FloorListPanel*) to display the list of floors in the project
- FloorPrefab: 3D GameObject containing the meshes of a floor in the 3D view. It also contains a container for the reviews
- NoteFilterAuthorToggle: GameObject 2D used to display a list of selectable items (review authors' names) on the *NoteFilterPanel*

- NoteItemListPrefab: 2D GameObject used in the GUI (*Scroll View of the ViewNotesPanel*) to display the filtered list of project reviews
- NotePrefab: 3D GameObject containing four cross-layered planes on which is applied the texture of the photo of a review for display in the 3D view
- NotePrefabCylinder: Like the previous one but in the shape of a cylinder (not used)
- ProjectButtonPrefab: 2D GameObject used to display recently opened projects in *TitleScene*

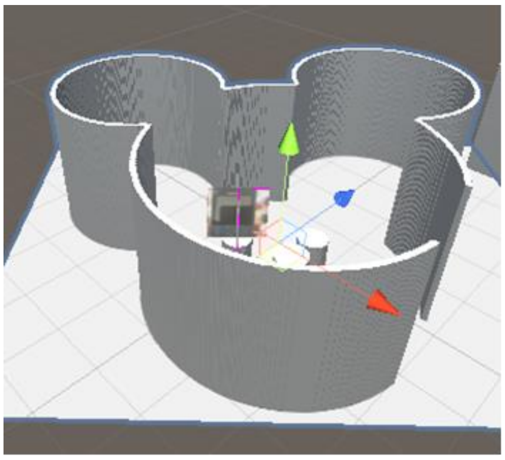
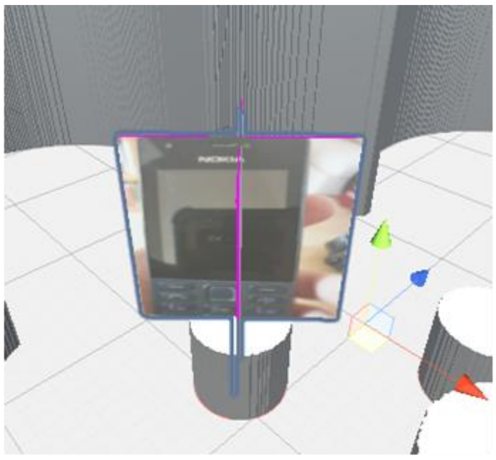
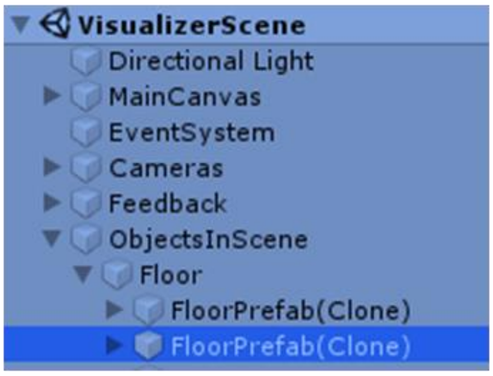


```
▼ VisualizerScene
    Directional Light
  ▼ MainCanvas
    ▼ MainPanel
      ▶ TabButtonsPanel
      ▼ ViewAndTabsPanel
        ▶ RawImage
        ▶ TabsPanel
        ▼ FloorListPanel
            Text
          ▼ Scroll View
            ▼ Viewport
              ▼ Content
                ▶ FloorItemListPrefab(Clone)
```

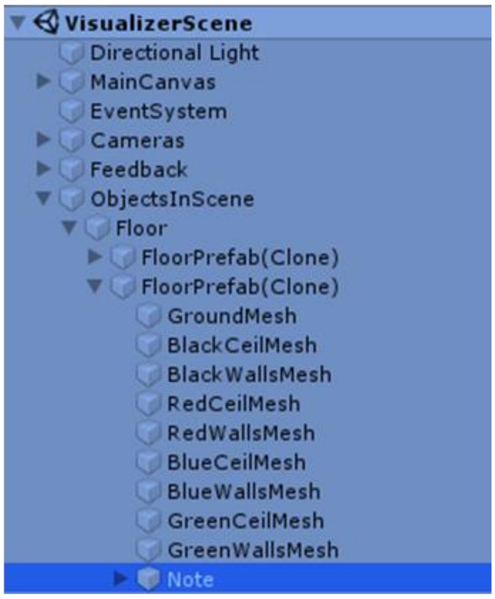
```
▼ VisualizerScene
    Directional Light
  ▼ MainCanvas
    ▼ MainPanel
      ▶ TabButtonsPanel
      ▼ ViewAndTabsPanel
        ▶ RawImage
        ▼ TabsPanel
          ▶ EditBuildingPanel
          ▼ ViewNotesPanel
            ▼ Scroll View
              ▼ Viewport
                ▼ Content
                  ▶ NoteItemListPrefab(Clone)
```

