

UrbApp : Application Android d'acquisition de caractéristiques de façades et de sols sur site à partir d'images

Rapport de stage

Réalisé par :	Bernard Alexis
Encadré par :	Myriam Servières Vincent Tourre
Professeur encadrant :	Jean-François Hue

DUT Informatique

Année 2013-2014

Remerciements

Je remercie mes maîtres de stage, Madame Servières et Monsieur Tourre, pour leurs conseils et pour le temps qu'ils m'ont consacré, ainsi que Monsieur Petit pour l'aide apportée.

Résumé

Dans le cadre de ma deuxième année à l'IUT Informatique de Nantes, les étudiants sont amenés à effectuer un stage d'une durée de dix semaines en entreprise afin de pouvoir finaliser leur DUT Informatique.

Ainsi, suite à de nombreuses recherches, j'ai décidé de réaliser mon stage au sein de l'Ecole Centrale de Nantes, pour l'IRSTV (Institut de Recherche en Sciences et Techniques de la Ville). Ma mission consiste à reprendre un projet informatique de troisième année de cette école qui a pour but le développement d'une application Android permettant la saisie de données sur les façades et les sols à partir d'images prises sur le terrain.

Pour réaliser cela, il me fallait tout d'abord connaître la programmation sous Android, j'ai donc commencé par suivre des tutoriels sur Android. Puis j'ai étudié l'application que les élèves de l'année dernière ont développée, afin de voir les fonctions à rajouter ou à corriger.

J'ai dû également me familiariser avec le logiciel SIG (Système d'Information Géographique) de l'IRSTV, OrbisGIS, qui permet de visualiser des bases de données géographiques sous forme de d'affichages en couches 2D.

Ainsi, ce stage m'a demandé d'être autonome et de m'adapter à l'environnement de travail.

Ce travail préparatoire m'a permis d'identifier les différentes tâches à effectuer tout le long du stage : visualisation des données contenues dans la base de données distante avec OrbisGIS, correction de certaines fonctionnalités de l'application, ajout de l'import/export de la liste des matériaux contenue dans la base de données au format XML, géolocalisation d'un sol à l'aide de multipoints, sélection automatique de la couleur prise directement sur la photo, l'exportation des données pour les entrer dans le logiciel Solene-Microclimat, logiciel de tests microclimatiques.

Abstract

As part of my final year at the University of Nantes, and in order to validate a two year's degree, students were asked to complete an internship of ten weeks in a laboratory.

So, after many searches, I decided to do my internship at the Ecole Centrale of Nantes, and work for the IRSTV. My mission consists in resuming a computer science project of third year students of this school which aims at developing an Android application allowing data acquisition on facades and grounds from pictures taken on the street.

To realize it, it was necessary to me to know Android programming, I thus began to follow tutorials on Android. Then, I studied the application which the students for the last year developed, to see the functions to be added or to be corrected.

I also had to get acquainted with the software of the IRSTV, OrbisGIS, which allows showing geographic databases in 2D models.

Si, this internship asked me to be autonomous and to adapt me with the working environment.

This preliminary work allowed me to identify the various tasks to make during the internship : display of the data contents in the distant database with OrbisGIS, debug of certain features of the application, the addition of the import/export of the list of materials contained in the database in XML, the geolocalizatio of the ground by multipoint, automatic selection of the colour taken directly on the picture, the exportation of the data towards the software Solene-Microclimat, the microclimatic tests software.

Sommaire

Remerciements	2
Résumé	2
<i>Abstract</i>	3
Sommaire	4
Introduction	5
I) Environnement : L'IRSTV.....	6
1) Création.....	6
2) Leurs missions.....	6
3) Organisation.....	7
II) Le Projet : UrbApp.....	10
1) Contexte.....	10
2) L'Existant	11
2.1) Spécification	11
2.2) Architecture logicielle.....	13
2.3) Résultat.....	13
3) Outils utilisés.....	16
4) Organisation du code l'application	17
5) Organisation de la base de données.....	18
6) Les tâches réalisées.....	19
6.1) Export de la liste des matériaux	23
6.2) Correction du récapitulatif	25
6.3) La géolocalisation d'un sol.....	26
6.4) Visualisation des données et génération des différents fichiers.....	30
7) Travail restant et évolutions possibles	32
III) Apports du stage	33
IV) Bibliographie/Webographie	34
V) Annexes.....	35

Introduction

Dans le cadre du DUT Informatique, les étudiants doivent effectuer un stage d'au moins dix semaines afin de valider des compétences professionnelles à travers la réalisation d'un projet en entreprise. C'est pourquoi, de mi-avril à fin juin, j'ai effectué un stage au sein de l'Ecole Centrale de Nantes pour l'IRSTV. Au cours de ce stage, j'ai pu m'intéresser à de nouvelles technologies comme la programmation Android et les bases de données spatiales.

Ce stage a été l'opportunité pour moi de mieux appréhender un métier dans le domaine de l'informatique dans une entreprise.

Ce stage m'a permis d'acquérir des connaissances en informatique mais également d'avoir une première expérience professionnelle en informatique et a renforcé mon idée dans le choix de mes études.

Mon stage à l'Ecole Centrale a consisté essentiellement à étudier l'application déjà développée et y ajouter différentes fonctionnalités ainsi que de visualiser les données enregistrées. Cette application permet la saisie de données sur les façades et les sols à partir d'images prises sur le terrain.

Afin de pouvoir expliquer plus en détails les dix semaines passées au sein de l'Ecole Centrale, je vais tout d'abord présenter l'environnement du stage ainsi qu'une présentation de l'IRSTV, puis je présenterais le projet en lui-même, tant son contexte que ses détails sur les différentes tâches. Nous verrons ainsi ce qui a été fait, les problèmes rencontrés et les solutions apportées le cas présent. Je finirais enfin par une conclusion où je présenterais ce que ce stage m'a apporté.

