

**Manuel utilisateur
Review App & Review Manager
Version Française, 10/07/2020**



Table des matières

Installation du serveur Unity.....	5
1. Paramètres et stockage de données	5
2. Lancement du programme	5
3. Licences.....	6
4. Langues.....	6
Installation de l'application Android.....	7
1. Paramètres et stockage de données	7
2. Lancement du programme	7
3. Licences.....	8
4. Langues.....	9
Glossaire	10
1. Review	10
2. Floor	10
3. Session	10
Utilisation du serveur Unity	11
1. Principe général.....	11
2. Interface utilisateur.....	11
3. Title screen.....	11
a. Recent.....	11
b. New.....	11
c. Open.....	12
d. Exit	12
4. Project screen	12
a. Barre de menu	13
b. Vue 3D	13
c. Panneau latéral	15
5. Ecrans spéciaux.....	16
a. Add Floor	17
b. Floor Autoscale	17
c. Note Details.....	18
d. Note Filter Edition.....	19

6. Aides.....	19
a. Créer / modifier une image de génération de floor	19
b. Navigation dans les formulaires	21
c. Unités de longueur	22
Utilisation de l'application Android	22
1. Principe général.....	22
2. Loading screen	22
3. Main screen	24
4. Add a review	25
a. Localisation sur un plan	25
b. Photo	26
c. Emotion scoring	26
d. Notes supplémentaires.....	27
5. Liste des reviews	28
6. Détails d'une review	29
7. Upload review.....	30
8. Erreurs potentielles.....	31
Organisation du code Unity	33
1. Interface	33
a. TitleScene	33
b. VisualizerScene	33
2. Fichiers ressources	35
a. Dossier Assets	35
b. Prefabs	36
3. Code C#.....	37
a. Structures de données.....	37
b. Classes utilitaires	38
c. Scripts principaux	39
4. Formatage des messages Websocket	40
a. Services	40
Organisation du code Android.....	43
5. Affichage	43
6. Code Java.....	45

a. Interactions utilisateur	47
b. Base de données	47
c. Requêtes réseau	49
7. Dépendances.....	50
Mentions légales	51

Installation du serveur Unity

Ce programme Windows permet de gérer différents projets, de générer et manipuler des espaces 3D à partir de plans de bâtiments et propose des outils pour collecter et gérer les collections de « Reviews ».

1. Paramètres et stockage de données

Chaque projet est sauvegardé sur le disque dur de l'ordinateur dans un dossier qui porte le même nom que le projet, et avec une hiérarchie précise. On trouve dans chaque dossier de projet :

- Un fichier *simulation.xml* contenant les informations relatives au projet (nom du projet, clé de session, liste des floors). Bien qu'il soit possible d'éditer ce fichier, sa modification peut mener à un crash de l'application au prochain chargement du projet et n'est donc pas recommandée.
- Un dossier *Floors* contenant un sous-dossier pour chacun des floors créés. Dans chaque sous-dossier de floors, on trouve :
 - a) Deux images : *FloorPlan.png* est une copie du fichier image utilisé pour générer le floor et *Heatmap.png* est une représentation de la densité de reviews sur le plan avec une transparence de 32%. Ces deux images ont la même résolution et peuvent être superposées simplement avec un logiciel de traitement d'images.
 - b) Un dossier *Notes* qui contient les reviews reçues pour ce floor. Chaque note est constituée de deux images jpeg (l'image originale reçue et une version de taille réduite dont le nom finit par *_light.jpg* qui est utilisée dans l'application serveur) et d'un fichier JSON contenant les autres données associées la review.

2. Lancement du programme

- Dézipper l'archive zip à l'emplacement de votre choix
- Dans le dossier *ReviewManager*, double cliquer sur *ReviewManager.exe*
- Si une fenêtre concernant le pare-feu apparaît, cocher la case « réseau privé » et éventuellement « réseau public » si le programme est amené à être utilisé sur ce type de réseau puis valider
- Sur la fenêtre *ReviewManager Configuration*, choisir les paramètres d'affichage désirés (paramètres recommandés : 1920x1080, Low, Display1) puis cliquer sur « Play ! »
- Pour créer un raccourci, clic droit sur *ReviewManager.exe*, puis « envoyer vers » -> « Bureau (créer un raccourci) »

3. Licences

Le programme a été réalisé sur le moteur de jeu Unity (version 2018.4). Il ne peut à ce titre pas être vendu sans licence payante Unity. Son utilisation reste gratuite.

Le code du programme est mis à disposition en Open Source, sous licence GPLv3.

En plus du framework .NET, deux plugins et trois scripts Open Source ont été utilisés.

- Plugin : UnitySimpleFileBrowser
Utilitaire de navigation de fichiers pour l'interface utilisateur
<https://github.com/yasirkula/UnitySimpleFileBrowser>
v1.1 (octobre 2019)
Licence MIT
- Plugin : websocket-sharp
Nécessaire au fonctionnement du serveur websocket
websocket-sharp.dll, compilé depuis <https://github.com/sta/websocket-sharp>
Commit 46f464910621054d6d68df0d5610b5409de9b580 (17 mars 2020)
Licence MIT
- Script : TextureScale.cs
Script de redimensionnement de textures et d'images
<http://wiki.unity3d.com/index.php?title=TextureScale&oldid=19637>
Version du 14 octobre 2016
Publié sous licence Creative Commons Attribution Share Alike (CC BY-SA 3.0)
- Script : TextureFilter.cs
Bibliothèque de fonctions de traitement d'images
<https://github.com/andrewkirillov/AForge.NET>
rev 1732
LGPL v3
- Script : MeshGenerator.cs
Génération de meshes à partir d'une texture
<https://github.com/SebLague/Procedural-Cave-Generation>
Commit 12f0b07d6f273e4dd12db937f939482b03eb69b4 (19 avril 2015)
Licence MIT

4. Langues

L'application a été développée exclusivement en anglais. Comme il s'agit d'un logiciel prévu pour être utilisé par une personne avertie, le support multilingue n'a pas été envisagé. Une traduction nécessiterait de modifier l'interface (projet Unity), ainsi que des chaînes de caractères dans les fichiers de code (C#).

Installation de l'application Android

Le code fonctionne sous Android. Il a été développé pour fonctionner normalement d'Android 5.0 (API 21, Lollipop, 2014) à Android 10.0 (API 29, Q, 2020).

Pour fonctionner correctement, l'application nécessite l'accès à une connexion internet, ainsi qu'à l'appareil photo et à la galerie photo.

L'affichage a été pensée et conçue à partir d'un Samsung A20e (6"4, résolution de 720x1560) sous Android 10. Le développement ne s'est réalisé que pour une version de l'application en mode portrait (le changement d'orientation a été bloqué dans l'application). Il est possible que l'affichage diffère sensiblement selon l'appareil utilisé et ses dimensions.

1. Paramètres et stockage de données

Les photos prises et téléchargées peuvent être retrouvées en connectant le téléphone à un ordinateur et en naviguant dans la partie data de l'application : « Phone/Android/data/com.numediart.reviewapp/files ».

Les floors téléchargés sont disponibles dans le dossier Download, rangés par clef de session. Les photos des Reviews sont disponibles dans le dossier Pictures, chaque nom correspondant à l'id de la Review. Dans le cas où l'utilisateur aurait utilisé une photo de sa galerie pour une Review, celle-ci sera dupliquée dans le dossier de l'application.

Attention cependant à ne pas supprimer ces différents fichiers ou dossiers car les accès à ceux-ci pour l'application sont régis par une base de données.

Si l'utilisateur souhaite libérer de l'espace mémoire, il est invité à désinstaller l'application ou à « supprimer les données » dans les paramètres de stockage. Attention cependant, cela supprimera toutes les données, y compris les images téléchargées et les Reviews qui n'auraient pas encore été envoyées vers le serveur.

2. Lancement du programme

Il existe deux versions compilées de l'application (APK), une version « debug » et une version « release ». La première version permet d'accéder au bouton « Generate Reviews » afin, notamment de vérifier la charge que le serveur est capable de supporter. La seconde ne dispose pas de ce bouton. Dans les deux cas, l'installation se déroule de la même manière :

- Connecter le téléphone à un ordinateur via le port USB
- Sur le téléphone, un message demande si l'on souhaite autoriser l'accès aux données du téléphone apparaît, appuyez sur « autoriser »
- Sur l'ordinateur, ouvrez l'explorateur de fichiers (via l'icône « Ce PC » par exemple) et accédez aux dossiers du téléphone. Cliquez sur l'icône du téléphone, puis sur

- « Phone » ou un nom équivalent. Lorsque vous arrivez sur plusieurs dossiers, choisissez le dossier « Download » et, une fois à l'intérieur, copiez-y l'APK.
- A ce stade, vous pouvez d'ores et déjà « éjecter le téléphone » et le débranchez du câble USB.
 - Sur le téléphone, cherchez l'explorateur de fichiers. Il peut parfois être caché dans des sous-dossiers et devrait posséder un nom comme « Mes fichiers ». Une fois dans cette application, cherchez le fichier que vous venez de transférer (via « fichiers récents » ou « fichiers install. » par exemple).
 - Appuyez sur le fichier pour lancer l'installation.
 - Il est possible qu'un message d'alerte s'affiche en indiquant que l'installation d'applications inconnues n'est pas autorisé, appuyez sur paramètres et cochez la case « Autorisation depuis cette source ». (N.B. une fois l'installation réalisée, il est tout à fait possible de décocher à nouveau cette option pour votre sécurité).
 - Un message demandant si vous souhaitez installer devrait apparaître, appuyez sur « Installer ».
 - Un warning « Play Protect » devrait s'afficher, appuyez sur « Installer quand même »
 - Un message demandant de « Faire analyser l'application ? » devrait apparaître, choisissez l'une des deux options.
 - Une fois l'installation terminée, retournez dans l'affichage rassemblant l'ensemble des applications et recherchez l'application « Review App » (ou « Review App Debug » pour la version Debug)

3. Licences

Le code a été développé en Open Source, en GPL v3, les librairies utilisées se basent sur des licences compatibles.

En dehors des librairies appartenant à Android SDK, les libraires suivantes ont été utilisées :

- 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
(Version compilée du fork d'Alexander Shirokih en complément au code original)

de Gusavila92 : <https://github.com/gusavila92/java-android-websocket-client> en date du 11 janvier 2020, également en Apache 2.0)

L'Android Debug Database n'est présente qu'à des fins de debug et il peut être envisagé de retirer celle-ci dans une version finale de l'application.

4. Langues

L'application a été développée exclusivement en anglais. Tous les textes apparaissant à l'utilisateur sont rassemblés dans le fichier « res/values/strings.xml » et il est possible, moyennant traduction de ces termes, de rendre l'application multilingue.

Glossaire

1. Review

Également appelée « Note » dans la partie code du serveur, elle est, dans son ensemble et avec toutes ses composantes (photos, localisation, score, titre, explications, ...), telle que l'utilisateur l'a enregistrée dans l'application Android et telle qu'elle sera envoyée au serveur.

2. Floor

Plan hébergé sur le serveur pour chaque projet, permettant la localisation de chaque note, grâce à des coordonnées relatives à l'image du plan.

3. Session

Définie par la clef de session, elle permet d'identifier un projet de manière unique, peu importe le nom d'utilisateur, l'adresse IP et le port du serveur.

Utilisation du serveur Unity


1. Principe général


Ce programme Windows permet de gérer différents projets, de générer et manipuler des espaces 3D à partir de plans de bâtiments et propose des outils pour collecter et gérer les collections de « Reviews ».

2. Interface utilisateur

Les composants principaux de l'interface utilisateur sont les boutons et les champs d'entrée texte. Ils sont systématiquement de forme rectangulaire, avec une couleur de fond gris clair pour les boutons et blanc pour les champs texte. Le composant qui a le focus (quand la souris passe au-dessus ou un champ texte que l'on modifie) prend une couleur cyan. Un composant inactif avec lequel on ne peut pas interagir prend une couleur gris foncé. Les champs textes sont presque toujours accompagnés d'une étiquette qui précise le contenu attendu.

Bouton actif : 

Champs d'entrée texte actif : 

Bouton ayant le focus : 

Bouton inactif : 

D'autres composants d'interface sont aussi utilisés (curseurs sur des images, boîtes à cocher) mais leur utilisation est spécifique et sera détaillée plus loin dans ce document.

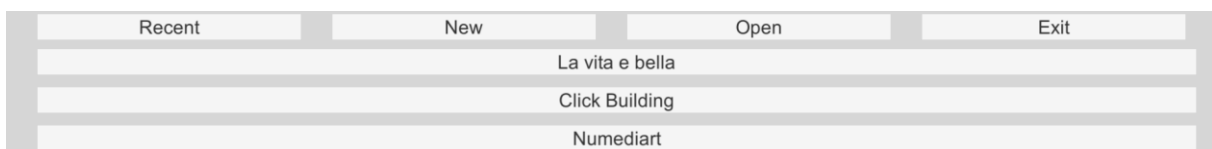
3. Title screen

Lors du premier démarrage de l'application, vous arrivez sur l'écran de gestion des projets. Il est constitué de 4 onglets.



a. Recent

C'est l'onglet affiché au lancement de l'application. Il affiche les projets chargés récemment. Pour ouvrir un projet récent, il suffit de cliquer sur son nom. Pour supprimer un projet de cette liste, il faut le déplacer ou le renommer sur le disque dur via l'explorateur Windows et relancer l'application.



b. New

Cet onglet permet de créer un nouveau projet. Pour cela :

- 1) Il faut entrer un nom dans le champ texte *Project Name*. Il n'est pas possible de donner à un nouveau projet le même nom que celui d'un projet déjà existant (affiché dans l'onglet *Recent*), dans ce cas un message d'information sera affiché en bas de l'écran.
- 2) Choisir une clé de session unique dans le champ *Session Key*. Cette clé sera utilisée par l'application Android pour se connecter à l'ordinateur. Le champ est initialisé avec une chaîne aléatoire de quatre lettres majuscules. Vous pouvez cependant la changer et utiliser des chiffres ou des lettres minuscules mais la longueur de la chaîne est limitée à quatre caractères.
- 3) Par défaut, le projet sera créé dans le dossier « Mes Documents » mais vous pouvez choisir un autre dossier et cliquant que le bouton *Change Directory*.
- 4) Le bouton *Start* permet de valider le formulaire et de créer le dossier du projet sur le disque dur.

The screenshot shows the 'New' tab of the application. At the top, there are four tabs: 'Recent', 'New', 'Open', and 'Exit'. Below the tabs, the 'Project Name' field is empty. The 'Session Key' field contains the text 'EWSO'. The 'Preferred Directory' field shows 'D:\Fabien\Documents'. Below this field is a 'Change Directory' button. At the bottom of the form is a 'Start' button.

c. Open

Ce bouton ouvre une fenêtre de dialogue qui permet de naviguer jusqu'au fichier *simulation.xml* du projet que l'on veut ouvrir.

d. Exit

Ce bouton permet de fermer l'application.