-

Representation in the Unity editor of FloorItemListPrefab (left) and NoteItemListPrefab (right) instantiated and associated hierarchies

```
▼ VisualizerScene
    Directional Light
  ▶ MainCanvas
    EventSystem
  ▶ Cameras
  ▶ Feedback
  ▼ ObjectsInScene
    ▼ Floor
      ▶ FloorPrefab(Clone)
      ▶ FloorPrefab(Clone)
```



```
▼ VisualizerScene
    Directional Light
  ▶ MainCanvas
    EventSystem
  ▶ Cameras
  ▶ Feedback
  ▼ ObjectsInScene
    ▼ Floor
      ▶ FloorPrefab(Clone)
      ▼ FloorPrefab(Clone)
          GroundMesh
          BlackCeilMesh
          BlackWallsMesh
          RedCeilMesh
          RedWallsMesh
          BlueCeilMesh
          BlueWallsMesh
          GreenCeilMesh
          GreenWallsMesh
        ▶ Note
```

Representation in the Unity editor of FloorPrefab (left) and NotePrefab (right) instantiated and associated hierarchies

3. Code C#

a. Data structures

These structures in the Helpers/SimulationHelper.cs file contain the data necessary to display the 3D images and models in the 3D view. Those declared as Serializable can be saved and loaded more easily to or from XML or JSON files.

FloorStruct (Serializable)

Structure containing the data used to extrude the building plan into a 3D model and place it in the 3D scene. The Id field is unique, and his value changes if the floor image is modified. It is used by the Android application to know if the floor image in the cache is the latest available version.

NoteStruct (Serializable)

Contains data relating to a review. The *Data* field (of type ReceivedData - Serializable) represents the content of the JSON document representing the review sent by the

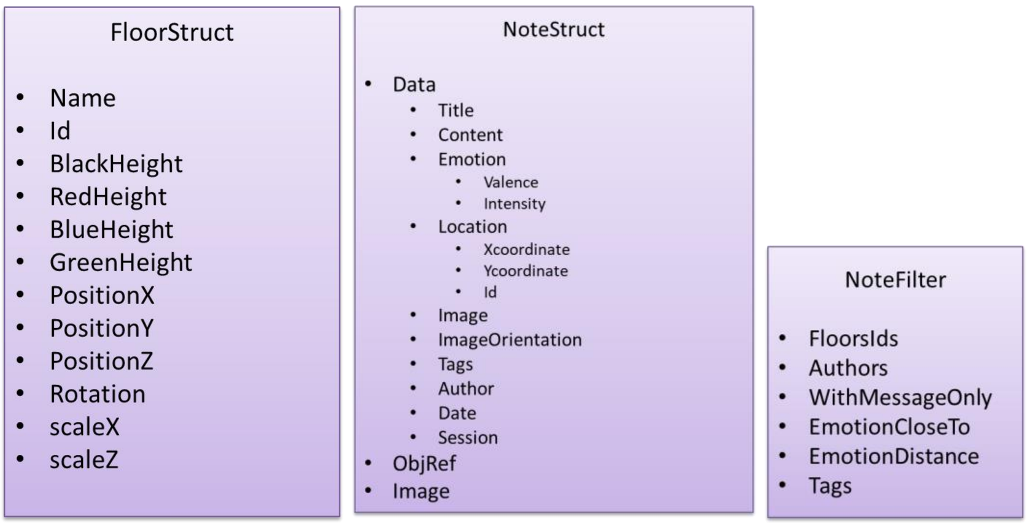
ReviewApp (see section 4). On receipt, the Data/Image field containing a string of characters base 64 representing the JPEG image is replaced by the name of the image file saved on the disk.

The *ObjRef* field contains a reference to the NotePrefab corresponding to this review in the 3D view.

The *Image* field is a Texture2D field in which the resized version of the photo in the review is loaded.

NoteFilter

Structure to simplify the filtering of a list of reviews.



b. Utility classes

These classes aim to simplify the management of images and the creation of 3D models of floors from their plans. Apart from *ImageExtruder*, none have been developed by us. The references are given in the Readme of the *Utils* directory and in each of the C# files concerned.

ImageExtruder

Class encapsulating *MeshGenerator* to simplify its use.

MeshGenerator

Class creating the meshes of a floor from an image loaded as a texture (one mesh per color on the texture).

TextureFilter

Library of image processing functions used to clean up images before generating meshes.

TextureScale

Image resizing tool. Used to reduce the size of images received with reviews to limit the program's memory usage.

c. Main scripts

ProjectManagerScript

Script of GameObject singleton with functions to load, parse, sort and save reviews and floors as well as structure lists containing review (*NotesList*) and floor (*SimulationSetup.FloorList*) data. The associated GameObject is declared in the *TitleScene* and is not destroyed at scene change. The main functions are:

- Clear: resets all the fields of the class
- IsNoteMessage: checks that a received websocket message contains the JSON formatted as expected
- Update: if a websocket message containing a review is received, the review is created in this function (main thread)
- SaveNote: saves the review on the hard disk in JSON format
- CreateNote: instantiate a review of type *NoteStruct*, create the *NotePrefab* in the 3D view and add the review in the *NotesList*
- LoadFloors: coroutine used to load floor data from the Simulation.xml file.
- LoadNotes: coroutine used to load all the notes of the project
- GetNotesFiltered: function returning the list of reviews corresponding to a given filter
- DeleteNote: deletes a review
- SaveSetup: saves the whole project
- Create/Rename/DeleteFloorFolder: management of the floor folder on the hard drive