I) Environnement : L'IRSTV

1) Création

L'IRSTV, ou Institut de Recherche en Sciences et Techniques de la Ville, mène des travaux de recherche sur l'imagerie urbaine et trouve son origine dans un besoin exprimé par le CNRS. Ce besoin est exprimé pour la première fois en 1993 et voit le jour en 1995 sous forme d'une fédération de recherche, FR73, qui deviendra par la suite la FR2488. Il faut attendre 2008 pour que l'IRSTV soit réellement créée. L'IRSTV compte près de 200 chercheurs et techniciens ainsi que de nombreux établissements partenaires comme le CNRS, l'Ecole Centrale ou l'Université de Nantes par exemple, c'est donc une fédération de recherche qui rassemble des laboratoires et établissements. L'IRSTV se répartie en plusieurs projets de recherche fédératifs (PRF) dans lesquels les chercheurs des laboratoires partenaires participent. L'IRSTV ne regroupe donc pas les chercheurs, mais juste les projets. De plus, tous les chercheurs des laboratoires partenaire ne travaillent pas forcément pour l'IRSTV.

2) Leurs missions

Diverses recherches sont effectuées au sein de l'IRSTV. Elles vont de l'observation de la ville à l'adaptation de celle-ci en passant par sa représentation. Ces recherches visent à développer des connaissances sur la micro-climatologie urbaine utiles à l'établissement de stratégies d'atténuation des effets du changement ou l'environnement sonore par exemple. Dans ces énergies on y trouve la thermique, l'aérodynamique, l'hydrologie ou le rayonnement par exemple. Ces énergies doivent être étudiées ensemble et en tenant compte des différentes échelles possible (bâtiment, rue, quartier, agglomération, etc).

Plusieurs recherches ont déjà été achevées comme le projet INOGEV qui visait à aider les gestionnaires de collectivités locales à réduire leurs rejets polluants. Il y a également un projet qui vient juste de finir avec comme sujet l'impact que peut avoir les toitures végétales sur la ville.

3) Organisation

L'IRSTV est un projet d'organismes qui engage les établissements partenaires sur des objectifs et des moyens. C'est un instrument de politique scientifique qui traduit la volonté de l'ensemble des partenaires de renforcer leurs compétences et leurs activités dans le domaine de la ville et qui a permis de bâtir un pôle d'excellence reconnu dans ce domaine de compétence. La direction est composée de 4 personnes puis viennent ensuite des laboratoires et établissements partenaires comme le CNRS ou bien l'Ecole Centrale de Nantes. Il existe différents projets de recherche fédératifs comme la Géo connaissances urbaines, les sols urbains ou l'environnement sonore. Mon stage se situe dans le PRF Ville et Image ainsi qu'une liaison avec celui de Micro climatologie Urbaine et Energie. En voici un organigramme :