4. Project screen

Cet écran est constitué de 3 zones d'interactions.



a. Barre de menu

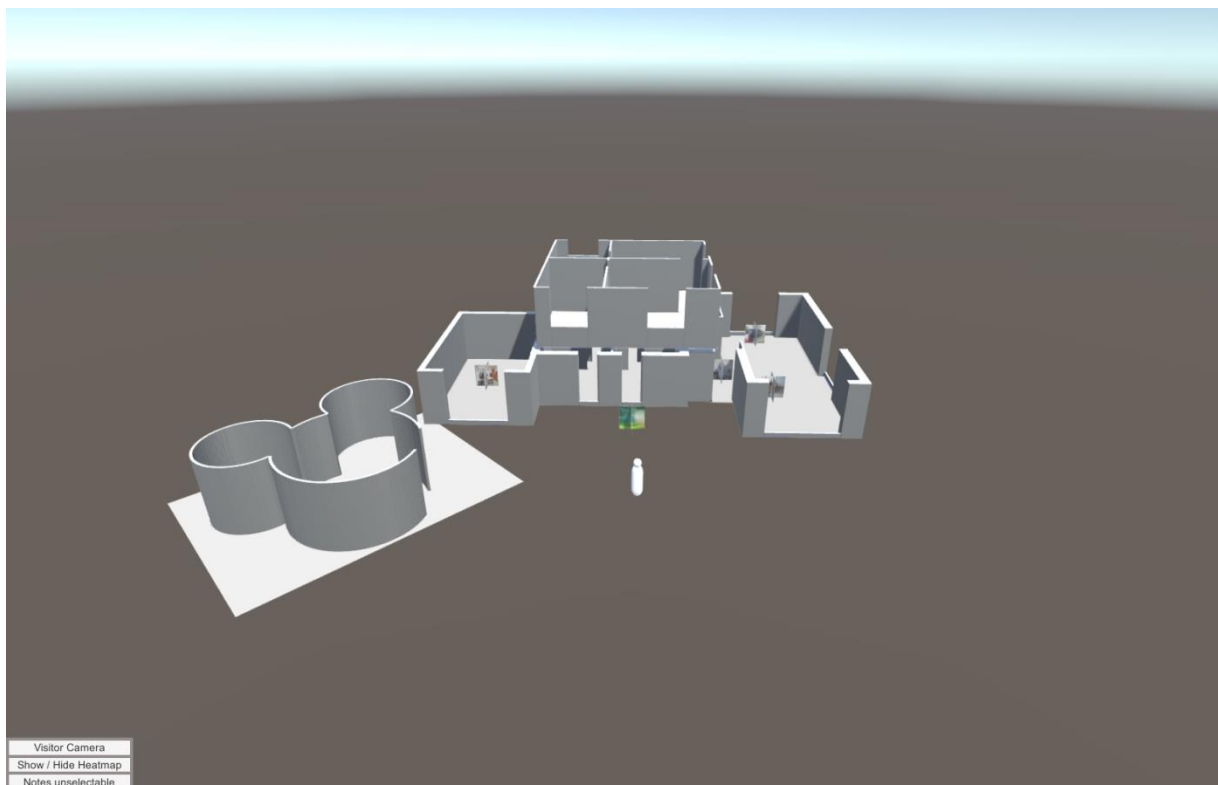
La barre de menu se trouve en haut de la fenêtre de l'application. On y trouve de gauche à droite :



- 1) Bouton Title : retour à l'écran d'accueil
- 2) Bouton Exit : fermeture de l'application (également possible en appuyant sur la touche echap du clavier)
- 3) Project : le nom du projet actuellement ouvert
- 4) Un bouton Start / Stop Server : démarrage / arrêt du serveur websocket. Le serveur ne peut communiquer avec l'application Android et donc transmettre les plans des floors et recevoir des reviews que lorsqu'il est démarré.
- 5) Un voyant rouge quand le serveur est arrêté, vert quand il est démarré.
- 6) Les informations relatives au serveur (Adresse IP, port, clé de session), à entrer sur l'écran d'accueil de l'application Android pour pouvoir se connecter au serveur et envoyer des reviews.
- 7) Bouton Notes Panel : déroule / repli le panneau d'affichage des notes. Voir la section c pour plus d'information.

b. Vue 3D

La vue 3D prend la plus grande partie de l'écran. Elle permet d'afficher le modèle 3D du bâtiment qui a été généré à partir des plans de chaque floors, de se déplacer dans ce modèle et d'afficher les images des reviews à l'emplacement où elles ont été localisées sur le plan.



Boutons de contrôle de la vue 3D

En bas à gauche de la vue 3D se trouvent 3 boutons.

- *Visitor / Omniscient Camera* : permet de basculer entre le contrôle par caméra omnisciente et le contrôle par caméra visiteur (voir sous-section suivante pour plus de détails).
- *Show / Hide Heatmap* : choisir entre l'affichage du sol tel que défini par l'image de plan et l'affichage de la carte de densité des reviews.
- *Notes unselectable / selectable* : choisir si un clic sur une des notes de la vue 3D permet d'ouvrir le panneau de détails sur cette note. La note actuellement pointée est mise en évidence par une surbrillance jaune.

Interactions caméra

Il existe 2 caméras dans la vue 3D.

- 1) La caméra omnisciente (sélectionnée par défaut à l'ouverture d'un projet) est représentée par une sphère blanche dans l'espace 3D et donne un contrôle sur la vue 3D avec la souris.
 - Le clic gauche maintenu contrôle la translation gauche / droite et avant / arrière
 - Le clic droit maintenu contrôle la translation gauche / droite et haut / bas
 - La molette permet de faire un zoom avant ou zoom arrière
 - Le clic maintenu sur la molette contrôle la rotation de la caméra
- 2) La caméra visiteur est représentée par un cylindre blanc surmonté d'une sphère blanche (visible sur la capture d'écran ci-dessus) et permet de se déplacer dans l'espace 3D grâce aux touches du clavier.
 - 'z' ou flèche vers le haut : avancer
 - 's' ou flèche vers le bas : reculer
 - 'q' ou flèche gauche : pivoter vers la gauche

- 'd' ou flèche droite : pivoter vers la droite
- 'Majuscule' ou 'PageUp' : monter
- 'Control' ou 'PageDown' : descendre

c. Panneau latéral

Floors

Ce panneau affiche une liste des floors créés dans le projet, avec pour chacun le nom qui lui a été attribué à sa création, ainsi qu'un nombre attribué automatiquement par le programme. Chacun des floors peut être sélectionné en cliquant dessus (un seul floor sélectionné à la fois, hormis lorsque le *Notes Panel* est déployé). Un floor sélectionné est de couleur violette.

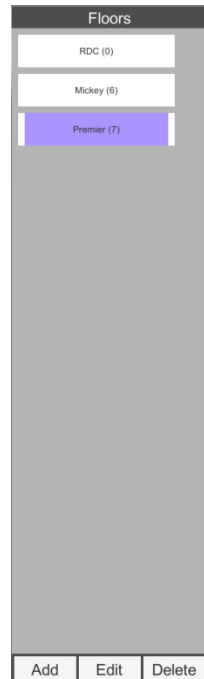
En-dessous de la liste des floors se trouvent trois boutons.

Add : Ouvre l'écran de création de floor. Voir la section 5.a.

Edit : Déploie le panneau d'édition des floors. Voir la suite de cette section pour plus de détails.
(inactif si aucun floor n'est sélectionné)

Delete : Ouvre une fenêtre de dialogue pour confirmer que l'on désire supprimer ce floor.
(inactif si aucun floor n'est sélectionné)

ATTENTION : Supprimer un floor supprime son répertoire sur le disque dur, ce qui inclut TOUTES les reviews qui lui sont associées.



Notes Panel

Cliquer sur le bouton *Notes Panel* de la barre de menu permet de déployer ou replier ce panneau. Dans ce dernier, sont affichées les reviews de tous les floors actuellement sélectionnés, avec pour chacune une miniature de la photo, le titre qui lui a été donné et l'auteur entre parenthèses.

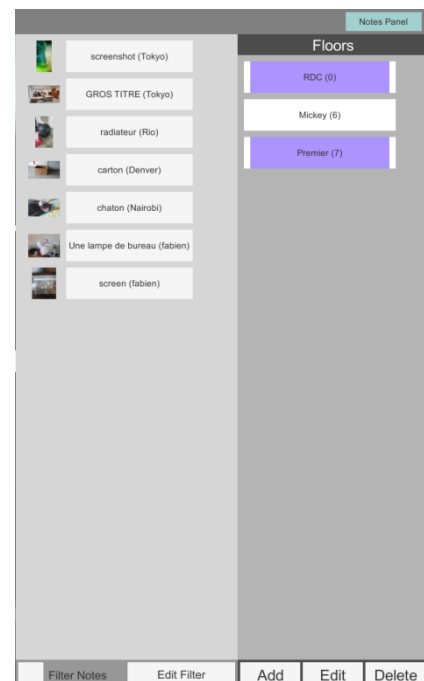
Pour sélectionner plusieurs floors, il faut maintenir la touche Control du clavier enfoncée et cliquer sur les floors que l'on souhaite sélectionner. Cliquer à nouveau sur un floor sélectionné le désélectionne.

Il est possible d'afficher une review et toutes les informations associées en cliquant sur le titre de celle-ci. Voir la section 5.c pour plus de détails.

En bas du panneau, se trouvent une case à cocher et un bouton.

Filter Notes : Si la case est cochée, le filtre est appliqué et seules les reviews qui correspondent à ce filtre sont affichées dans la liste et dans la vue 3D. Si la case est décochée, toutes les reviews des floors sélectionnés sont affichées.

Edit Filter : Ouvre l'écran d'édition du filtrage des reviews. Voir aussi la section 5.d.



Edit Panel

Lorsqu'un floor est sélectionné, cliquer sur le bouton *Edit* déploie ou replie ce panneau. Si le *Notes Panel* était ouvert, *Edit Panel* le remplace et vice-versa.

Ce panneau est divisé en deux parties.

Celle du haut permet de modifier les valeurs données à la création du panneau. La prise en compte des modifications faites sur cette partie nécessite de cliquer sur le bouton *Update 3D Model*, sans quoi ces modifications seront perdues à la fermeture du panneau.

Floor Name : Champ d'entrée texte pour modifier le nom du floor

Floor ID : (non modifiable) identifiant du floor pour le programme

Change Image : Bouton qui ouvre une fenêtre de dialogue pour changer l'image du plan du floor.

ATTENTION : si la nouvelle image a un ratio largeur / hauteur différent de la précédente, les reviews ne se trouveront pas au bon emplacement. De plus, si l'image est modifiée alors que des applications android sont connectées au serveur, elles n'auront pas la bonne version du plan.

Black / Red / Blue / Green height : Hauteur des murs et autres éléments dans le modèle 3D en fonction de la couleur sur l'image de floor

Update 3D Model : Valider les changements, sauvegarder les données du projet et régénérer le modèle 3D du floor

La seconde partie contient les contrôles permettant de modifier la position et l'échelle du modèle 3D. Les changements faits sur cette partie sont immédiatement pris en compte dans la vue 3D et sauvegardés.

Position X / Z : Translation du modèle 3D sur l'axe gauche/ droite ou avant/arrière par rapport à l'origine. A sa création, le modèle est centré en $X = 0$, $Z = 0$.

Altitude : Hauteur à laquelle la base du modèle 3D doit se trouver

Rotation : Tourne le modèle autour de son centre, valeur en degrés

Scale et bouton Autoscale : Facteur multiplicatif d'échelle à appliquer sur les dimensions X et Z. Pour plus de simplicité, un utilitaire permet de calculer automatiquement ce facteur. Il s'ouvre en cliquant sur le bouton Autoscale. Voir la section 5.b pour les détails.

5. Ecrans spéciaux

En plus de cet écran Projet, certains contrôles font apparaître des écrans spécifiques à une tâche. En plus des écrans de confirmation qui sont assez explicites, le fonctionnement de certains écrans est précisé ci-dessous.

a. Add Floor

Sur ce formulaire, on retrouve les mêmes informations à donner que sur la première partie de l'Edit Panel.

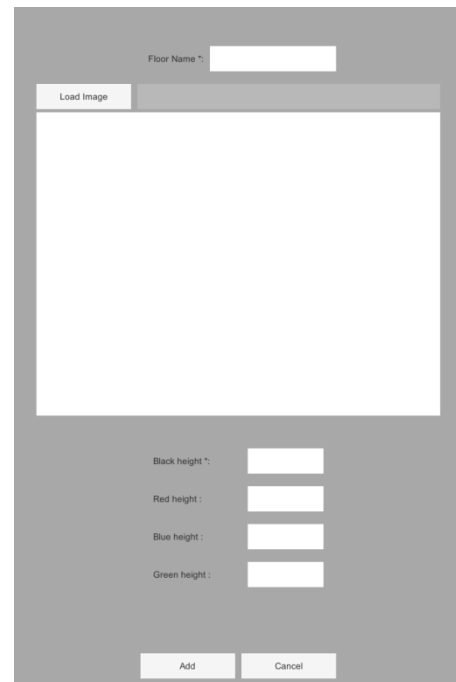
Floor Name : Champ d'entrée texte pour donner un nom au nouveau floor

Load Image : Bouton qui ouvre une fenêtre de dialogue pour choisir l'image de floor. Pour connaître les bonnes pratiques à respecter pour créer cette image, voir la section 6.a.

Black / Red / Blue / Green height : Hauteur des murs et autres éléments dans le modèle 3D en fonction de la couleur sur l'image de floor

Add : Valider les changements, sauvegarder les données du projet et générer le modèle 3D du floor

Cancel : Annuler les modifications et revenir à l'écran Projet.



b. Floor Autoscale

Cet outil permet de calculer le rapport d'échelle à appliquer au modèle 3D du floor pour que sa représentation dans l'espace 3D soit « naturelle ».



Go Back : Annuler les modifications et revenir à l'écran Projet

Instructions : rappel des instructions pour utiliser cet outil

Point A / B : Position des points A et B en pixels sur l'image de plan, par rapport à l'angle en bas à gauche. Il est possible de déplacer les curseurs en cliquant dessus et en glissant la souris jusqu'à la position désirée ou en utilisant les boutons Select.

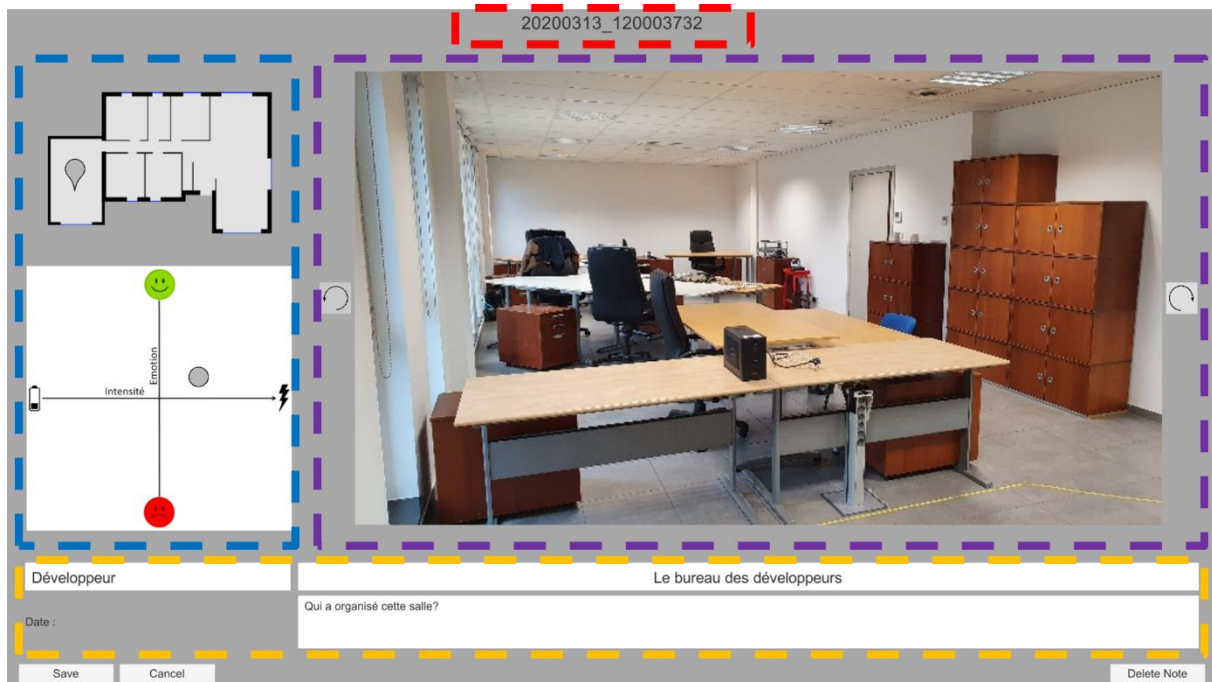
Select : Transformer le curseur de la souris en croix de la couleur du point à sélectionner. Cliquer sur l'image pour placer ce point.