FloorsManagerScript

This script is attached to the GameObject *FloorListPanel*, a panel displaying the list of project floors. It is in charge of managing the floors and their representation in the GUI. Its main functions are:

- Clear: destroys the meshes on all floors and resets the class.
- DeleteCurrentFloor: deletes the currently selected floor
- Update: if a floor needs to be created or modified, this is done in this function (main thread)
- CreateFloorMesh: creates the mesh associated with the floor for the 3D view
- CreateFloorItemList: adds an item to the list of floors in the scrollable view

FloorItemListPrefabScript

Although being a prefab script mainly dedicated to the GUI events management, this script is nevertheless important because here the heatmap of each floor is calculated when loading the project or adding a new review. It is in this script that the texture associated with the floor and its representation in base64 character string is stored, computed at loading time.

WebsocketServerScript

This script is in charge of running the websocket server. It is attached to the GameObject *WebsocketServerGameObject*. For details about the websocket messages exchanged between the ReviewApp and the server, see section 4.

CameraManipulationScript

This script is attached to the GameObject *RawImage* in the *MainPanel*. It allows to switch from the omniscient camera view to the visitor camera view in the 3D view and manages the user interaction with this view to move the active camera. It also allows highlighting a flyover review in the 3D view, if the *NotesSelectable* option is enabled.

4. Formatting Websocket messages

Description of the messages exchanged between the websocket server and Android smartphones connected to the same WiFi network

Detail of terms used:

- *IP* is the IPv4 address of the computer on the network.
- *Port* is the Websocket listening port set in the Unity editor (7777 by default)
- *ProjectName* is the name of the project opened in the server application
- *SessionKey* is a 4-character code chosen at the creation of the project. It is displayed in the server information bar.

a. Services

Ping

Allows you to check if the server responds

URI: ws://IP:Port/SessionKey

Expected text message: **None**

Text message response: **None**

Answer to ping: **"Pong"**

Floors

Query to obtain the name, identifier and image of each of the floors of the project

URI: ws://IP:Port/SessionKey/floors

Expected text message: **None**

Text message response: **JSON formatted as follows**

```
{
  "Floors":[
    {
        "Name": (string) "FloorName_0",
        "Id": (int - unique) Id_0,
        "Image": (string) "FloorImageData_0 (as PNG Base64 string)"
    },
        {
        "Name": (string) "FloorName_1",
        "Id": (int - unique) Id_1,
        "Image": (string) "FloorImageData_1 (as PNG Base64 string)"
    }
    ...
  ]
}
```

Ping response: **same as text message response**

Notes

Sending a note to the server

URI: ws://IP:Port/SessionKey/note

```
{
    "Title": (string) "the title review",
    "Content": (string) "the content review",
    "Emotion": {
        "Valence": (float) [-1 -> 1],
        "Intensity": (float) [-1 -> 1]
    },
    "Location": {
        "XCoordinate": (float) [0 -> 1],
        "YCoordinate": (float) [0 -> 1],
        "Id": (int - unique) Id
    },
    "Image": (string) "NoteImageData (as JPG Base64 string)",
    "Author": (string) "Author of the review",
    "Date": (string) "Date of creation",
    "Tags": (string) "tags separated with commas (optional)",
    "Session": (string) "Session key code"
}
```

Note: X and Y coordinates are given relative to the floor image. The origin is placed at the top left of the image.
Note 2: The image must be in JPG format, encoded as a Base64 character string.

Reply to text message:

- **INCORRECT_NOTE_FORMAT**: The message does not start with the { character or does not end with }, or one of the following fields is missing: "NoteMessage", "Emotion" , "Location", "NoteData".
- **JSON_PARSING_ERROR**: The server failed to decode JSON data.
- **TEXTURE_CONVERSION_ERROR**: NoteImageData is not a valid Base64 string
- **SAVE_DATA_ERROR**: The server failed to decode or save the note image.
- **UNKNOWN_FLOOR_ID**: The 'Location Id' field takes a value that does not exist in the list of floors in the project.
- **ERROR**: Unidentified error - see server side log files for details
- **OK**: The note is created and the associated files are saved.

Answer to ping: **None**

Android code organization

In this section we will explain the general organization and the main principles used to develop the application. We will first go through the different displays and then explain the structure used to organize the code.

The following explanations are based on the folder root "/ReviewApp/app/src/main".

1. Display

All the images and icons used in the application are gathered in the "res/drawable" folder.

The layouts developed for each of the different screens of the application can be found in the "res/layout" folder. Basically, each activity has a corresponding layout. In addition to this, the "fragment_review.xml" file manages the display of Reviews in the summary list.