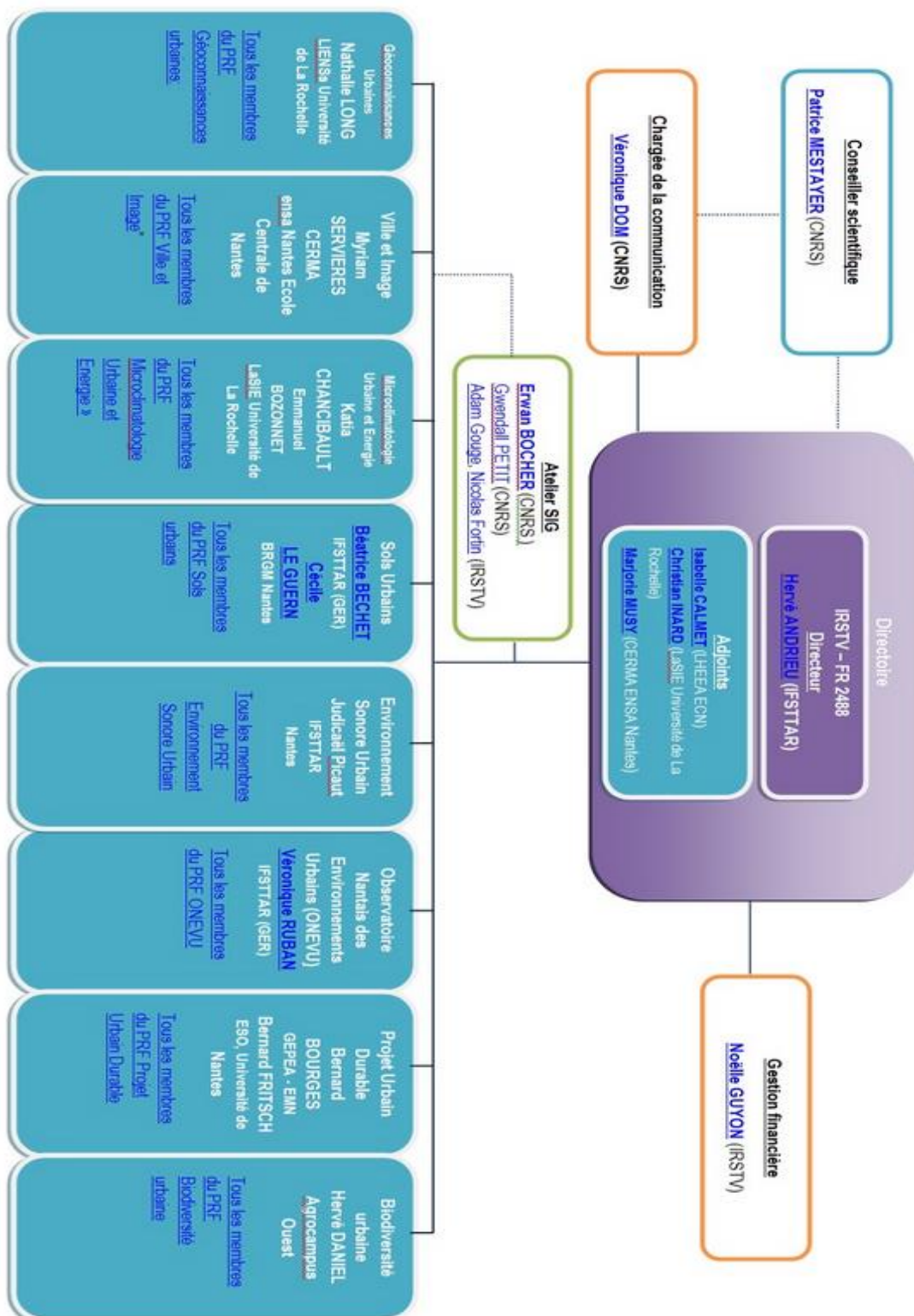


Figure 1 : Organigramme de l'IRSTV

Voici une rapide présentation de ces PRF :

- Le PRF Géo-connaissances urbaines a pour objectif de construire un espace de géo-connaissance urbaine à travers les outils de la géomantique, allant de l'acquisition des données jusqu'à la production d'analyse spatiale.
- Celui des sols urbains se focalise quant à lui sur le réaménagement d'anciens sites contaminés par des activités industrielles et artisanales pour pallier le manque de fertilité des sols urbains.
- Il y a aussi l'environnement sonore urbain qui travaille sur la prise en compte des ambiances sonores en milieu urbain.
- L'observatoire nantais des environnements urbains a pour objectif d'établir des bilans sur flux d'eaux et de polluants pour les bassins mais travaille également sur la qualité de l'air ou des sols.
- Le projet urbain durable consiste quant à lui à développer, tester, et mettre en œuvre des méthodes d'évaluation de la « durabilité » des pratiques actuelles d'urbanismes.
- On trouve également la biodiversité urbaine qui cherche à étudier l'impact qu'aurait une végétalisation sur l'environnement urbain (projet des toitures végétales)
- Mon stage se situe à l'intersection des problématiques des deux PRF restant. Il y a le microclimat urbain et énergie. Leurs recherches visent à développer des connaissances sur la micro-climatologie urbaine utiles à l'établissement de stratégies d'atténuation des effets du changement climatique et d'adapter la ville à ces changements. Cela prend en compte aussi bien les comportements thermiques que l'hydrologie ou le rayonnement et peut s'observer à différentes échelles.
- Le dernier PRF est Ville et Image. Celui-ci utilise l'image comme outil de compréhension de la ville et son objectif est de mettre en œuvre des approches automatiques d'extraction d'informations à partir d'images, mais également comme un outil de conception où l'on s'appuie sur des modèles et représentation des données urbaines récupérées.

On peut donc voir que mon stage se situe précisément dans ce dernier PRF. Je vais donc maintenant vous parler du stage en lui-même.

II) Le Projet : UrbApp

1) Contexte

Ce projet a lieu suite à l'apparition d'un besoin auprès de chercheurs dans le domaine de la thermique. En effet, pour générer des simulations microclimatiques, les chercheurs ont besoins de renseigner sur les modèles 3D utilisés tout un ensemble de caractéristiques comme par exemple le type de matériaux, leurs couleurs, le pourcentage de vitrage, la position et la longueur des balcons, etc. Actuellement, ces renseignements sont relevés sur terrain, à la main et de manière imprécise. Les informations sont écrites puis stockées en base de données pour les tests. Cependant, pour un relevé sur par exemple une rue, les différences entre les bâtiments ne sont pas prises en compte, les chercheurs vont par exemple considérer que tous les toits des bâtiments de la rue sont en ardoises, les murs en béton et la route en pavé par exemple. Or dans une même rue, chaque maison peut avoir des caractéristiques différentes (mur en béton, en pierre voire en verre/métal pour les constructions modernes).

L'idée serait alors de prendre une photo d'une façade à l'aide d'un téléphone ou d'une tablette, puis que les renseignements soit ajoutées comme des annotations sur cette image que l'on pourrait géo-localiser mais aussi les enregistrer dans une base de données spatiale afin qu'il y ait un traitement automatique, cette dernière tâche constitue une partie de mon stage.

2) L'Existant

Une application déjà fonctionnelle a été réalisée par des étudiants de troisième année de l'Ecole Centrale dans le cadre de leur projet, en voici la description faite par les élèves qui étaient sur le projet l'année dernière.

2.1) Spécification

Pour pouvoir renseigner sur site les matériaux de sols et de façades, l'application réalisée possède les fonctionnalités suivantes :

- Renseignement de la position GPS de l'appareil mobile
- Gestion de la caractérisation des éléments
- Affichage et navigation tactile
- Synchronisation des bases de données

Les données sont organisées en projet, chaque projet contenant des photos géolocalisées de la zone étudiée. Au lancement, l'utilisateur est amené à choisir ou prendre une photo sur laquelle travailler, ou peut aussi charger un projet déjà existant. Ensuite, sur la photo sélectionnée, il crée, supprime, modifie et caractérise des zones. Une fois le travail terminé, il valide ses changements et en effectue la sauvegarde. La figure ci-dessus décrit le fonctionnement général de l'application