Distance : Espacement entre les points dans l'espace réel – longueur de la ligne rouge (en mètres par exemple).

Autoscale : Calculer le facteur d'échelle et revenir à l'écran Projet.

c. Note Details

Cet écran présente les informations associées à une review et permet de les modifier. Les fonctions des différentes zones de cet écran sont précisées ci-dessous.



En rouge : Nom du fichier JSON associé à la review. Il s'agit de la date à laquelle la review a été reçue par le serveur.

En bleu : Représentation de l'emplacement de la review par rapport au plan du floor et représentation du curseur émotion sur le diagramme valence / intensité. Ces deux curseurs peuvent être déplacés en cliquant dessus puis en les glissant à l'emplacement désiré.

En violet : Zone de présentation de l'image de la review. Cette image est en basse définition pour ménager la mémoire de l'ordinateur. La version originale est stockée sur le disque dur. Deux boutons présentant des flèches permettent de faire pivoter l'image de 90 degrés si nécessaire.

En orange : Les champs textes associés à la review.

En haut à gauche de la zone se trouve le champ Auteur.

En haut à droite, le champ Titre.

En bas à gauche, le champ Date (date de création de la note sur l'application Android), non modifiable.

En bas à droite, le texte descriptif de la review.

Autres boutons :

Save : sauvegarder les modifications et revenir à l'écran Projet.

Cancel : ignorer les modifications et revenir à l'écran Projet.

Delete : Supprimer la review et l'image associée.

d. Note Filter Edition

Cet écran permet de personnaliser le filtre appliqué sur les reviews dans le *Notes Panel*, si la case *Filter Notes* est cochée.

La partie haute permet de filtrer selon les auteurs.

Select All : sélectionner tous les auteurs

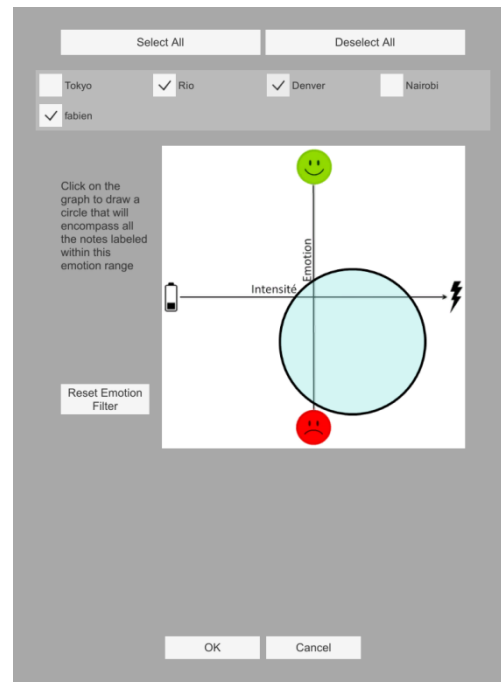
Deselect All : ne sélectionner aucun auteur

Liste des auteurs avec cases à cocher : sélectionner spécifiquement un ou des auteurs. Si aucun auteur n'est sélectionné, le filtrage sur les auteurs ne s'applique pas et le résultat est le même que s'ils étaient tous sélectionnés.

La partie centrale permet de filtrer selon l'émotion associée à la review. Pour cela, il faut cliquer au centre de la zone d'émotion à sélectionner puis maintenir le bouton de la souris enfoncé en glissant pour ajuster la taille du cercle. Toutes les reviews dont le curseur émotion se trouve dans ce cercle seront ainsi gardées. Pour ne pas utiliser ce filtre, cliquer sur le bouton *Reset Emotion Filter*.

OK : valider le filtre et revenir à l'écran *Projet*

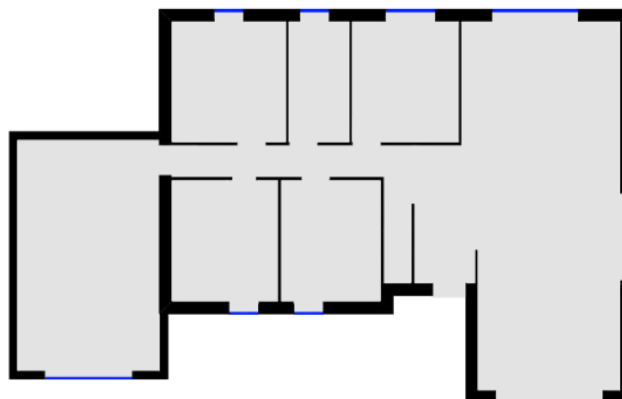
Cancel : ignorer les modifications faites sur le filtre et revenir à l'écran *Projet*



6. Aides

a. Créer / modifier une image de génération de floor

Ici est présenté un exemple de plan adéquat pour la génération de modèle 3D. Ce processus est appelé extrusion.



L'image doit être au format PNG ou JPEG. Le format PNG est préférable car il permet une gestion plus fine de la transparence (canal alpha) et n'introduit pas d'artéfacts de compression à l'enregistrement.

La résolution de l'image ne doit pas être trop élevée pour limiter l'utilisation de la mémoire et le temps de calcul nécessaire à la génération du modèle 3D. Typiquement, la plus grande dimension de l'image devrait toujours être inférieure à 1200 pixels.

Le fond de l'image doit être blanc ou gris clair et sans inscriptions, il correspond à des zones qui ne seront pas extrudées. Il est possible de détourer le bâtiment et de laisser l'extérieur transparent (la transparence doit être appliquée sur du blanc - #00FFFFFF). Cette option est pratique pour les niveaux supérieurs et permet de créer des étages qui s'alignent parfaitement.

Un code couleur permet de créer jusqu'à 4 hauteurs d'extrusion différentes. Les murs sont en noir #000000, préférentiellement pleins. Les autres couleurs utilisables sont **rouge** #FF0000, **bleu** #0000FF et **vert** #00FF00. Elles permettent par exemple de représenter des fenêtres (en bleu sur l'image d'exemple) ou du mobilier fixe. Les parties de l'image extrudées ne sont pas modifiables dans la vue 3D.

Pour convertir un plan d'architecte en image exploitable pour le programme, il faudra utiliser un logiciel de traitement d'image et travailler avec des calques.

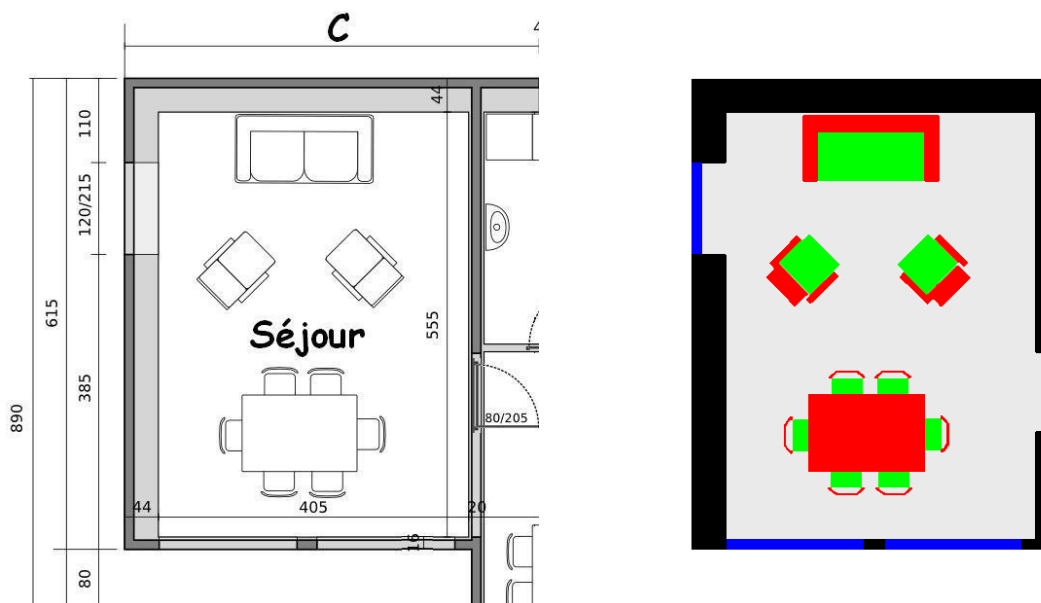


Figure 1 - Exemple de plan adapté pour l'extrusion avec conservation du mobilier fixe

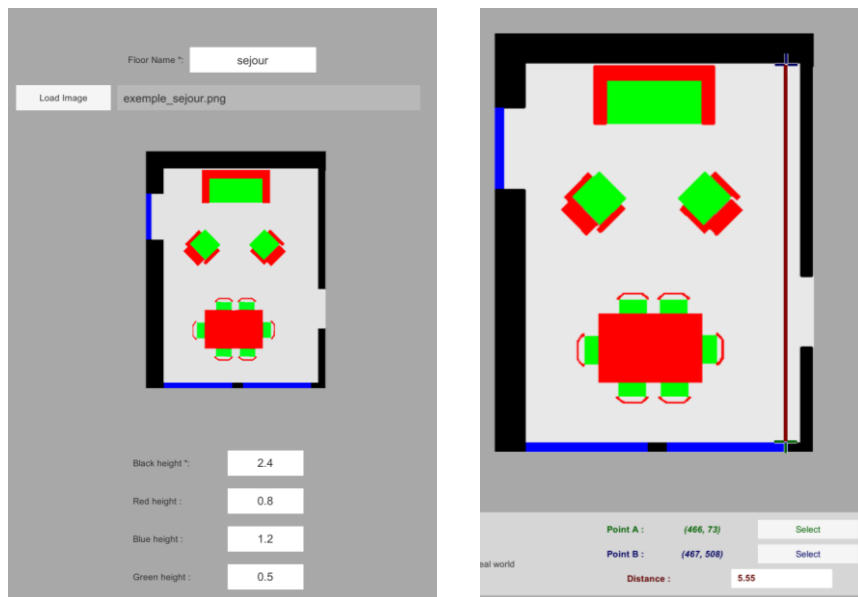


Figure 2 - Création du floor et mise à l'échelle avec l'outil Autoscale

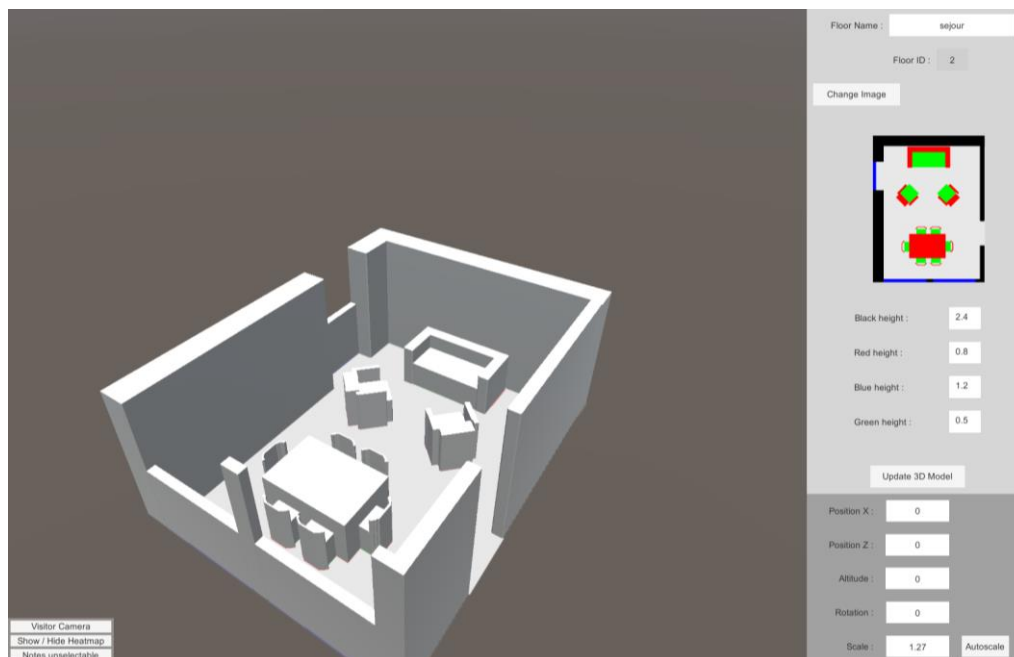


Figure 3 - Résultat dans la vue 3D

b. Navigation dans les formulaires

Dans tous les formulaires, il est possible de passer au champ suivant en utilisant la touche Tabulation. La couleur du champ qui a le focus est cyan.

Dans certains formulaires, des champs sont à remplir obligatoirement. Ils sont en général signalés par un astérisque *. Si un de ces champs n'est pas rempli et que l'utilisateur essaie de valider le formulaire, un message d'information apparaît et le ou les champs concernés deviennent rouges.

c. Unités de longueur

L'unité de longueur choisie pour définir l'échelle du modèle ou la hauteur des murs n'est pas importante en soit (ce peut être des mètres, des centimètres, des pieds, des coudées ou autre), mais elle doit toujours être la même sur le projet. Si ce n'est pas le cas, les modèles 3D seront déformés. Il est conseillé d'utiliser des mètres.

Utilisation de l'application Android

1. Principe général

Cette application permet de se connecter à un serveur afin de lui envoyer des Reviews localisées. L'utilisateur peut créer une Review à la fois, relire les Reviews qu'il a déjà enregistré et les uploader vers le serveur.

2. Loading screen

Lors du premier démarrage de l'application, vous arrivez sur l'écran de chargement. Sur celui-ci, il vous est demandé d'autoriser l'accès aux photos et contenus multimédias. Ceci afin de pouvoir sauvegarder des photos mais également pouvoir éventuellement utiliser celles présentes dans la galerie pour créer des « Reviews ».

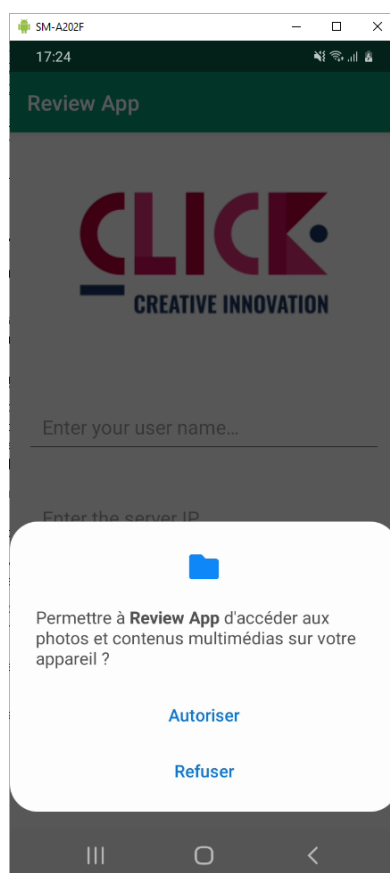


Figure 4 - Premier démarrage

Sur ce premier écran, il vous est demandé de rentrer un nom d'utilisateur, l'adresse IP du serveur ainsi que le port qu'il utilise. Enfin, vous devez entrer la clef de session liée au projet (voir la partie « serveur »).

En cas d'erreur dans les paramètres entrés, un message sera renvoyé à l'utilisateur. Si l'application arrive à contacter le serveur, elle passera sur l'écran « Main screen ».

Il est également possible d'utiliser l'application en mode « offline ». Ce mode est prévu pour pouvoir entrer dans une session déjà existante (pour laquelle les « Floors » ont déjà été téléchargés) et continuer de créer des Reviews, cela même si le serveur est actuellement hors de portée de l'application.

A chaque « log in », l'application vérifiera également que les Floors sont à jour et retéléchargera les nouvelles versions.