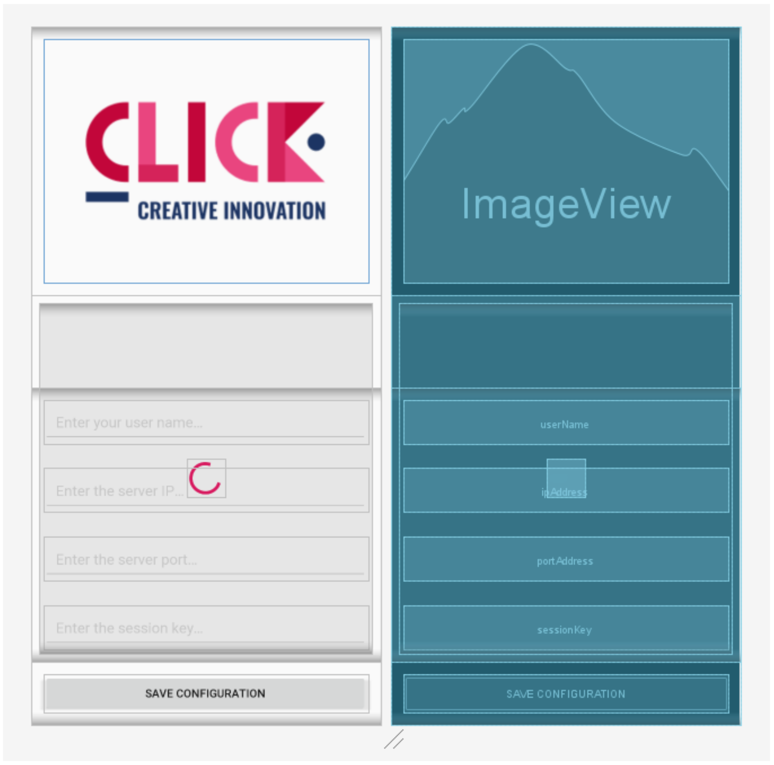


Figure 14 - Displaying the "activity_loading.xml" file

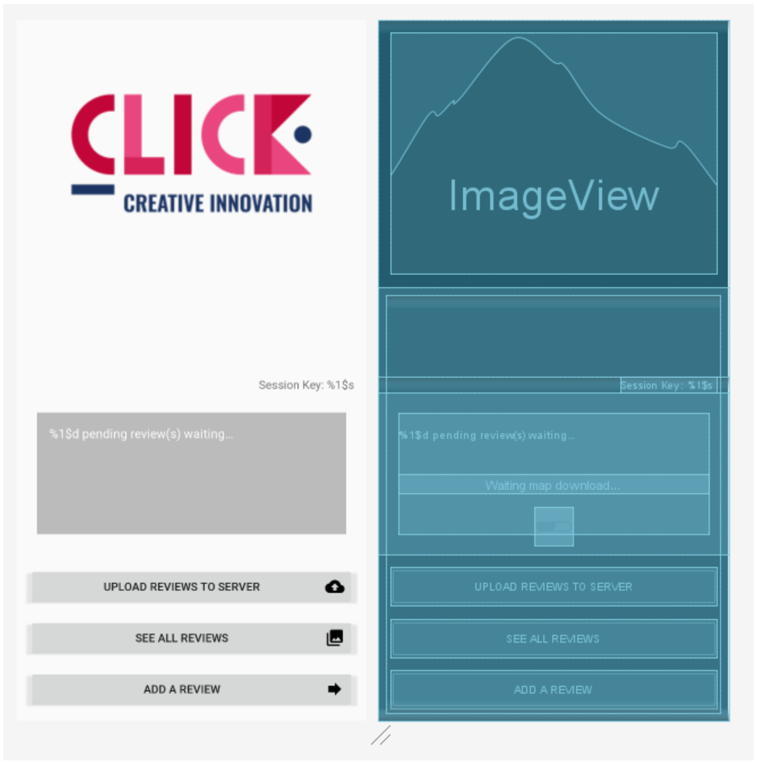


Figure 15 - Displaying the "activity_recap.xml" file

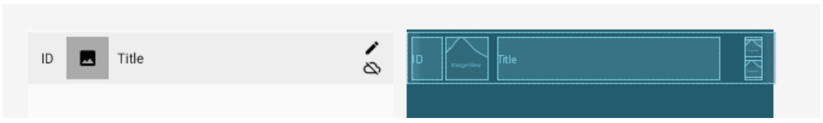
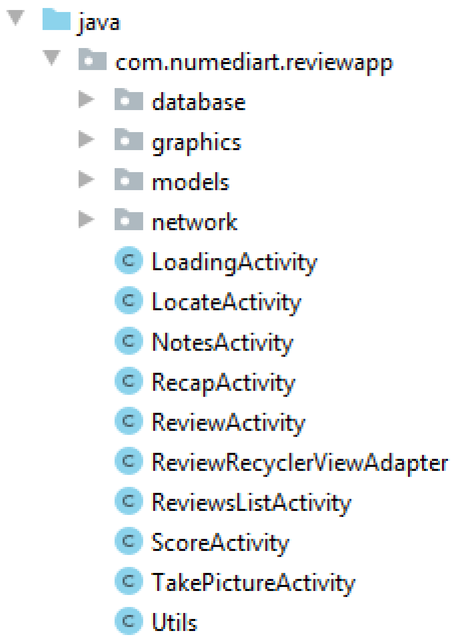


Figure 16 - Displaying the "fragment_review.xml" file

As previously mentioned, all the texts displayed to the user are collected in the file "res/values/strings.xml".