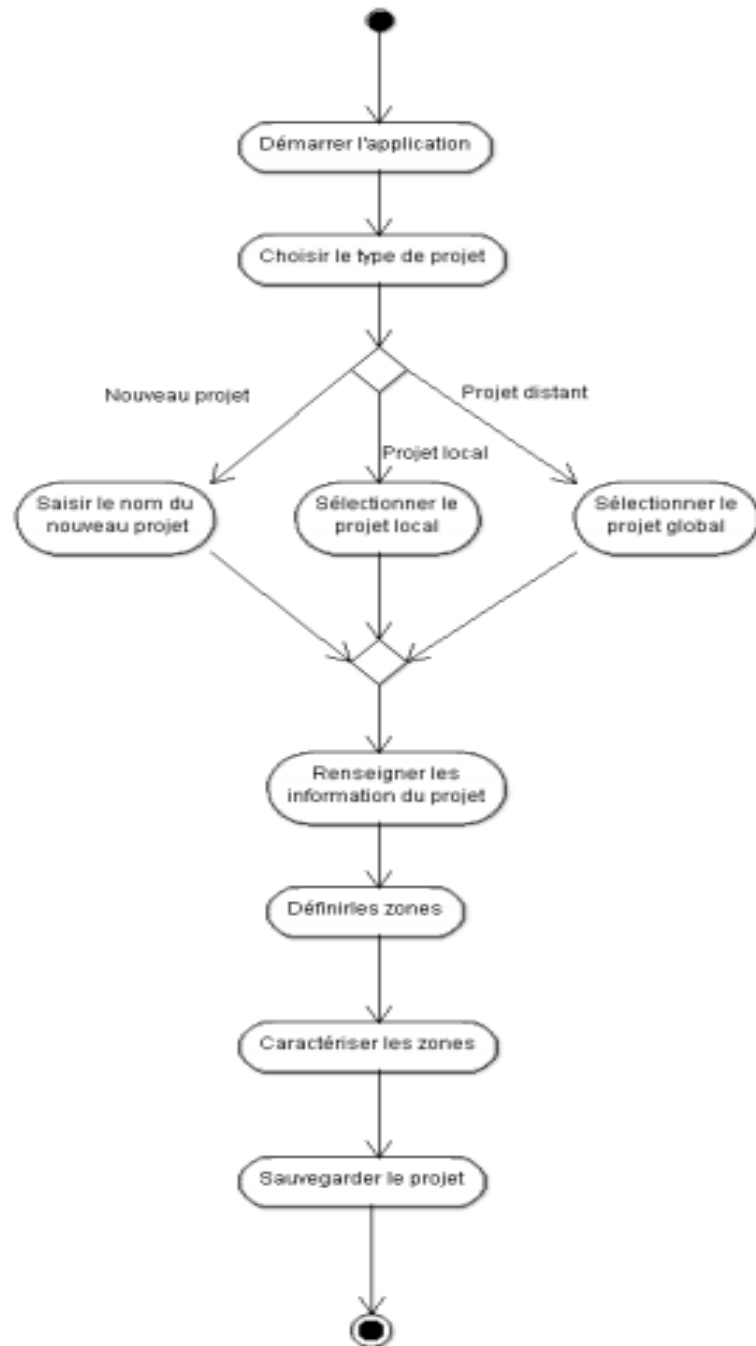


Figure 2 : Schéma d'activité de l'application

Source : Projet de l'année dernière

2.2) Architecture logicielle

L'application possède une base de données embarquée et se connecte à une base donnée spatiale distante qui permet de synchroniser et de mettre en commun toutes les saisies de projets. Il est supposé qu'en fonctionnement nominal, l'accès à un réseau est toujours possible.

2.3) Résultat

Démarrage de l'application :

A l'ouverture de l'application, le menu de démarrage s'affiche (Figure 3) et propose soit de prendre une photo via le capteur de l'appareil mobile, soit de charger une photo présente dans la mémoire de l'appareil, soit de charger un projet local (de l'appareil) ou distant (du serveur). Il est impossible de passer à un autre onglet avant d'avoir sélectionné un projet.