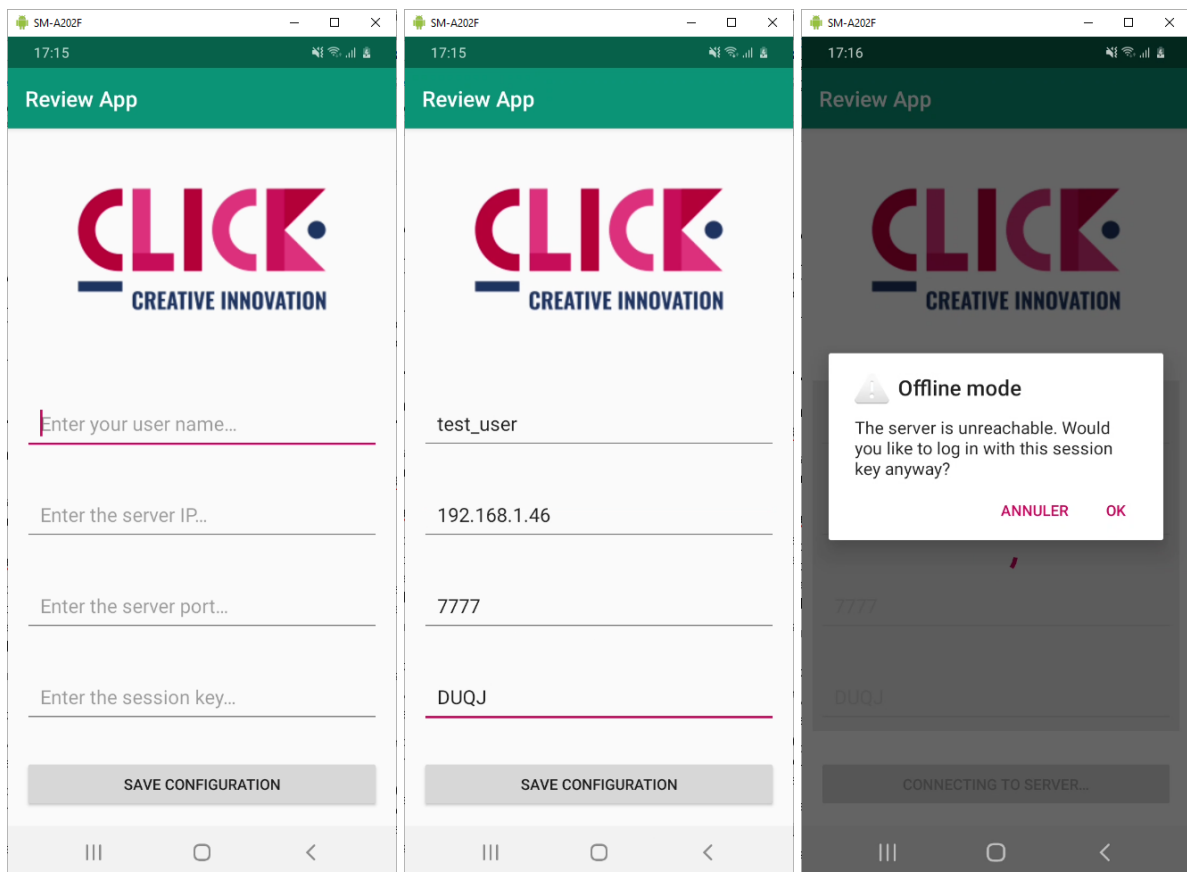


Figure 5 - Loading Screens

3. Main screen

En arrivant sur le Main screen, l'application commencera tout d'abord par mettre à jour les Floors (ou les télécharger pour la première fois). Une fois cela fait, il vous sera possible d'effectuer différentes actions.

Lors de votre première connexion, le bouton « Upload reviews » ne sera pas disponible, mais il le deviendra lorsque des Reviews auront été enregistrées et pourront être envoyées vers le serveur. De la même manière, le bouton « See all reviews » vous amène vers un écran qui récapitulera toutes les Reviews enregistrées pour la session mais qui est pour l'instant vide. Enfin, le bouton « Add a review » vous permet de créer votre première Review. Si vous étiez en train d'enregistrer précédemment une Review, il vous est possible de terminer son édition via le même bouton.

Il est possible de se déconnecter de cette session en utilisant le bouton de retour. Celui-ci vous demandera alors de confirmer votre déconnection.

N.B. : Le bouton en haut à droite « Generate Reviews » n'est disponible que lorsque la variable DEBUG est utilisée dans le code. Il ne sert qu'à générer automatiquement des Reviews par paquet, à des fins de test.

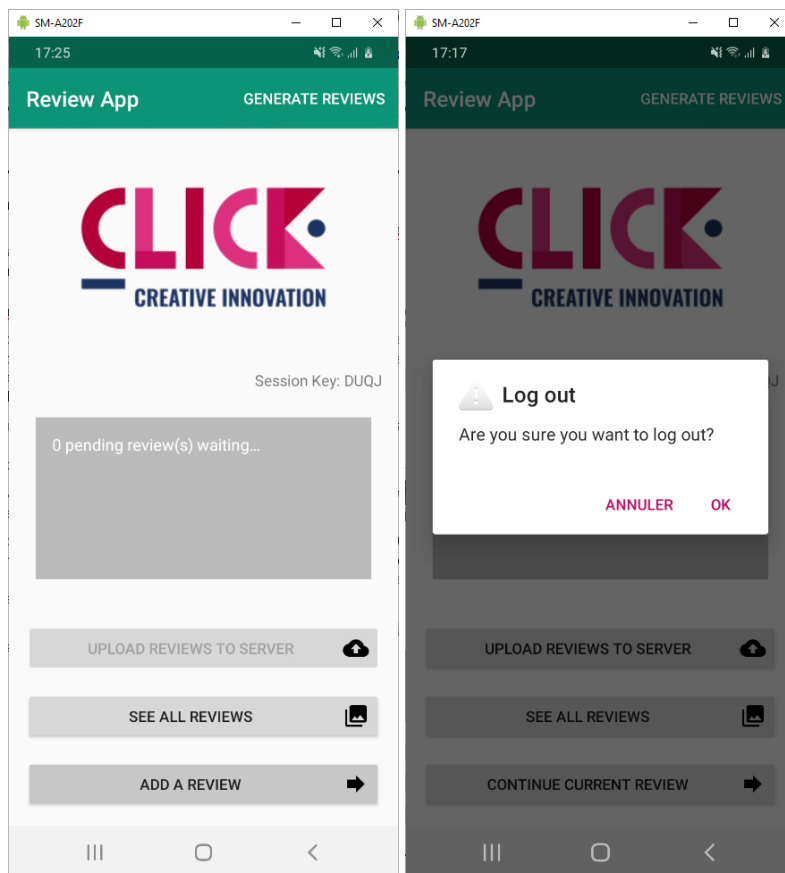


Figure 6 - Main screen

4. Add a review

En cliquant sur le bouton « Add a review » ou « Continue current review », vous passerez systématiquement par les mêmes écrans, dans le même ordre. A chaque étape, les différentes informations sont sauvegardées mais la Review ne sera validée et ne pourra être envoyée qu'en validant chaque étape, ce qui finira par vous ramener sur le Main screen.

a. Localisation sur un plan

Vous arrivez tout d'abord sur l'écran vous permettant de localiser la Review. En cliquant simplement sur le plan, vous pouvez positionner un point indiquant où vous vous trouvez. En « pinchant » avec deux doigts, il est possible de zoomer ou dézoomer le plan en question, ainsi que de le déplacer en glissant un seul doigt.

Dans le cas où plusieurs Floors sont disponibles, vous pouvez naviguer à travers ceux-ci grâce aux flèches directionnelles, pour choisir le plan sur lequel vous souhaitez localiser la Review.

Une fois correctement positionné, vous pouvez cliquer sur « Take a picture » pour passer à la prise de la photo.

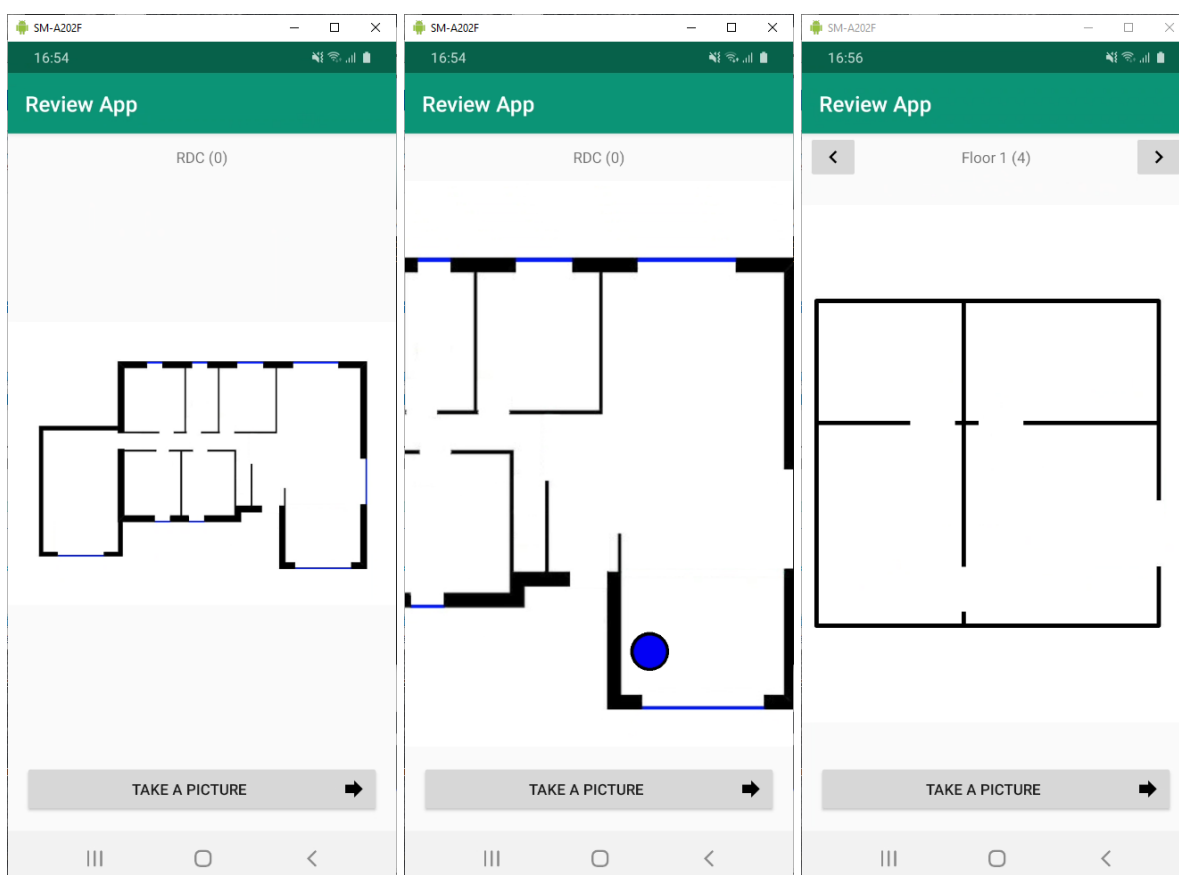


Figure 7 - Locate Screen

b. Photo

Cette partie vous ouvre directement l'appareil photo. Une fois votre photo prise, il vous sera demandé de confirmer celle-ci puis vous arriverez sur l'écran de gestion. A cette étape, il vous est possible de sélectionner une autre photo dans la galerie ou d'effectuer une autre prise.

Lorsque l'appareil photo s'ouvre et si vous ne souhaitez pas prendre de photo (par exemple parce que la photo précédente était meilleure), le bouton de retour vous enverra sur l'écran de gestion. Ceci vous permettra éventuellement de sélectionner une photo dans votre galerie. Un deuxième retour en arrière vous ramènera à l'écran de localisation.

Avant de pouvoir aller plus loin, il sera de toute façon nécessaire de fournir une photo à l'application pour qu'elle puisse l'enregistrer.

Le bouton « Set your mood about it » vous enverra vers l'écran suivant.

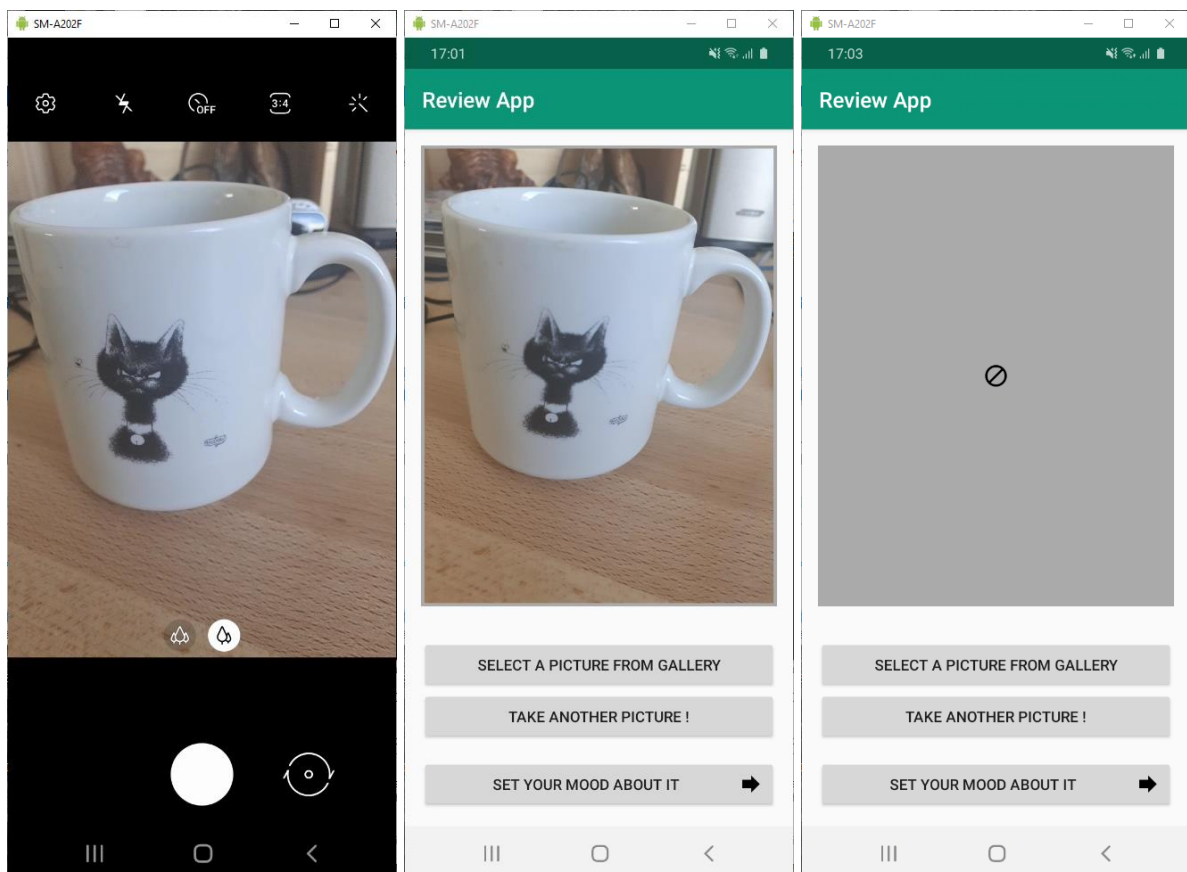


Figure 8 - Picture screens

c. Emotion scoring

Dans cet écran, il vous sera demandé d'indiquer votre humeur par rapport à la photo prise. Un simple clic vous permettra de positionner celle-ci sur le graphique. Le bouton « Give more details » vous emmènera vers les dernières informations à rentrer.