2. Java Code

The code can be found in "/ReviewApp/app/src/main/java/com/numediart/reviewapp". The main classes are in the folder itself and have a name ending with "Activity". Each of these is in correspondence with the layout of the same name.



```
java
  com.numediart.reviewapp
    database
    graphics
    models
    network
    LoadingActivity
    LocateActivity
    NotesActivity
    RecapActivity
    ReviewActivity
    ReviewRecyclerViewAdapter
    ReviewsListActivity
    ScoreActivity
    TakePictureActivity
    Utils
```

Figure 17 - Code Organization

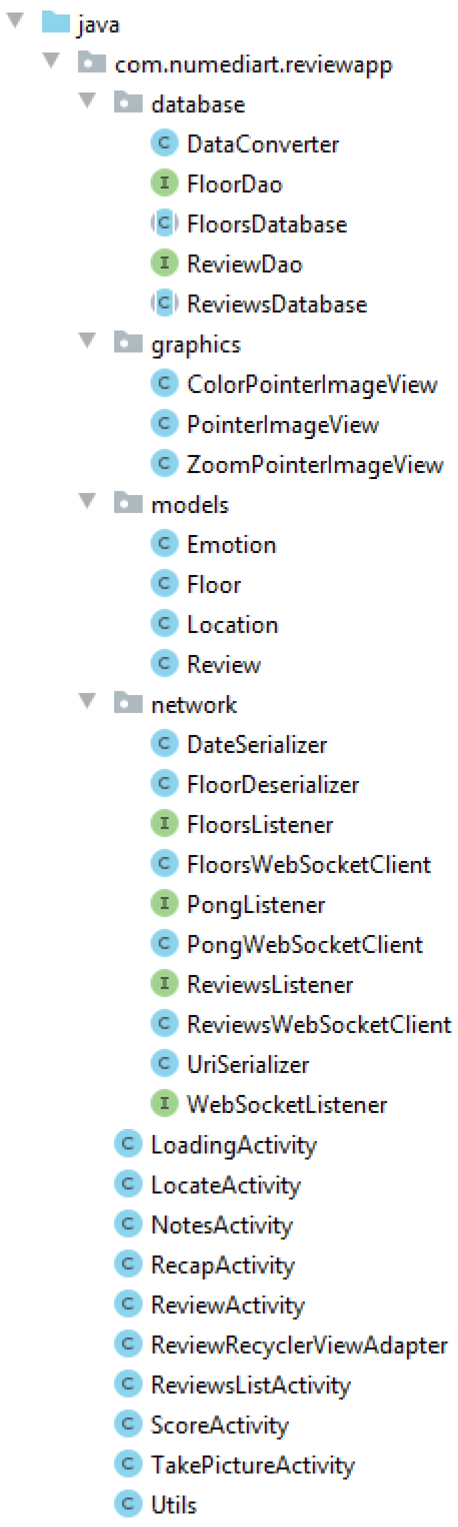
A Utils class includes some functions to abstract network queries and interface chaining from the user's point of view. It also has an AsyncImageTask to load images asynchronously.

The **database** package contains the classes for managing the two application databases.

The **graphics** package contains the classes extending ImageView so that it is clickable (**PointerImageView**) or zoomable (**ZoomPointerImageView**, used in **LocateActivity**). The **ColorPointerImageView** class (used in **ScoreActivity**) finally allows to customize the cursor by choosing its color according to its position in a reference image.

The **models** package contains, among others, the **Review** and **Floor** entities, useful at any level of the application.

Finally, the **network** package gathers the classes allowing to make the different requests to the server.



```
java
  com.numediart.reviewapp
    database
      DataConverter
      FloorDao
      FloorsDatabase
      ReviewDao
      ReviewsDatabase
    graphics
      ColorPointerImageView
      PointerImageView
      ZoomPointerImageView
    models
      Emotion
      Floor
      Location
      Review
    network
      DateSerializer
      FloorDeserializer
      FloorsListener
      FloorsWebSocketClient
      PongListener
      PongWebSocketClient
      ReviewsListener
      ReviewsWebSocketClient
      UriSerializer
      WebSocketListener
    LoadingActivity
    LocateActivity
    NotesActivity
    RecapActivity
    ReviewActivity
    ReviewRecyclerViewAdapter
    ReviewsListActivity
    ScoreActivity
    TakePictureActivity
    Utils
```

Figure 18 - Code organization (detailed)

a. User Interactions

From the user's point of view and according to the actions he performs, the user will move through the **Activities** according to the following principle scheme. For all intents and purposes, the correspondence with the graphical layout has also been indicated.