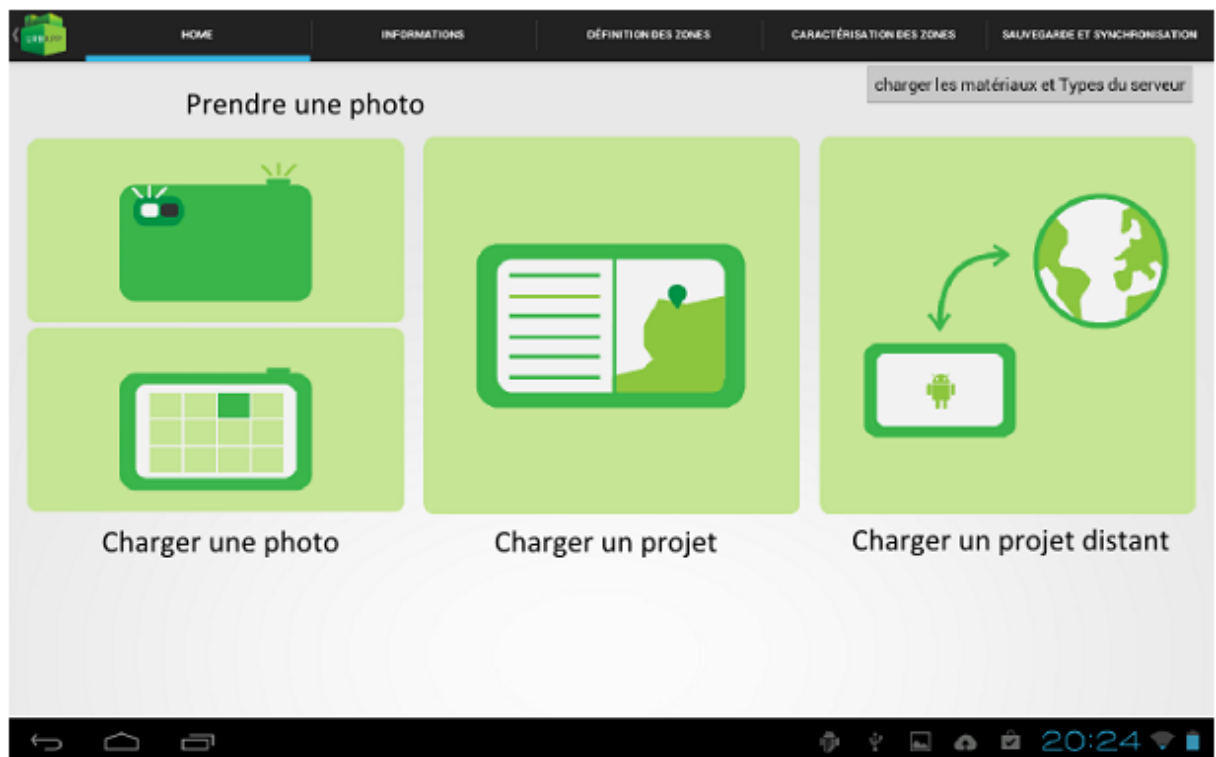


Figure 3 : Menu de démarrage de l'application UrbApp

Chargement d'un projet :

Le chargement d'un projet distant ou local donne lieu à la même vue. Sur une carte GoogleMap centrée sur la France, les projets existants sont indiqués avec le nombre de photographies qui leur sont associées.

Cliquer sur une des épingles symbolisant un projet provoque l'affichage sous forme de liste des photos qui le constitue (Figure 4).

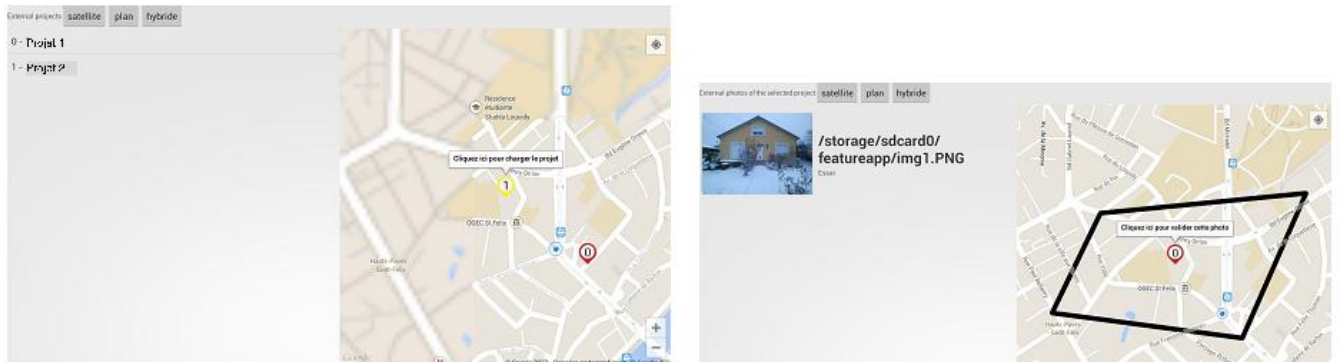
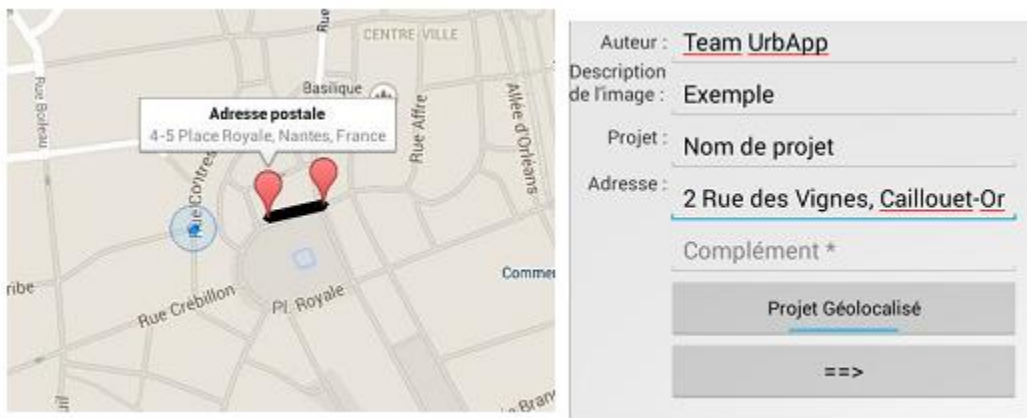


Figure 4 : Affichage des projets (à gauche) et des photos associées à un projet (à droite)

Création d'un nouveau projet :

Un nouveau projet est créé soit par la prise d'une photo, soit par le chargement d'une photo locale. Toute photo chargée doit être géolocalisée. Soit le GPS fournit une information et elle est proposée à l'utilisateur sur une vue d'une carte *GoogleMap*, soit il définit sur la même vue les coordonnées géographiques de la zone. La vue *GoogleMap* propose une visualisation satellite, plan ou hybride. A la création d'un nouveau projet, différentes informations (auteur, description, nom, localisation) doivent être renseignées (Figure 5). Une nouvelle photo peut aussi être ajoutée à un projet existant.



F

Figure 5 : Localisation d'une façade (à gauche) et définition des informations d'un projet (à droite)

Ajout d'informations sur une photo :

Une fois la zone de travail géolocalisée, toutes les photos appartenant au projet courant vont pouvoir être traitées. Manuellement, l'utilisateur va sélectionner les zones qu'il souhaite caractériser et lui affecter des caractéristiques (type, matériaux et couleur) (Figure 6).