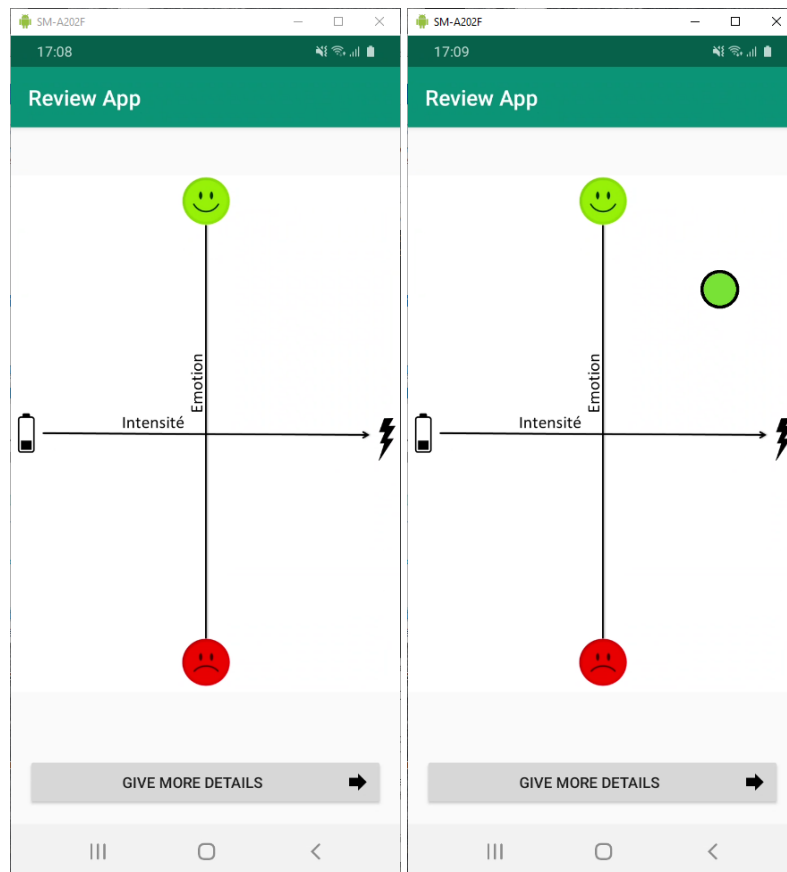


Figure 9 – (Emotion) Score screen

d. Notes supplémentaires

Ici vous pourrez donner un titre à votre Review ainsi que des détails supplémentaires. Restez concis pour le titre car celui-ci s'affichera dans les récapitulatifs (côté application et côté serveur) et permettra d'identifier rapidement chacune des notes.

Une fois le tout complété, il vous reste à valider votre Review en cliquant sur le bouton en bas. En retournant sur le Main screen, un message vous indiquera comme votre Review a bien été enregistrée et est prête à être envoyée vers le serveur.

Il est à noter qu'une fois ce bouton pressé, il vous sera désormais impossible de modifier votre Review.

N.B. : Lorsque vous êtes dans l'édition des détails, le bouton pour refermer le clavier correspond également à celui effectuant un retour en arrière. Afin d'éviter les malencontreux double-clic et perdre ainsi tout le contenu rédigé, le titre et les détails seront sauvegardés même si vous effectuez un retour en arrière !

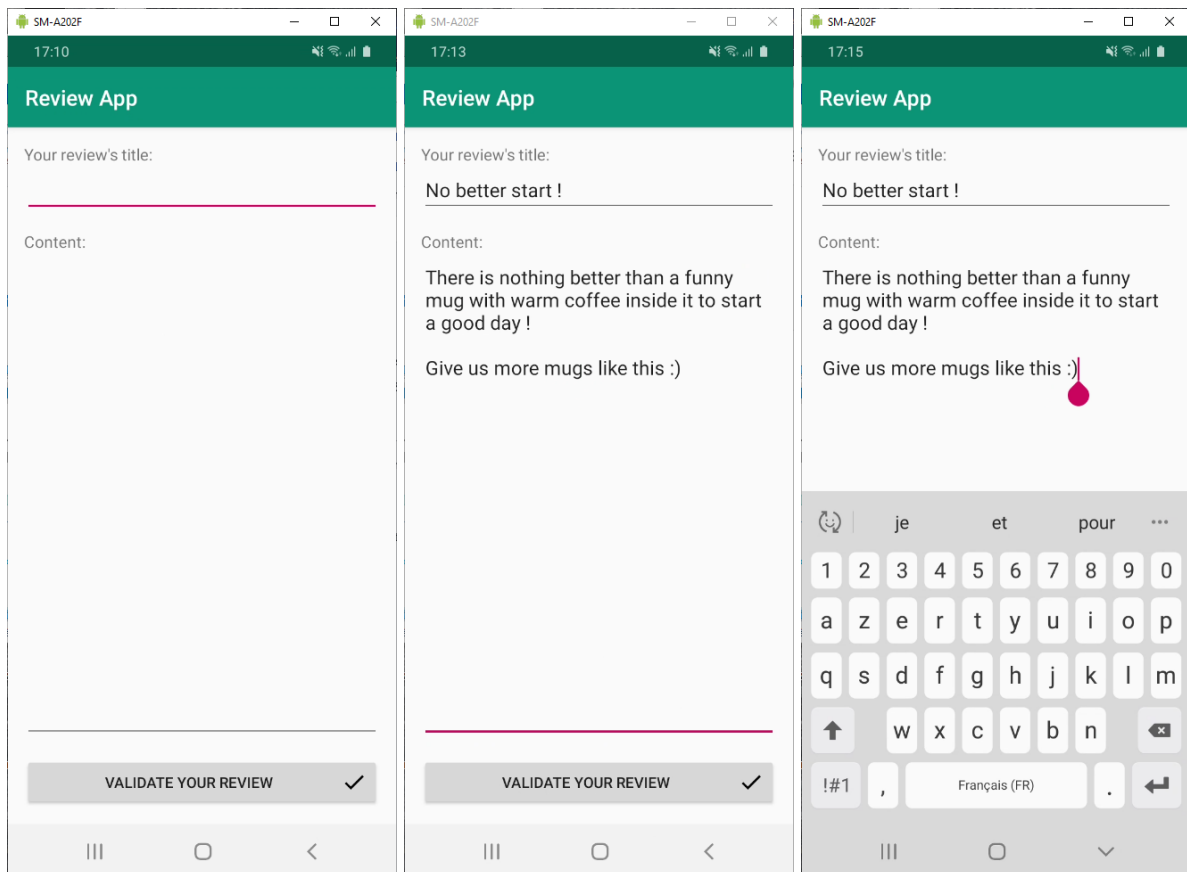


Figure 10 - Notes screen

5. Liste des reviews

Afin de pouvoir retrouver les Reviews que l'utilisateur a écrites, il est possible d'afficher la liste de celles-ci. De retour dans le Main screen, appuyez sur « See all reviews ». L'application vous amènera vers la liste des Reviews enregistrées pour la session actuelle (la clef de session vous est renotée à titre indicatif dans le Main screen).

En swipant vous pouvez voyager à travers toutes les Reviews enregistrées. L'affichage de l'image étant différé, il est possible que vous deviez attendre un peu avant de voir l'icone correspondante apparaître. L'icone « Image » vous indique que l'image n'a pas encore été chargée alors qu'une icône de sens interdit vous indique qu'il n'existe pas d'image pour la Review en question.

De gauche à droite, les différentes informations sont : un chiffre correspondant à l'ID de la Review dans la base de données locale, la photo prise pour la Review, le titre de celle-ci et puis deux icones.

L'icône supérieure (un crayon ou un grand V) indique l'état local de la Review (le crayon indiquant qu'elle est encore en édition, le grand V indiquant qu'elle a été validée par l'utilisateur).

L'icône inférieure (un nuage barré, un symbole d'alerte ou un nuage avec un V à l'intérieur) indique l'état de la Review par rapport au serveur. Le nuage barré signifie que la Review

n'a pas encore été uploadée vers le serveur. A l'inverse, le nuage avec un V indique que cette Review a bien été enregistrée sur le serveur.

Enfin, le symbole d'alerte vous informe qu'il y a eu une erreur lors de l'envoi de cette Review vers le serveur. Lors de la prochaine tentative d'upload, l'application ne tentera de renvoyer cette Review qu'après avoir envoyé toutes celles marquées d'un nuage barré.

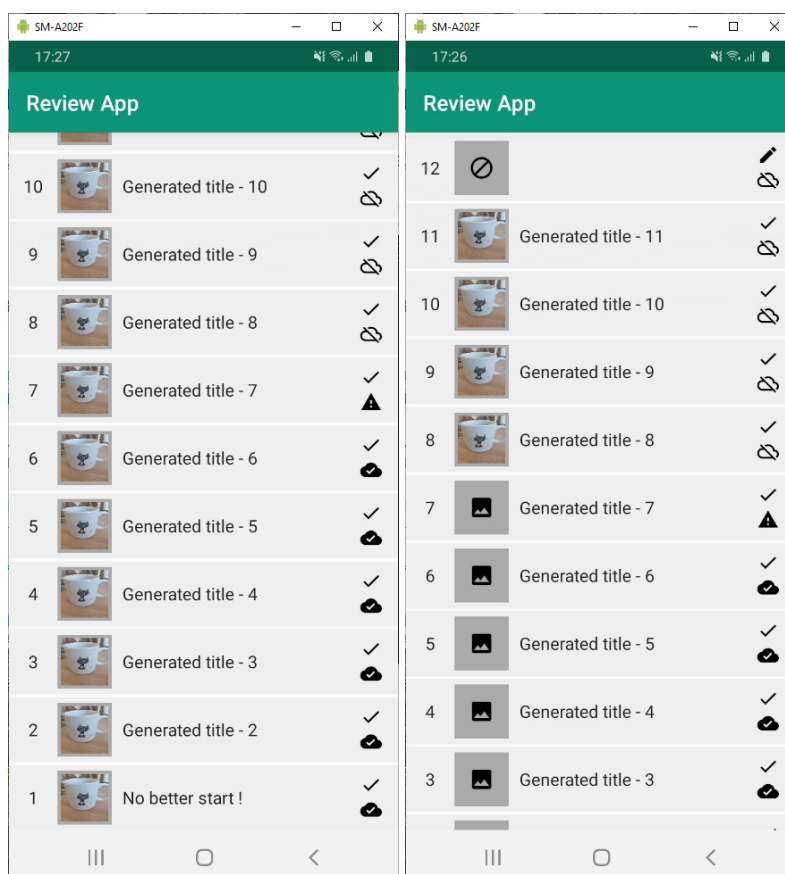


Figure 11 - Reviews List

6. Détails d'une review

Lorsque vous cliquez sur une Review dans la liste, vous passez à son affichage détaillé. De la même manière que pour l'affichage en liste, il est possible que l'image et la localisation prennent quelques secondes avant de s'afficher. Cet écran permet uniquement de visualiser les Reviews mais il ne permet pas de les modifier, peu importe leur état.

Dans cette affichage, en circulant de gauche à droite et de bas en haut, vous pouvez retrouver ces différentes informations :

- La photo enregistrée
- Le titre
- Le nom de l'utilisateur
- La date d'enregistrement

- Le nom du Floor
- L'ID du Floor
- L'image du Floor avec un point bleu représentant la localisation de la Review
- Le graphe d'émotion avec un point bleu représentant l'émotion pour la Review
- Les détails supplémentaires annotés
- L'ID de la Review dans la base de données locale
- La clef de session utilisée pour la Review
- Un indicateur (Complete/Incomplete) pour dire si la Review a été validée localement
- Un indicateur (Not sent/Sent/Error while sent) pour dire si la Review a été uploadée vers le serveur

Enfin, le troisième screenshot ci-dessous vous montre l'état de l'affichage lorsqu'aucune information n'a été rentrée et que toute la Review reste à compléter.

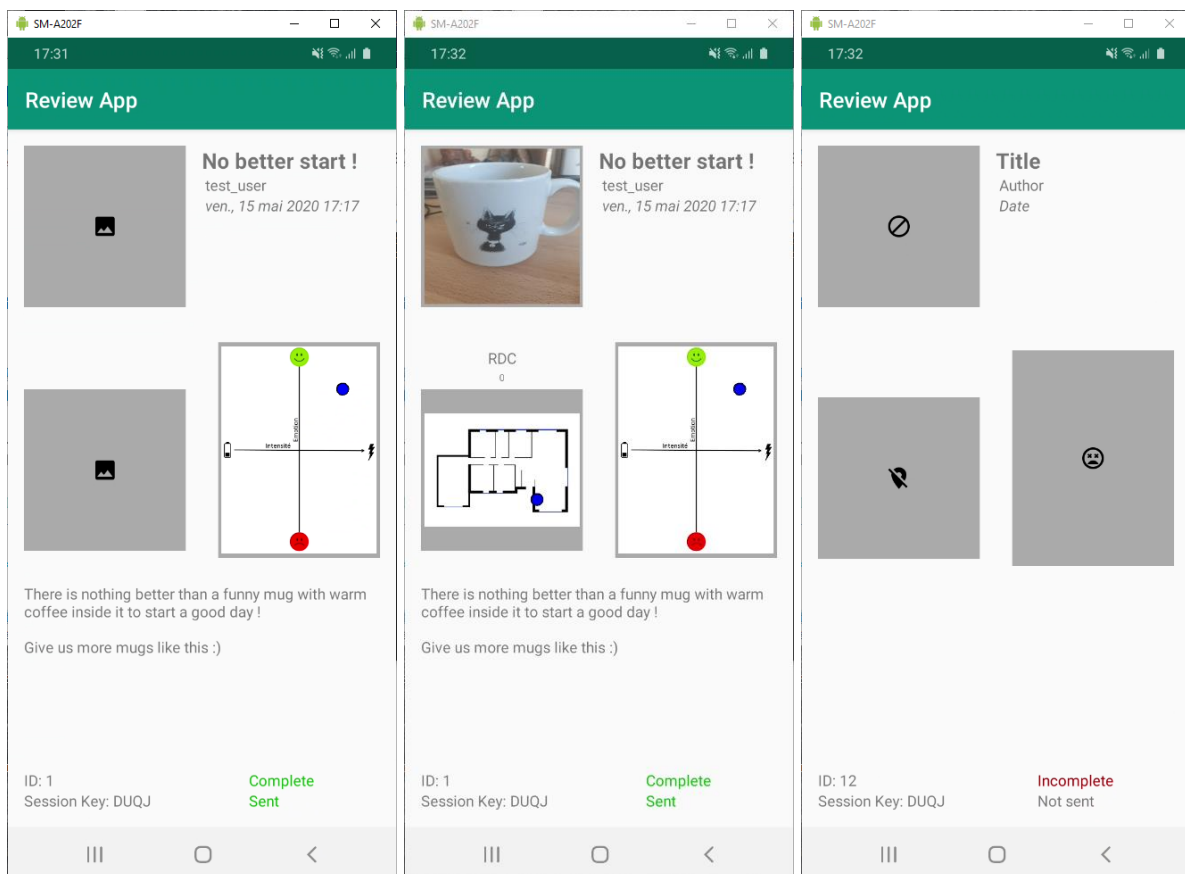


Figure 12 - Detailed Review

7. Upload review

En cliquant sur le bouton « Upload Reviews to server », vous démarrerez la liaison avec le serveur et commencerez à uploader les Reviews vers celui-ci. Lorsque l'upload fonctionne,

une progress bar circulaire s'affiche au milieu de l'écran et le bouton Upload devient indisponible.

Au fur et à mesure, le nombre de « pending reviews » est mis à jour. De même, si des Reviews passent en erreur, l'affichage sera actualisé. L'upload sera arrêté si l'utilisateur quitte cet écran (in extenso : s'il se déconnecte, crée une nouvelle review, affiche la liste des reviews).

Lorsque l'application rencontre une erreur pour une Review, l'upload s'arrête et un message d'erreur est affiché. Il est alors possible à l'utilisateur de relancer l'upload. Dans le cas présent, l'application informe que l'utilisateur essaye d'uploader une Review pour laquelle le Floor n'existe plus (l'administrateur l'a supprimé sur le serveur).