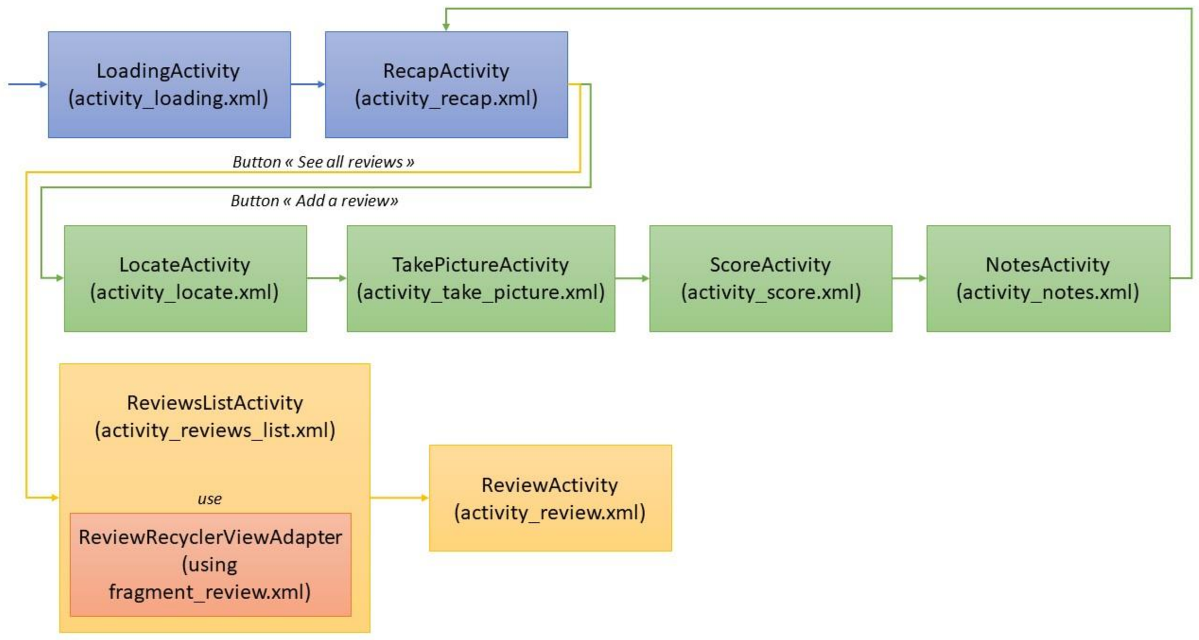

LoadingActivity (activity_loading.xml)
RecapActivity (activity_recap.xml)
Button « See all reviews »
Button « Add a review»
LocateActivity (activity_locate.xml)
TakePictureActivity (activity_take_picture.xml)
ScoreActivity (activity_score.xml)
NotesActivity (activity_notes.xml)
ReviewsListActivity (activity_reviews_list.xml)
use
ReviewRecyclerViewAdapter (using fragment_review.xml)
ReviewActivity (activity_review.xml)

Figure 19 - User Diagram

At startup, it will arrive in the **LoadingActivity** to enter its login information. It will then switch to **RecapActivity** where it can choose between "See all reviews", "Add a review" or "Upload all reviews to server" (see the "Network Requests" section for this last action).

In the first case, it will arrive in **ReviewsListActivity,** implementing a RecyclerView to display the fragments described in "fragment_review.xml". If he clicks on one of the fragments, he will be redirected to **ReviewActivity** to get the details of this Review.

In the second case, the one where he wants to add a Review, he will go through the different screens allowing him to complete a Review, until he is sent back by the application to the **RecapActivity**.

b. Database

There are two databases in the application, one to store **Floors,** the other to store **Reviews**. The organization in classes is essentially the same in both cases.

In the **model** package, we find either the **Floor** class containing all its variables, or the **Review** class which instantiates two subclasses, **Location** and **Emotion**. These two models, thanks to Room annotations, are transformed into a database and the **FloorDao** and **ReviewDao** interfaces provide all possible interaction functions towards the respective databases.

Finally, to manage possible concurrent accesses and to be able to complexify the requests made to the database, the **FloorsDatabase** and **ReviewsDatabase** classes are the two entry points for all calls in the Java code.