Figure 6 : Sélection d'une fenêtre (à gauche) et affectation des caractéristiques de la zone (à droite)

Une fois la zone (ici la façade) entièrement caractérisée (Figure 7), l'utilisateur peut sauvegarder à la fois dans la base locale et dans la base distante ses caractéristiques.



Figure 7 : Façade entièrement caractérisée

Dans mon stage, mon travail s'agissait donc de corriger certaines de ces fonctionnalités, en améliorer d'autres, et surtout faire la transition des données vers le logiciel de simulations microclimatiques.

3) Outils utilisés

Du fait que les photos doivent être acquises obligatoirement sur site, l'idée de faire une application Android a été retenue comme étant la plus pertinente de par sa mobilité, de plus, la majorité des smartphones et des tablettes de nos jours possèdent une puce GPS, ce qui facilite la géolocalisation. L'ensemble du programme sera donc développé en Java sous environnement Android 4.3 (version la plus à jour lors de la création du projet). L'interface de développement utilisée ici est Eclipse qui est un environnement de développement spécialisé dans le langage Java. Pour pouvoir développer des applications Android, il faut également installer le plugin ADT (*Android Development Tools*) mais aussi un kit de développement (*SDK Android*) mis à disposition par Google afin de développer une application Android.

Il y a deux manières de tester son programme, soit en utilisant un terminal Android (smartphone ou tablette) soit en utilisant l'AVD (*Android Virtual Device*) qui est une machine virtuelle Android. Durant mon stage, une tablette m'a été mise à disposition pour effectuer mes tests.

Les données que l'on stocke étant géo-localisées et sous forme de polygones (on dessine une zone à caractériser, une ligne pour géo-localiser une façade, etc) il faut une base de données spatiale ainsi qu'un logiciel pouvant représenter ces données. Pour la base de données, le choix de PostgreSQL a été fait. PostgreSQL est un système de gestion de base de données relationnelle et objet. C'est également un outil disponible librement.

L'avantage qu'à PostgreSQL est qu'il peut stocker plus de type de données que les types standard (entier, caractère, ...). L'utilisateur peut créer ses propres types de données ou fonctions. Pour visualiser ces données, un logiciel de l'IRSTV a été utilisé, OrbisGIS. OrbisGIS a été développé pour répondre à des besoins scientifiques. Cette initiative s'inscrit dans le cadre du Projet de Recherche Fédératif « Ville et Image » dont l'un des axes principal est la création d'outils et de méthodes pour l'analyse spatiale en milieu urbain. Il permet de représenter un modèle 2D des données traités ainsi que de les superposer entre eux.

Une autre base de données a été utilisée. Il fallait pouvoir sauvegarder les informations localement au cas où le serveur serait par exemple indisponible. SQLite a donc été utilisé car il permet d'être intégré directement au programme et donc de sauvegarder les données localement.

Un dépôt GitHub sera également utilisé à des fins de partage de l'application. GitHub est un service web d'hébergement et de gestion de développement de logiciels, utilisant le logiciel de gestion de versions Git.

4) Organisation du code l'application

Le code source est réparti en plusieurs packages définit comme suit :

- *Activities* : contient toutes les activités de l'application
- *DB* : contient toutes les classes décrivant la structure de la base de données
- *Dialogs* : contient toutes les boites de dialogues de l'application
- *Fragments* : contient tous les fragments (morceaux d'interfaces)
- *SyncToExt* : contient toutes les classes liées à la synchronisation avec la base de données distante
- *Utils* : contient des outils divers (notamment le nuancé de couleurs pour la sélection de la couleur)
- *Zones* : contient toutes les classes liées à la gestion des zones que l'on caractérise

Les trois derniers packages sont ceux dont je n'ai apporté aucunes modifications dans le code source.

Le diagramme de classe complet est quant à lui disponible en annexe

5) Organisation de la base de données

La base de données est composée de sept tables.

Il y a d'abord des tables comme Material et ElementType qui référencent respectivement la liste des matériaux et des types d'élément (façade, toiture, sol).

Il y a aussi deux tables qui correspondent aux valeurs géométriques : GpsGeom qui contient la géolocalisation d'une photo, et PixelGeom qui contient les polygones dessinés lors de l'étape de définition des zones.

Une table Photo qui contient une description de la photo, un auteur, la date de dernière modification et son adresse. Elle utilise aussi l'identifiant d'un GpsGeom pour la localisation.

La table Project contient quant à elle le nom du projet et utilise elle aussi un GpsGeom.

La table centrale de cette base est la table Element. Elle contient les identifiants (utilisé comme clef primaire dans chaque table) de tout ce qui compose une zone que l'on a caractérisée : ses matériaux, son type d'élément, sa localisation, son polygone et sa photo.

La dernière table, Composed, fait le lien entre une photo et un projet.

Voici un schéma de cette base de données :