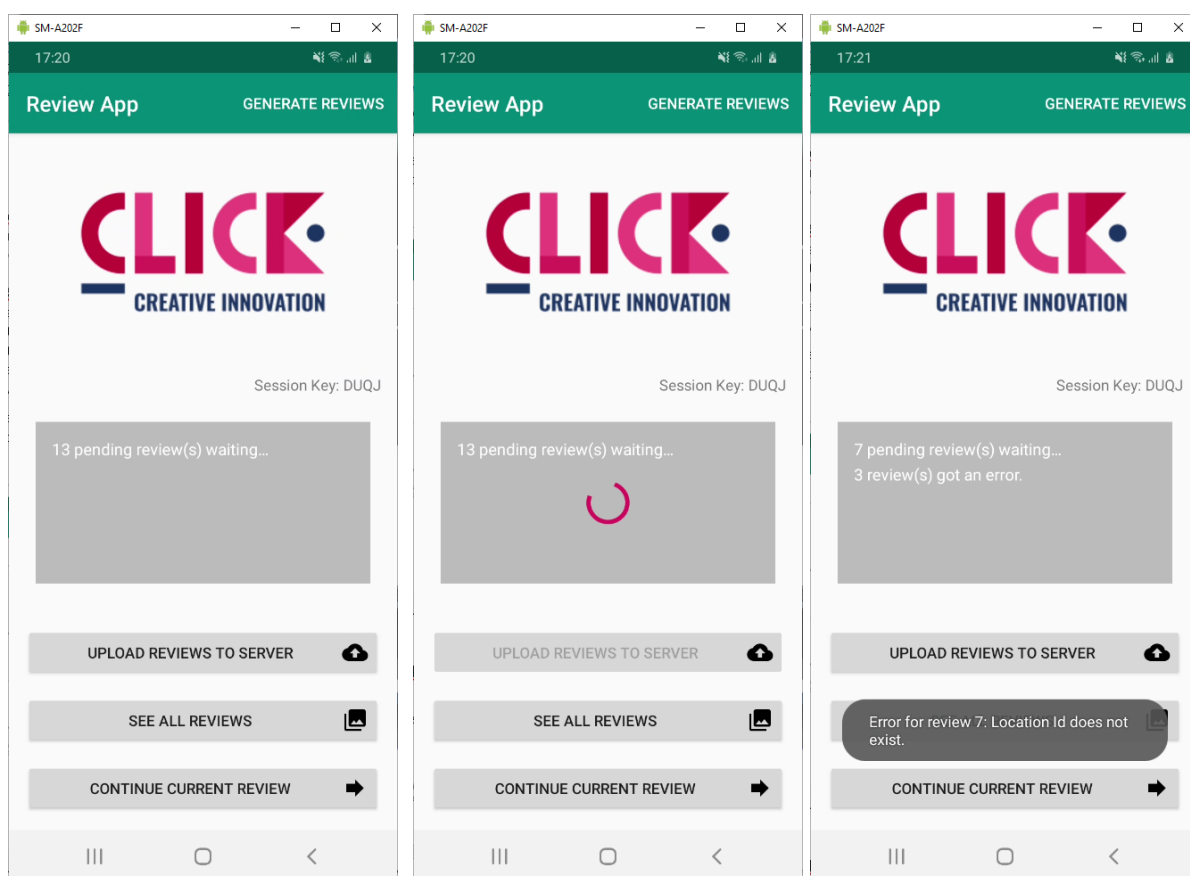


Figure 13 - Uploading reviews

8. Erreurs potentielles

L'application a été conçue afin d'être robuste face aux pertes de réseau, coupures volontaires ou involontaires de l'utilisateur et administrateur. Dans tous ces cas, la Review en cours d'upload n'est pas notée en erreur et seul un message d'information est affiché pour l'utilisateur.

A l'opposé, dans le cas où le serveur renvoie une erreur par rapport à la Review, celle-ci est notée en erreur. Elle n'est pour autant pas supprimée de l'application et celle-ci

continuera d'essayer de la renvoyer une fois que toutes les Reviews qui ne sont pas en erreur auront été envoyées.

Outre les erreurs de développement (par exemple, une mauvaise structure de données envoyée) et les erreurs de ressource sur le serveur (par exemple, plus d'espace disque disponible), l'erreur la plus susceptible d'arriver est le cas qui figure sur l'image. Si l'administrateur supprime un Floor alors que des Reviews n'ont pas encore été envoyées par les utilisateurs, celles-ci resteront bloquées sur l'application, sans pouvoir être envoyées. Il sera dès lors impossible pour l'administrateur de les récupérer sur le serveur.

Organisation du code Unity

Cette section, complémentaire du manuel utilisateur, vise à expliquer le fonctionnement général du programme serveur et à indiquer dans quelles classes se trouvent les structures de données importantes, ainsi que les fonctions pour les manipuler.

1. Interface

L'interface est constituée de 2 scènes Unity, *TitleScene* et *VisualizerScene*.

Pour chacune de ces scènes, nous présenterons l'arborescence des GameObjects.

Les noms donnés aux scripts reprennent celui du GameObject auquel ils sont attachés, suivi du suffixe *Script*. C'est également le cas pour les scripts associés à des Prefabs.

a. TitleScene

Cette scène correspond au premier écran que l'utilisateur voit lorsque le programme est lancé.

Le GameObject *ProjectCanvas* contient toute l'interface utilisateur destinée à la gestion des projets, la logique de gestion étant implémentée dans le script *ProjectCanvasScript*.

Le GameObject *NewPanel* est initialement désactivé. Il ne deviendra actif que si l'utilisateur clique sur le bouton *New* (callback dans *ProjectCanvasScript*).

Le GameObject *ErrorPanel* contient un champ texte qui peut être utilisé pour indiquer à l'utilisateur si une erreur s'est produite ou si une action interdite a été réalisée, par exemple créer un projet avec un nom déjà existant dans la liste des projets récents.

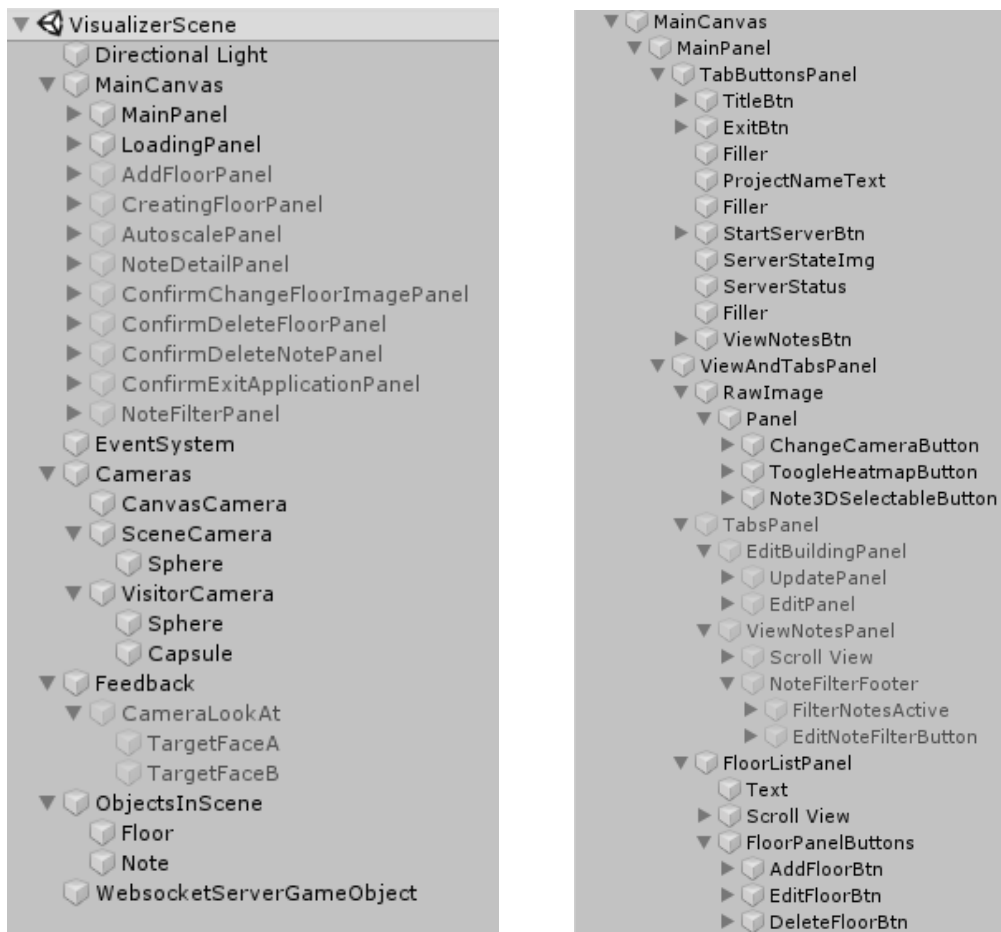
C'est dans cette scène qu'est créé le GameObject *ProjectManager* (singleton), qui n'est pas détruit lorsqu'on passe à la *VisualizerScene*. Le script *ProjectManagerScript* lui est attaché. Ce script est le plus important du programme.



b. VisualizerScene

Cette scène contient le cœur du programme, à l'exception du *ProjectManager*, créé précédemment et non détruit lors du chargement de la *VisualizerScene*. Lorsqu'on l'on travaille dans l'éditeur Unity, le programme doit être lancé obligatoirement à partir de la première scène et non à partir de la *VisualizerScene*.

Le GameObject *MainCanvas* contient l'intégralité de la GUI, répartie sur des GameObjects de type Panel.



Parmi ceux-ci, le *MainPanel* (déployé dans l'image de droite) est le panel principal. Il est divisé en sous-panels. Le *TabButtonPanel* représente la barre de menu qui se trouve en haut de la fenêtre. Le *ViewAndTabsPanel* contient :

- La vue 3D et ses boutons de contrôle (GameObject *RawImage*)
- Le *TabsPanel* qui peut se déployer ou se replier pour afficher l'éditeur de floor ou la liste des reviews (instanciation de *NoteItemPrefab*)
- Le *FloorListPanel* qui contient l'affichage des floors (instanciation de *FloorItemPrefab*).

Les autres Panels sont des surcouches de GUI venant au premier plan lorsqu'ils sont activés par l'un des scripts. Seul *LoadingPanel* est activé. C'est une nécessité car le *ProjectManagerScript* va le rechercher au chargement de la *VisualizerScene* pour y afficher les informations relatives au chargement des modèles 3D et des reviews.

Le GameObject *Cameras* contient les caméras utilisées pour calculer le rendu de l'affichage. La *CanvasCamera* correspond à ce qui est affiché à l'écran, les deux autres servent à explorer le monde 3D et sont représentées par des modèles 3D simples (sphère ou capsule).

Le GameObject *Feedback* permet d'afficher un réticule sur l'écran. Il est désactivé par défaut.

Le GameObject *ObjectsInScene* possède le conteneur dans lequel seront placés les meshes des floors à leur chargement (instanciation de *FloorPrefab*), chaque floor contenant ses GameObject 3D reviews associées (instanciation de *NotePrefab*).

Enfin, le *WebsocketServerGameObject* est un conteneur pour le script implémentant le serveur Websocket.

2. Fichiers ressources

a. Dossier Assets

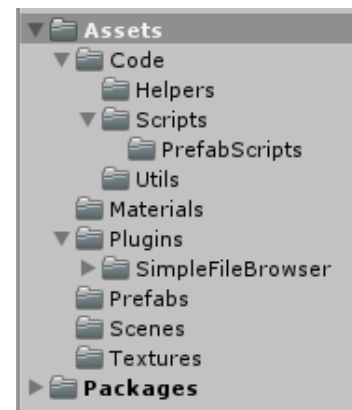
Le dossier Assets contient tous les fichiers ressources du programme.

Le dossier Code est subdivisé en trois dossiers. Le dossier *Helper* contient la classe *GuiHelper* possédant des fonctions statiques utilisées dans divers scripts et le fichier *SimulationHelper* contient la définition des structures de données manipulées par le programme (plus de détails dans la section 3.a). Le dossier *Scripts* contient tous les scripts attachés aux GameObjects, ainsi que les scripts de prefabs dans le sous-dossier *PrefabScripts*. Le dossier *Utils* contient des classes dont l'objectif est de simplifier la manipulation des données, principalement pour du traitement d'image.

Le dossier *Plugins* contient une bibliothèque DLL compilée de websocket-sharp¹. Elle est fournie ici pour garantir le fonctionnement du projet mais devrait systématiquement être remplacée par la dernière version disponible. Le dossier *SimpleFileBrowser* contient le code du plugin de même nom² utilisé pour proposer des fenêtres de navigation dans les fichiers à l'utilisateur. Il est également recommandé de la remplacer par la dernière version disponible.

Le contenu du dossier *Prefabs* est présenté dans la sous-section suivante.

Le dossier *Textures* contient toutes les images utilisées dans le programme, notamment pour la GUI, qui ne sont pas des photos associées aux reviews.

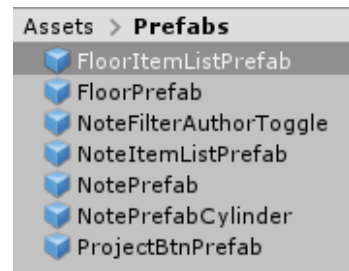


¹ <https://github.com/sta/websocket-sharp>

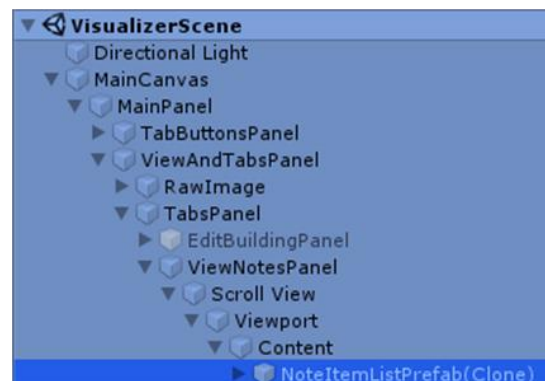
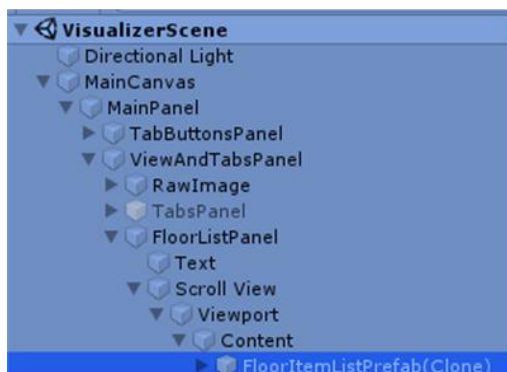
² <https://github.com/yasirkula/UnitySimpleFileBrowser>

b. Prefabs

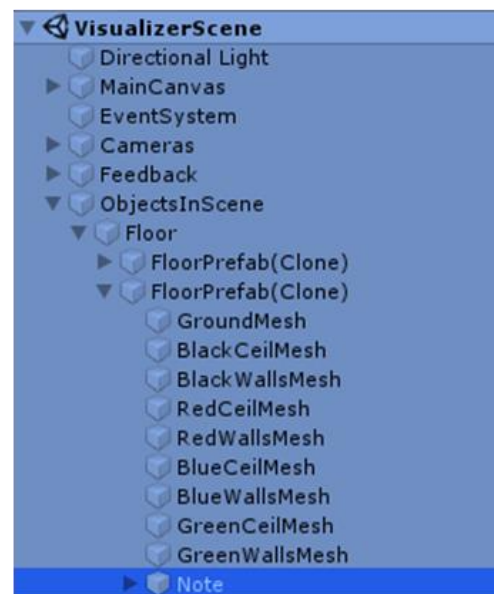
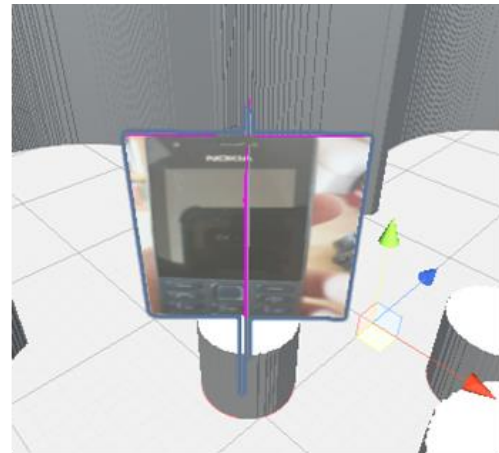
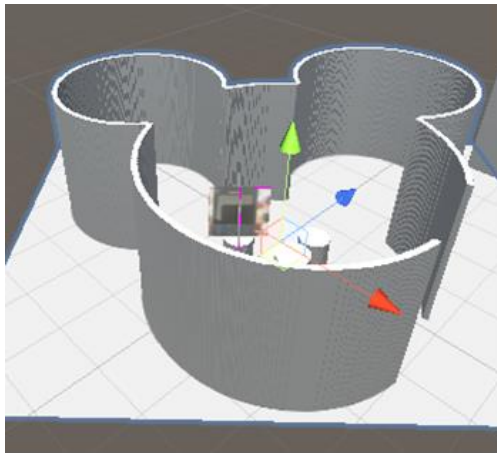
Les prefabs sont des GameObjects plus complexes que les GameObjects de base pouvant être instanciés plus facilement depuis les scripts.