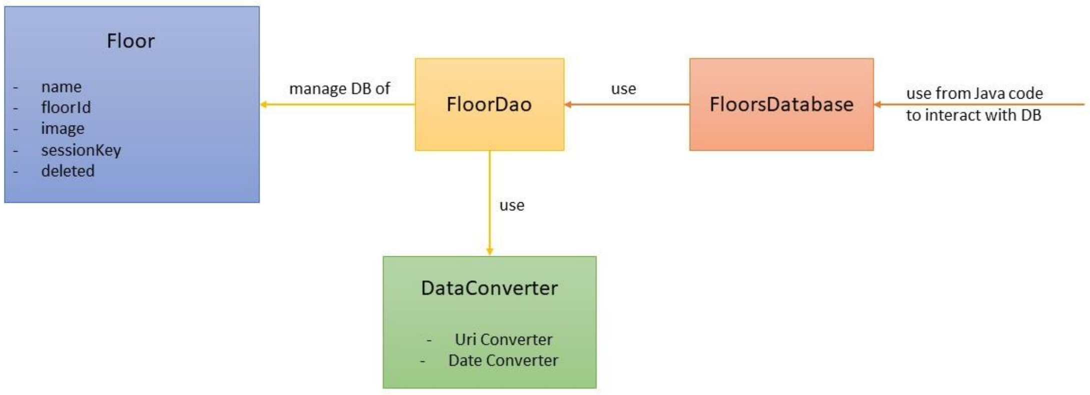


Figure 20 - Floor DB interactions

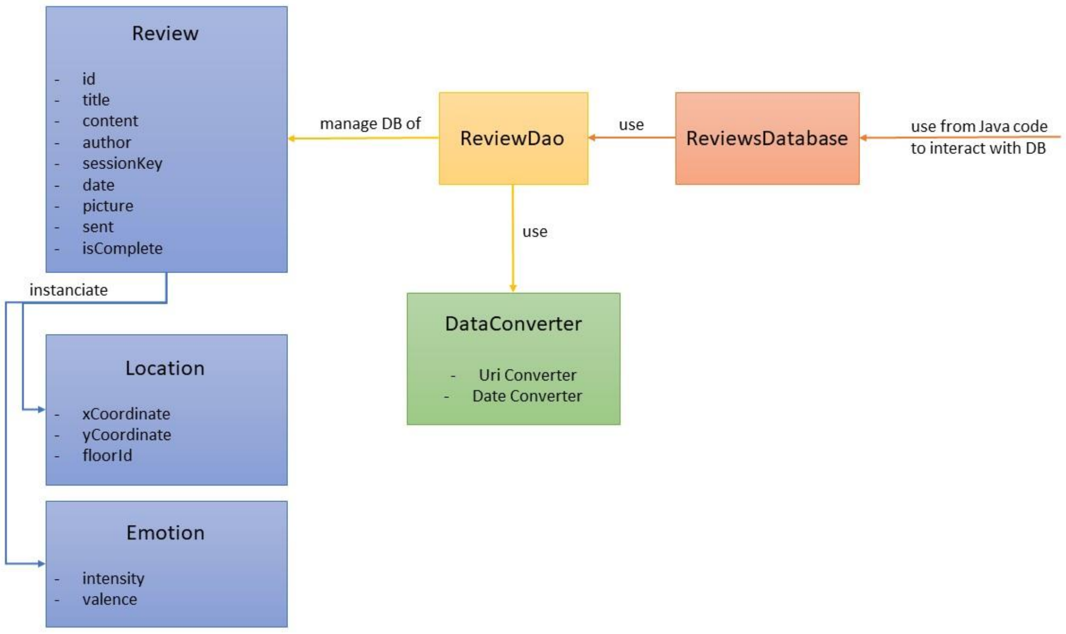


Figure 21 - Review DB interactions

c. Network requests

Three different exchanges are made with the server at different times in the application.

The first request, the simplest, is implemented in **PongWebSocketClient**. It verifies that the application can contact the server with the information entered by the user (IP address, port, session key). The main thread is notified thanks to a callback system whose interface is implemented in the activity.

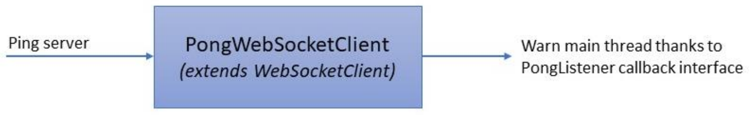

Ping server
PongWebSocketClient
(extends WebSocketClient)
Warn main thread thanks to
PongListener callback interface

Figure 22 - Pong server request

The second request allows the application to receive the **Floors** related to the current session. Thanks to the **FloorsWebSocketClient** class, the application receives a JSON message sent by the server and decodes it thanks to the **FloorDeserializer** class. The FloorDeserializer class instantiates a **Floor** model and also stores the received images in the application data. Finally, it remains to update the database with this information.

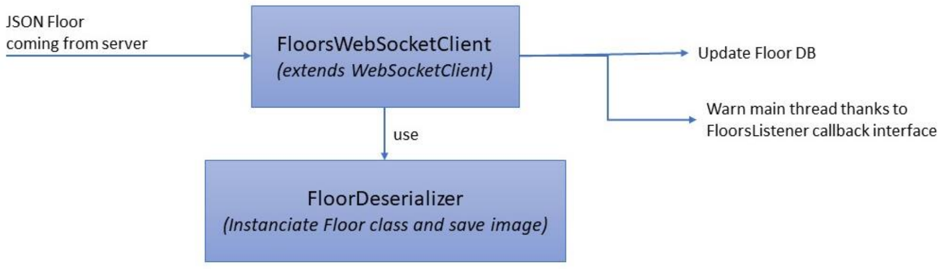

JSON Floor
coming from server
FloorsWebSocketClient
(extends WebSocketClient)
Update Floor DB
Warn main thread thanks to
FloorsListener callback interface
use
FloorDeserializer
(Instanciate Floor class and save image)

Figure 23 - Floors server request

Finally, the third request sends the **Reviews** to the server. It starts when the user requests a synchronization. To do so, the **ReviewsWebSocketClient** class uses the UriSerializer and DateSerializer classes. The transformation of the model into a JSON message is managed thanks to GSON annotations (different from Room annotations for database management) in the **Review**, **Location** and **Emotion** classes.