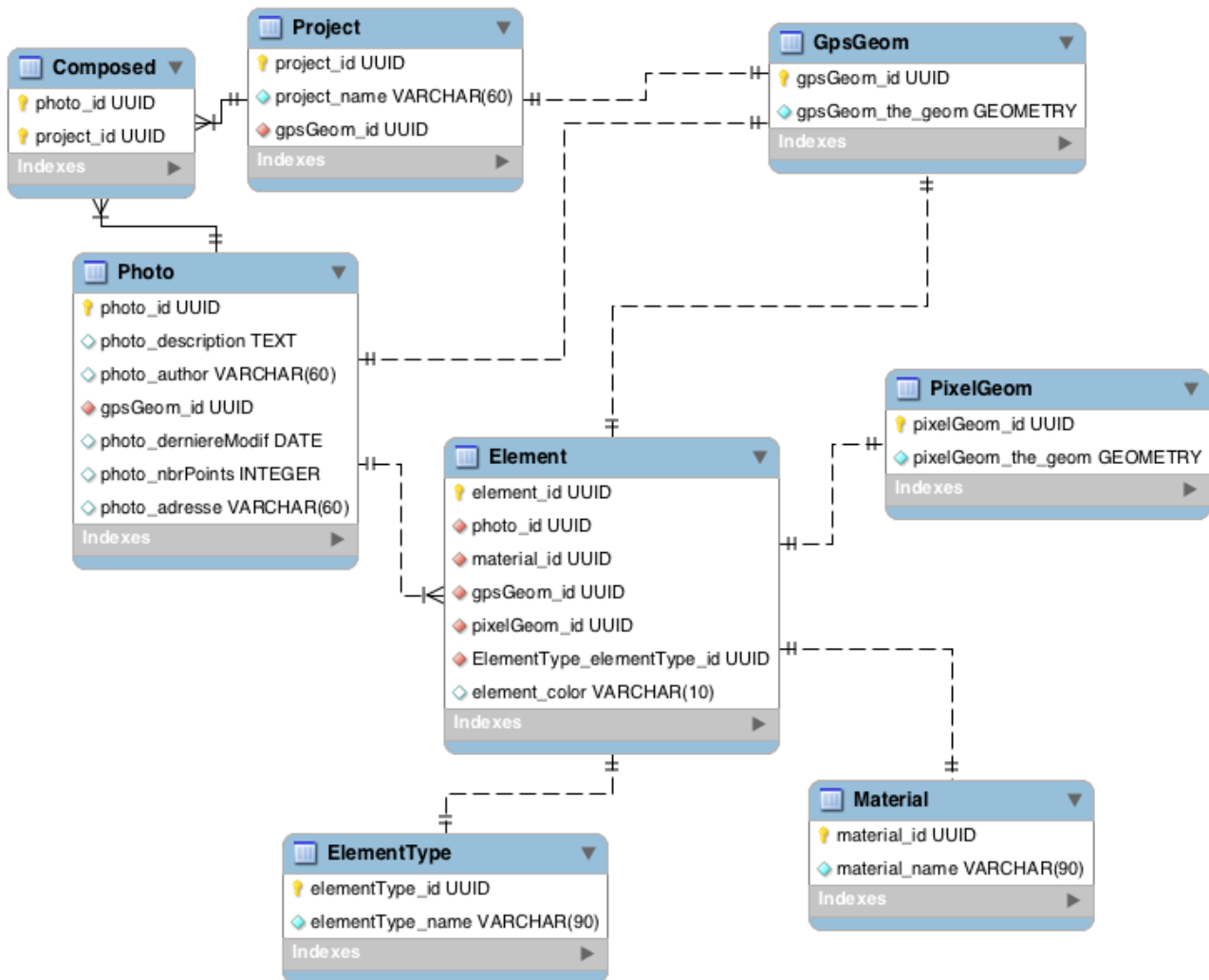


Figure 8 : Schéma de la base de données

Source : Projet de l'année dernière

6) Les tâches réalisées

Comme je l'ai dit, mon stage consiste à modifier et compléter le travail déjà effectué. Puis ensuite de faire transiter ces données jusqu'à Solene-Microclimat. Pour ce faire, j'utilise le logiciel OrbisGIS, un logiciel de l'IRSTV, qui me permet de visualiser les données sous forme de couches 2D, mais aussi d'exporter ces données sous certains formats de données. Ce logiciel se présente comme suit :