- FloorItemListPrefab : GameObject 2D utilisé dans la GUI (*FloorListPanel*) pour afficher la liste des floors du projet
- FloorPrefab : GameObject 3D contenant les meshes d'un floor dans la vue 3D. Il contient également un conteneur pour les reviews
- NoteFilterAuthorToggle : GameObject 2D utilisé pour afficher une liste d'items sélectionnables (nom des auteurs de reviews) sur le *NoteFilterPanel*
- NoteItemListPrefab : GameObject 2D utilisé dans la GUI (*Scroll View* du *ViewNotesPanel*) pour afficher la liste filtrée des reviews du projet
- NotePrefab : GameObject 3D contenant quatre plans disposés en croix sur lesquels est appliquée la texture de la photo d'une review pour affichage dans la vue 3D
- NotePrefabCylinder : Similaire au précédent mais en forme de cylindre (non utilisé)
- ProjectButtonPrefab : GameObject 2D utilisé pour l'affichage des projets récemment ouverts dans la *TitleScene*



Représentation dans l'éditeur Unity de *FloorItemListPrefab* (gauche) et *NoteItemListPrefab* (droite) instanciés et hiérarchies associées



Représentation dans l'éditeur Unity de FloorPrefab (gauche) et NotePrefab (droite) instanciés et hiérarchies associées

3. Code C#

a. Structures de données

Ces structures contenues dans le fichier Helpers/SimulationHelper.cs contiennent les données nécessaires à l'affichage des images et modèles 3D dans la vue 3D. Celles déclarées Serializable peuvent être sauvegardées et chargées plus simplement vers ou depuis des fichiers XML ou JSON.

FloorStruct (Serializable)

Structure contenant les données utilisées pour extruder le plan du bâtiment en modèle 3D et le placer dans la scène 3D. Le champ Id est unique et change de valeur si l'image du floor est modifiée. Il est utilisé par l'application Android pour savoir si l'image de floor dans le cache est la dernière version disponible.

NoteStruct (Serializable)

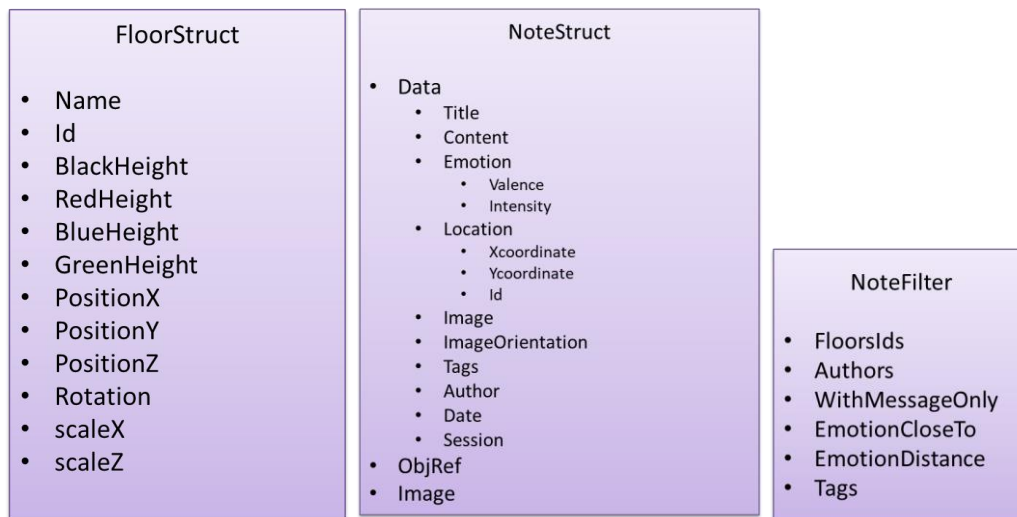
Contient les données relatives à une review. Le champ *Data* (de type *ReceivedData* – *Serializable*) représente le contenu du document JSON représentant la review envoyé par la *ReviewApp* (voir section 4). À la réception, le champ *Data/Image* contenant une chaîne de caractères base 64 représentant l'image JPEG est remplacé par le nom du fichier image sauvé sur le disque.

Le champ *ObjRef* contient une référence vers le *NotePrefab* correspondant à cette review dans la vue 3D.

Le champ *Image* est une *Texture2D* dans laquelle est chargée la version redimensionnée de la photo de la review.

NoteFilter

Structure permettant de simplifier le filtrage d'une liste de reviews.



b. Classes utilitaires

Ces classes visent à simplifier la gestion des images et la création des modèles 3D des floors à partir de leurs plans. Hormis *ImageExtruder*, aucune n'a été développée par nos soins. Les références sont données dans le Readme du répertoire *Utils* et dans chacun des fichiers C# concernés.

ImageExtruder

Classe encapsulant *MeshGenerator* pour simplifier son utilisation.

MeshGenerator

Classe créant les meshes d'un floor à partir d'une image chargée comme texture (un mesh par couleur sur la texture).

TextureFilter

Bibliothèque de fonctions de traitement d'image utilisées pour nettoyer les images avant la génération des meshes.

TextureScale

Outil de redimensionnement d'images. Utilisé pour réduire la taille des images reçues avec les reviews afin de limiter l'utilisation de mémoire par le programme.

c. Scripts principaux

ProjectManagerScript

Script de GameObject singleton disposant des fonctions pour charger, parser, trier et sauvegarder les reviews et les floors ainsi que les listes de structures contenant les données des reviews (*NotesList*) et des floors (*SimulationSetup.FloorList*). Le GameObject associé est déclaré dans la *TitleScene* et n'est pas détruit au changement de scène. Les fonctions principales sont :

- Clear : réinitialise tous les champs de la classe
- IsNoteMessage : vérifie qu'un message websocket reçu contient bien le JSON formaté comme attendu
- Update : si un message websocket contenant une review est reçu, la review est créée dans cette fonction (main thread)
- SaveNote : sauvegarde la review sur le disque dur au format JSON
- CreateNote : instancie une review de type *NoteStruct*, crée le *NotePrefab* dans la vue 3D et ajoute la review dans la liste *NotesList*
- LoadFloors : coroutine utilisée pour charger les données des floors à partir du fichier *Simulation.xml*
- LoadNotes : coroutine utilisée pour charger toutes les notes du projet
- GetNotesFiltered : fonction renvoyant la liste des reviews correspondantes à un filtre donné
- DeleteNote : supprime une review
- SaveSetup : sauvegarde tout le projet
- Create/Rename/DeleteFloorFolder : gestion du dossier de floor sur le disque dur

FloorsManagerScript

Ce script est attaché au GameObject *FloorListPanel*, panel d'affichage de la liste des floors du projet. Il est en charge de la gestion des floors et de leur représentation dans la GUI. Ses principales fonctions sont :

- Clear : détruit les meshes de tous les floors et réinitialise la classe.
- DeleteCurrentFloor : supprime le floor actuellement sélectionné
- Update : si un floor doit être créé ou modifié, ceci est assuré dans cette fonction (main thread)
- CreateFloorMesh : crée le mesh associé au floor pour la vue 3D
- CreateFloorItemList : ajoute un élément à la liste des floors dans la vue scrollable

FloorItemListPrefabScript

Bien qu'étant un script de prefab principalement dédié à la gestion des événements de la GUI, ce script est néanmoins important car ici qu'est calculée la heatmap de chaque floor lorsqu'on charge le projet ou qu'on ajoute une nouvelle review. C'est dans ce script qu'est stocké la texture associée au floor et sa représentation en chaîne de caractères base64, calculée au chargement.

WebsocketServerScript

Ce script a en charge le fonctionnement du serveur websocket. Il est attaché au GameObject *WebsocketServerGameObject*. Pour plus de détails sur les messages websocket échangés entre la ReviewApp et le serveur, voir la section 4.

CameraManipulationScript

Ce script est attaché au GameObject *RawImage* dans le *MainPanel*. Il permet de passer de la vue caméra omnisciente à la vue caméra visiteur dans la vue 3D, et gère les interactions de l'utilisateur avec cette vue pour déplacer la caméra active. Il permet aussi de mettre en surbrillance une review survolée par le curseur dans la vue 3D, si l'option *NotesSelectable* est activée.

4. Formatage des messages Websocket

Description des messages échangés entre le serveur websocket et les smartphones Android connectés sur le même réseau WiFi

Détail des termes utilisés :

- *IP* est l'adresse IPv4 de l'ordinateur sur le réseau
- *Port* est le port d'écoute Websocket paramétré dans l'éditeur Unity (7777 par défaut)
- *ProjectName* est le nom du projet ouvert dans le programme du serveur
- *SessionKey* est un code à 4 caractères choisi à la création du projet. Il est affiché dans la barre d'informations du serveur.

a. Services

Ping

Permet de vérifier si le serveur répond

URI : ws://IP:Port/SessionKey

Message texte attendu : **Aucun**

Réponse au message texte : **Aucune**

Réponse au ping : **"Pong"**

Floors

Requête pour obtenir le nom, l'identifiant et l'image de chacun des floors du projet

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

Message texte attendu : **Aucun**

Réponse au message texte : **JSON formaté comme suit**

```
{
  "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)"
    }
    ...
  ]
}
```