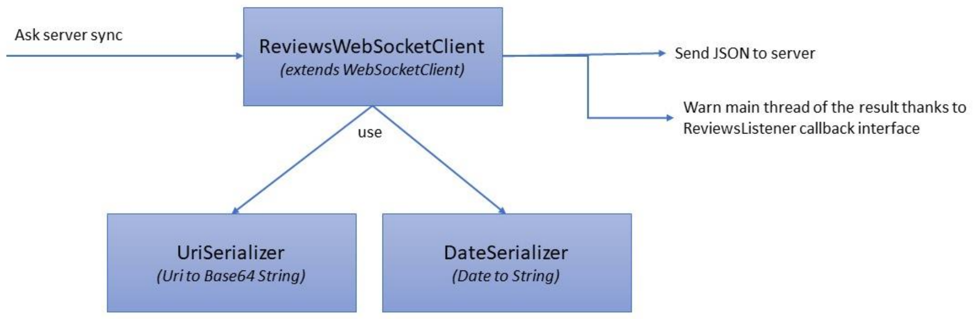


Figure 24 - Reviews server request

These three queries are used at different times. The **PongWebSocketClient** allows the user to enter the **RecapActivity** if the server could be contacted. Otherwise and if the **Floors** already exist, the offline mode will be proposed. Once in the **RecapActivity**, the **FloorsWebSocketClient** will download for the first time (or refresh) the **Floors** corresponding to the session. Finally, when the user clicks on the "Upload reviews to server" button, an asynchronous task will be started, thanks to the internal class **AsyncUploadTask** (present in **RecapActivity**). This class will then use **ReviewsWebSocketClient** to send **Reviews** one after the other.

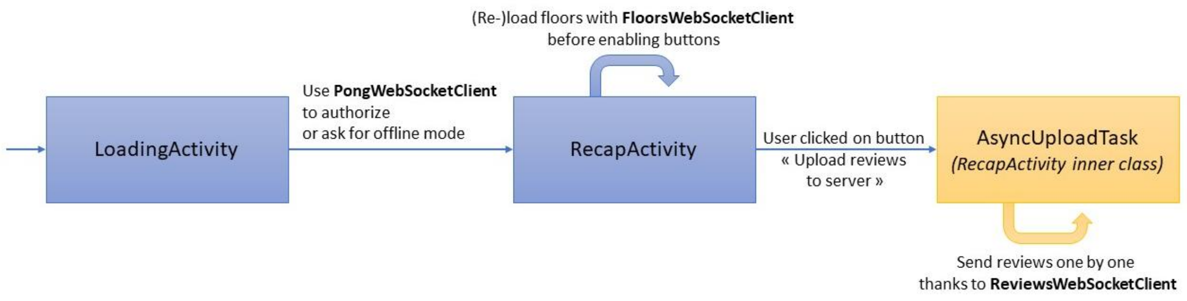

(Re-)load floors with FloorsWebSocketClient before enabling buttons
Use PongWebSocketClient to authorize or ask for offline mode
LoadingActivity
RecapActivity
User clicked on button « Upload reviews to server »
AsyncUploadTask
(RecapActivity inner class)
Send reviews one by one thanks to ReviewsWebSocketClient

Figure 25 Requests in user interactions

3. Dependencies

There are three libraries on which the code is dependent:

- Android Debug Database
- GSON
- Android Websocket Client

The Android Debug Database library can be removed very easily and is only present for debug purposes: it allows access to the application database from a URL.

The other two libraries are crucial for message exchanges with the server.

The Android Websocket Client library has been integrated in a local compiled version ("ReviewApp/WebSocketClient" folder). This made it possible to integrate Alexander

Shirokih's developments when they had not been published on JCenter. It is possible that these developments will be integrated later in the development of Gusavila92 and that a Gradle import from this directory may be sufficient. Beware however, Alexander Shirokih's developments cover the segmentation into fragments of a Websocket package. Without this implementation, it is impossible to send a complete WebSocket package over the network.

Legal Notice

This publication has been produced in the framework of the Interreg cross-border cooperation project C2L3PLAY, co-financed by the European Union. Avec le soutien du Fonds européen de développement régional / Met steun van het Europees Fonds voor Regionale Ontwikkeling.

GoToS3
C2L3Play

This work was produced as part of the FEDER Digistorm project, co-financed by the European Union and the Wallonia Region.