Réponse au ping : **identique à la réponse au message texte**

Notes

Envoi d'une note vers le serveur

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

Message texte attendu : **JSON formaté comme suit**

```
{
  "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 : Les coordonnées X et Y sont données relativement à l'image de floor. L'origine est placée en haut à gauche de l'image.

Note 2 : L'image doit être au format JPG, encodée en chaîne de caractères Base64

Réponse au message texte :

- **INCORRECT_NOTE_FORMAT** : Le message ne commence pas par le caractère { ou ne finit pas par }, ou l'un des champs suivants est manquante : "NoteMessage", "Emotion", "Location", "NoteData"
- **JSON_PARSING_ERROR** : Le serveur n'a pas réussi à décoder les données JSON
- **TEXTURE_CONVERSION_ERROR** : NoteImageData n'est pas une chaîne de caractères Base64 valide
- **SAVE_DATA_ERROR** : Le serveur n'a pas réussi à decoder ou sauvegarder l'image de la note
- **UNKNOWN_FLOOR_ID** : Le champ 'Location Id' prend une valeur qui n'existe pas dans la liste de floors du projet
- **ERROR** : Erreur non identifiée – voir les fichier logs côté serveur pour plus de détails
- **OK** : La note est créée et les fichiers associés sont sauvegardés

Réponse au ping : **Aucune**

Organisation du code Android

Dans cette section nous expliquerons l'organisation générale ainsi que les grands principes utilisés pour développer l'application. Nous passerons tout d'abord sur les différents affichages pour ensuite expliquer la structure utilisée pour organiser le code.

Les explications suivantes se font à partir de la racine de dossiers « /ReviewApp/app/src/main ».

1. Affichage

L'ensemble des images et icônes utilisées dans l'application sont rassemblées dans le dossier « res/drawable ».

Les dispositions développées pour chacun des différents écrans de l'application se trouvent dans le dossier « res/layout ». Basiquement chaque activité possède un layout correspondant. En plus de cela, le fichier « fragment_review.xml » gère l'affichage des Reviews dans la liste récapitulative.



Figure 14 - Affichage du fichier « activity_loading.xml »



Figure 15 - Affichage du fichier « activity_recap.xml »

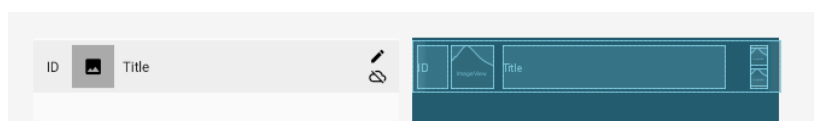


Figure 16 - Affichage du fichier « fragment_review.xml »

Comme mentionné précédemment, tous les textes affichés à l'utilisateur sont rassemblés dans le fichier « res/values/strings.xml ».

2. Code Java

Le code est présent dans « /ReviewApp/app/src/main/java/com/numediart/reviewapp ». Les classes principales sont dans le dossier en lui-même et portent un nom terminant par « Activity ». Chacune de celle-ci est en correspondance avec le layout du même nom.

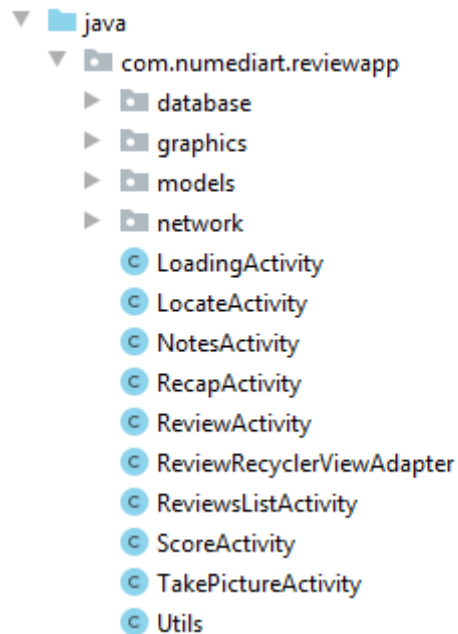


Figure 17 - Organisation du code

Une classe **Utils** reprend quelques fonctions permettant d'abstraire les requêtes réseaux et l'enchainement des interfaces du point de vue de l'utilisateur. Elle possède également une **AsyncImageTask** afin de charger les images de manière asynchrones.

Le package **database** contient les classes permettant la gestion des deux bases de données de l'application.

Le package **graphics** contient les classes étendant **ImageView** afin que celui-ci soit cliquable (**PointerImageView**) ou zoomable (**ZoomPointerImageView**, utilisée dans **LocateActivity**). La classe **ColorPointerImageView** (utilisée dans **ScoreActivity**) permet finalement de customiser le curseur en choisissant sa couleur selon sa position dans une image de référence.

Le package **models** contient, entre autres, les entités **Review** et **Floor**, utiles à n'importe quel niveau de l'application.

Enfin le package **network** rassemble les classes permettant d'effectuer les différentes requêtes vers le serveur.

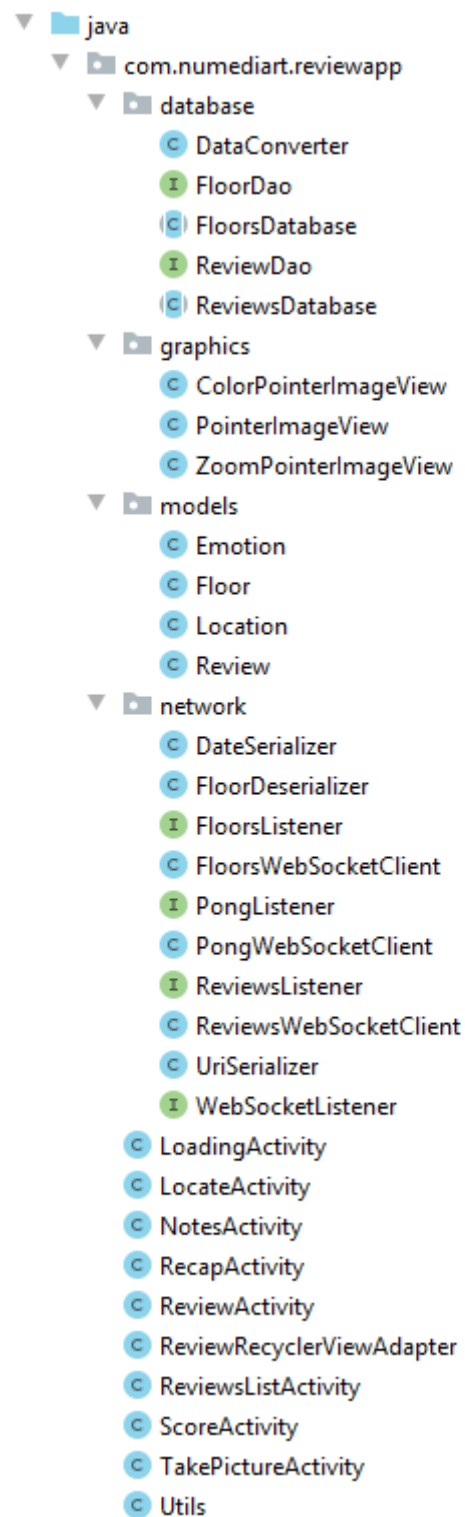


Figure 18 - Organisation du code (détaillé)

a. Interactions utilisateur

Du point de vue de l'utilisateur et selon les actions qu'il réalise, ce dernier se déplacera dans les **Activity** selon le schéma de principe suivant. A toute fin utile, la correspondance avec le layout graphique a également été indiquée.

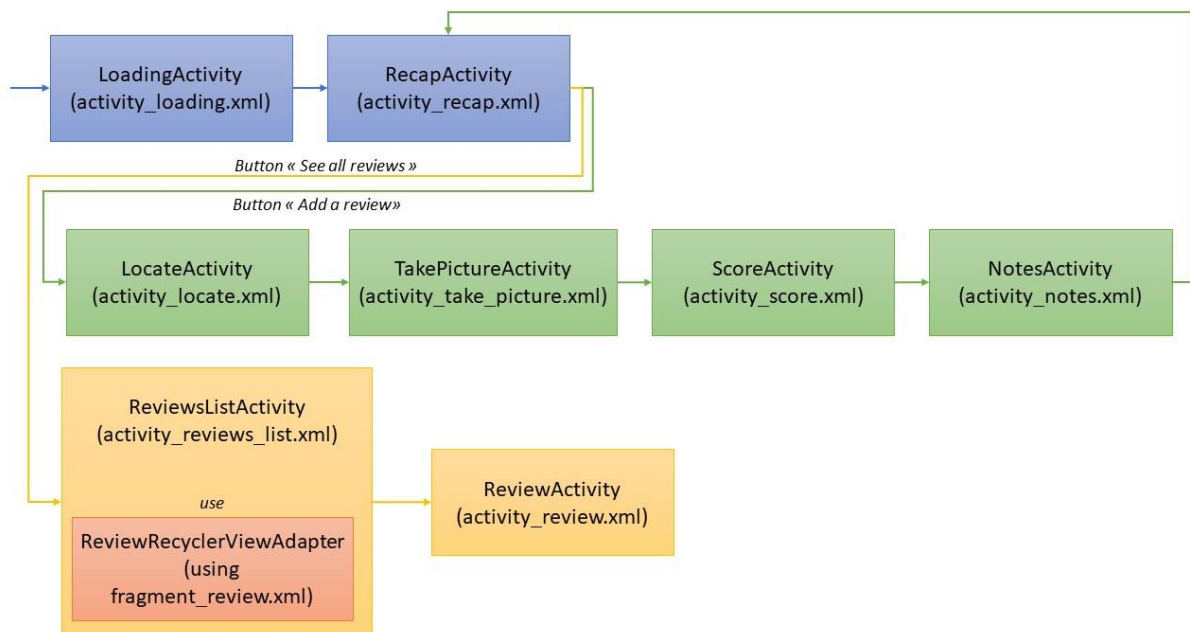


Figure 19 - Schéma utilisateur

Au démarrage, il arrivera dans la **LoadingActivity** afin d'y rentrer ses informations de connexion. Il passera ensuite dans **RecapActivity** où il pourra choisir entre « See all reviews », « Add a review » ou « Upload all reviews to server » (voir la section « Requêtes réseau » pour cette dernière action).

Dans le premier cas, il arrivera dans **ReviewsListActivity**, implémentant une RecyclerView afin d'afficher les fragments décrits dans « fragment_review.xml ». S'il clique sur l'un des fragments, il sera redirigé vers **ReviewActivity** afin d'avoir les détails de cette Review.

Dans le second cas, celui où il souhaite ajouter une Review, il passera à travers les différents écrans permettant de compléter une Review, jusqu'à être renvoyé, par l'application vers la **RecapActivity**.

b. Base de données

Il existe deux bases de données dans l'application, l'une pour stocker les **Floors**, l'autre pour stocker les **Reviews**. L'organisation en classe est sensiblement la même dans les deux cas.

Dans le package **model**, on trouve soit la classe **Floor** contenant toutes ses variables, soit la classe **Review** qui instancie deux sous-classes, **Location** et **Emotion**. Ces deux modèles, grâce à des annotations Room sont transformés en base de données et les

interfaces **FloorDao** et **ReviewDao** fournissent toutes les fonctions d'interaction possibles envers les bases de données respectives.

Enfin, afin de gérer les éventuels accès concurrents et pouvoir complexifier les requêtes faites à la base de données, les classes **FloorsDatabase** et **ReviewsDatabase** sont les deux points d'entrée pour tous les appels dans le code Java.

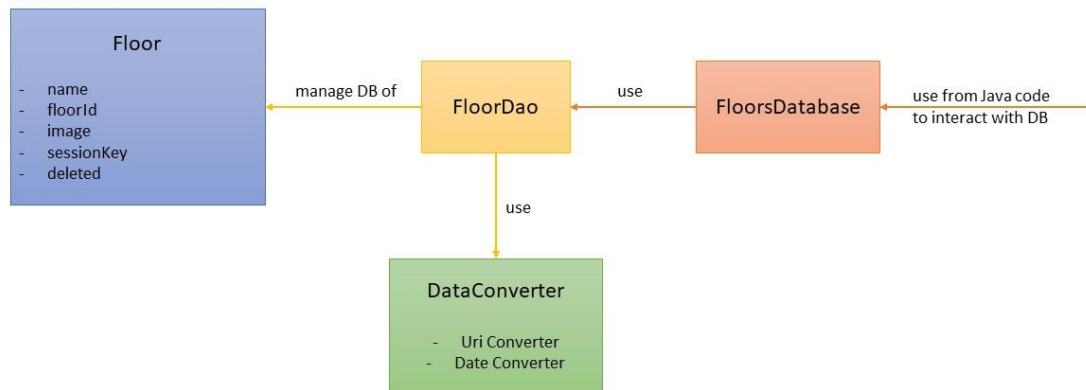


Figure 20 - Floor DB interactions

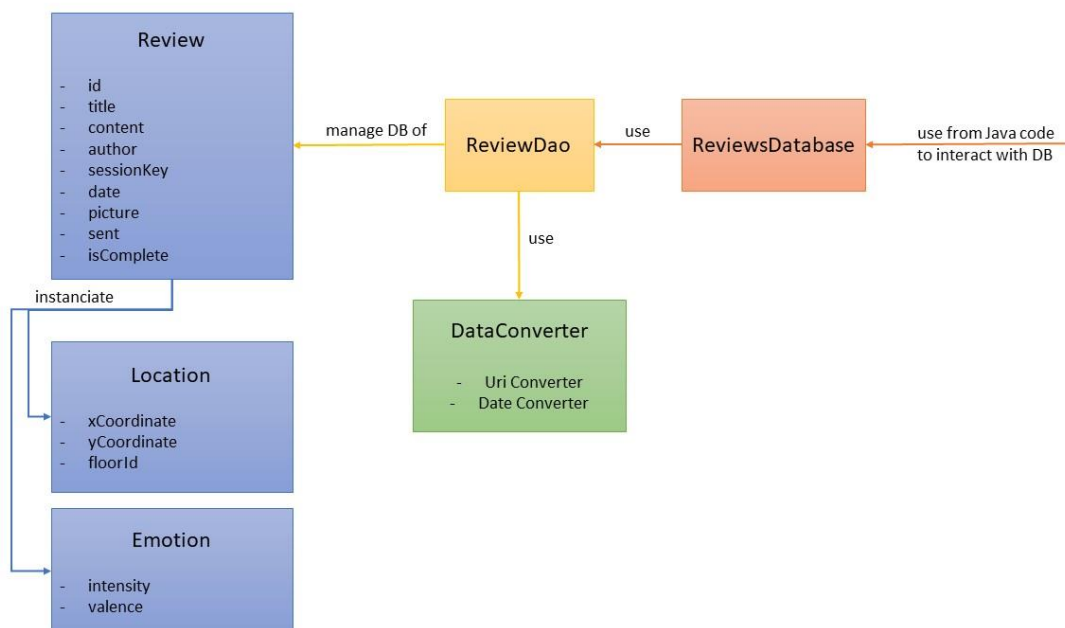


Figure 21 - Review DB interactions

c. Requêtes réseau

Trois échanges différents sont réalisés avec le serveur à différents moments dans l'application.

La première requête, la plus simple, est implémentée dans **PongWebSocketClient**. Elle permet de vérifier que l'application est capable de contacter le serveur avec les informations rentrées par l'utilisateur (adresse IP, port, clef de session). Le thread principal est averti grâce à un système de callback dont l'interface est implémentée dans l'activité.



Figure 22 - Pong server request

La seconde requête permet à l'application de recevoir les **Floors** liés à la session en cours. Grâce à la classe **FloorsWebSocketClient**, l'application reçoit un message JSON envoyé par le serveur et le decode grâce à la classe **FloorDeserializer**. Celle-ci instancie un modèle **Floor** et stocke également, dans les données de l'application, les images reçues. Il reste enfin à mettre à jour la base de données avec ces informations.

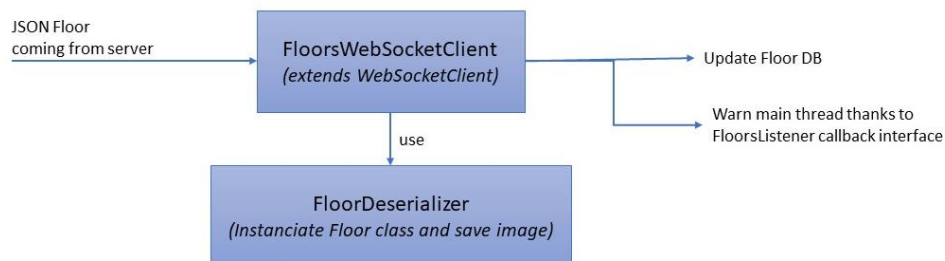


Figure 23 - Floors server request

Enfin, la troisième requête se charge d'envoyer les **Reviews** vers le serveur. Elle démarre suite à la demande de l'utilisateur d'effectuer une synchronisation. Pour y arriver, la classe **ReviewsWebSocketClient** utilise les classes **UriSerializer** et **DateSerializer**. La transformation du modèle en un message JSON est géré grâce à des annotations GSON (différentes des annotations Room pour la gestion de la base de données) dans les classes **Review**, **Location** et **Emotion**.

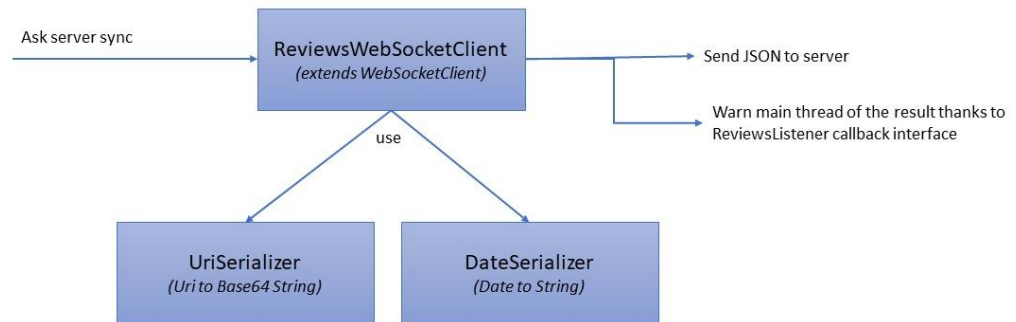


Figure 24 - Reviews server request

Ces trois requêtes sont utilisées à différents moments. Le **PongWebSocketClient** autorise l'utilisateur à entrer dans la **RecapActivity** si le serveur a pu être contacté. Dans le cas contraire et si les **Floors** existent déjà, le mode offline lui sera proposé. Une fois dans la **RecapActivity**, le **FloorsWebSocketClient** téléchargera pour la première fois (ou rafraichira) les **Floors** qui correspondent à la session. Enfin, lorsque l'utilisateur cliquera sur le bouton « Upload reviews to server », une tâche asynchrone sera démarrée, grâce à la classe interne **AsyncUploadTask** (présente dans **RecapActivity**). Celle-ci utilisera alors **ReviewsWebSocketClient** pour envoyer les **Reviews** les unes après les autres.

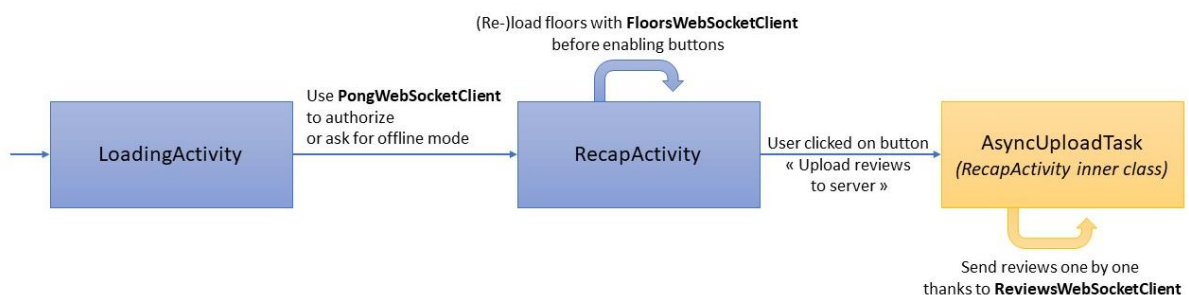


Figure 25- Requests in user interactions

3. Dépendances

Il y a trois librairies dont le code est dépendant :

- Android Debug Database
- GSON
- Android Websocket Client

La librairie Android Debug Database peut être retirée très facilement et n'est présente qu'à des fins de debug : elle permet d'accéder à la base de données de l'application à partir d'une URL.

Les deux autres librairies sont cruciales pour les échanges de message avec le serveur.

La librairie Android Websocket Client a été intégrée dans une version locale compilée (dossier « ReviewApp/WebSocketClient »). Ceci a permis d'intégrer les développements d'Alexander Shirokih alors qu'ils n'avaient pas été publiés sur JCenter. Il est possible que ces développements soient intégrés par la suite dans le développement de Gusavila92 et qu'un import Gradle de ce répertoire puisse suffire. Attention cependant, les développements d'Alexander Shirokih couvre la segmentation en fragments d'un paquet Websocket. Sans cette implémentation, il est impossible d'envoyer un paquet WebSocket complet à travers le réseau.

Mentions légales

Cette publication a été réalisée dans le cadre du projet Interreg de coopération transfrontalière C2L3PLAY, cofinancé par L'Union Européenne. 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.



<https://creativecommons.org/licenses/by/4.0/legalcode>