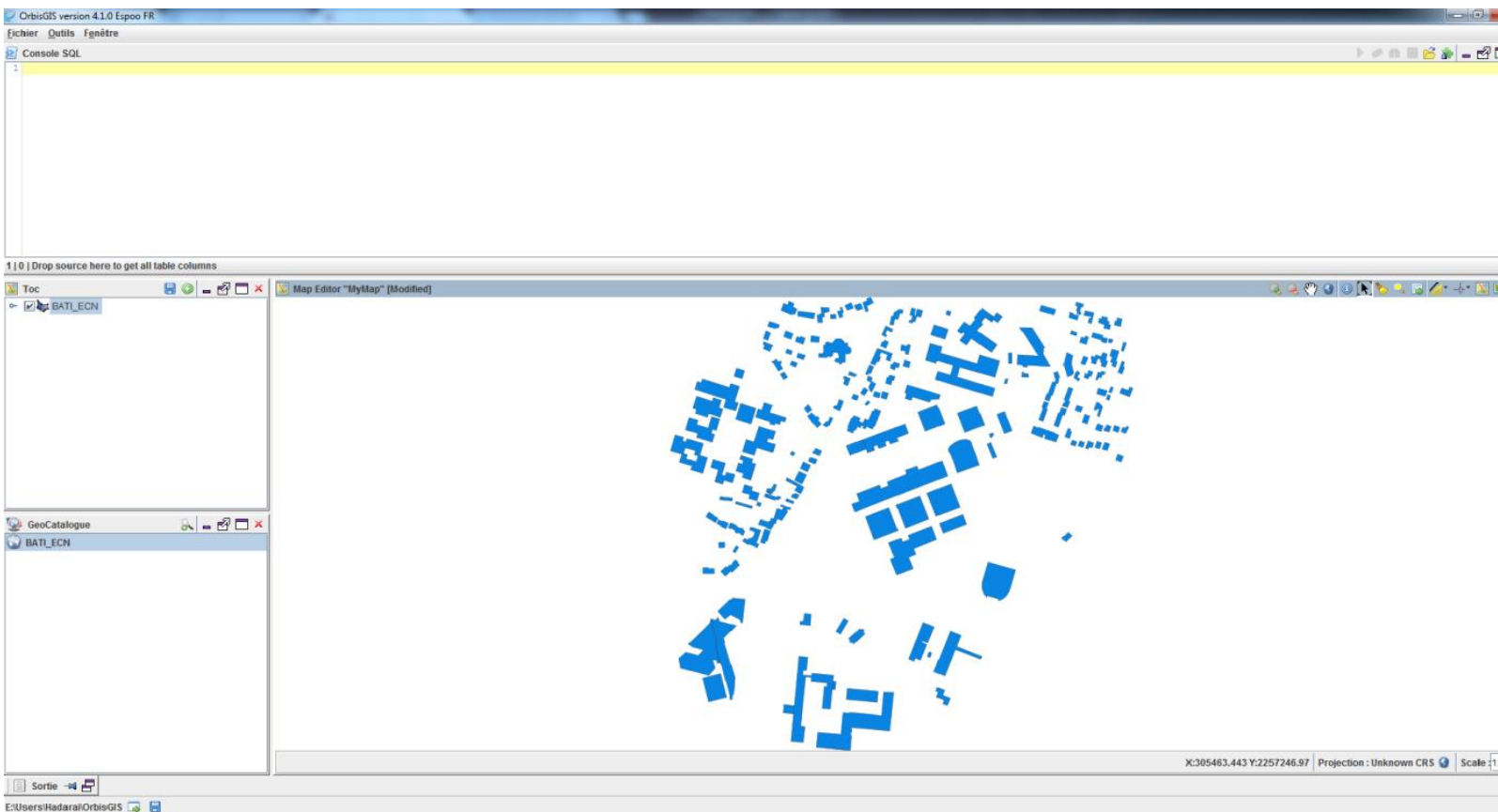


Figure 9 : Fenêtre principale d'OrbiGIS

Dans le cadre GéoCatalogue (en bas à gauche), on ajoute les données que l'on veut représenter ou dont on a besoin. On peut y ajouter des shapefiles ou des bases de données spatiales. Les shapefile (ou .shp) sont des fichiers de formes et contiennent des informations liées à la géométrie des objets décrits (points, lignes ou polygones). Pour créer sa carte, il suffit de faire un glisser-déposer des fichiers du GéoCatalogue vers le cadre Toc (au milieu à gauche) et la carte se dessine. Ici, on peut voir la carte 2D de l'Ecole Centrale de Nantes et ses alentours. Il y a également une console SQL en haut qui permet d'exécuter des instructions SQL sur nos bases de données et fichiers. Sur la carte, plusieurs options sont possible : on peut zoomer/dezoomer, faire une sélection d'une zone, faire des opérations de mesures sur une surface, un angle ou un segment (figure 10), ou alors afficher les attributs d'une sélection (figure 11).

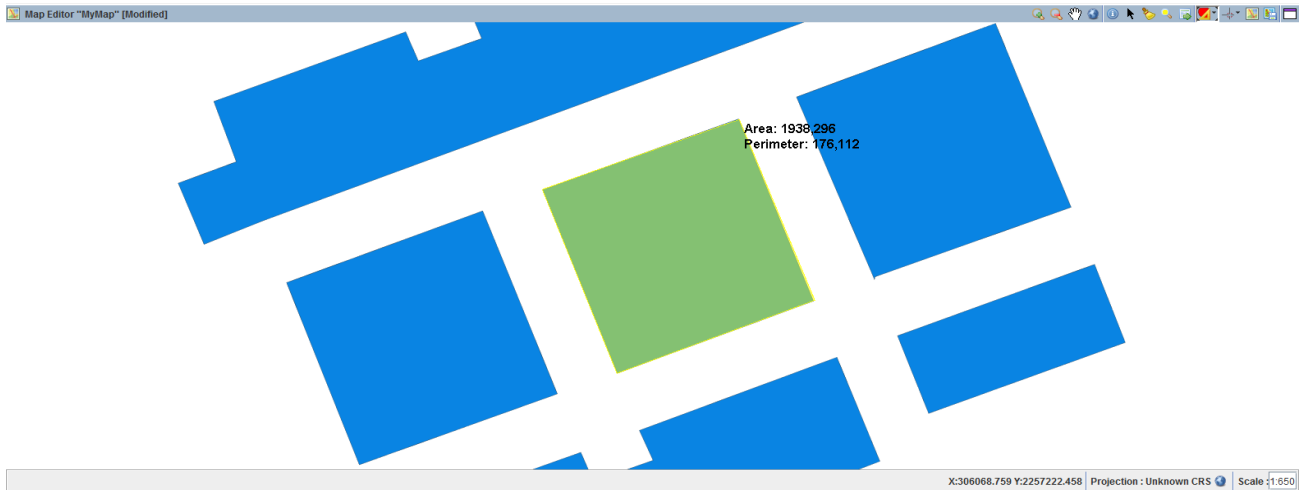


Figure 10 : mesure d'une surface

BATI_ECN - 0/163								
	THE_GEOM	ID	PREC_PLANI	PREC_ALTI	ORIGIN_BAT	HAUTEUR	Z_MIN	Z_MAX
1	MULTIPOLYGON (((306395.596933...	BATIMENT0000000029063752	1,5	1	Autre	4	28,5	28,6
2	MULTIPOLYGON (((306392.319497...	BATIMENT0000000029068906	1,5	1	Autre	7	34,1	35,5
3	MULTIPOLYGON (((306194.072606...	BATIMENT0000000029068904	1,5	1	Autre	20	22,6	36,3
4	MULTIPOLYGON (((306133.124785...	BATIMENT0000000029068903	1,5	1	Autre	20	33,6	45,7
5	MULTIPOLYGON (((306190.165363...	BATIMENT0000000029068911	1,5	1	Autre	5	31,1	31,1
6	MULTIPOLYGON (((306368.254901...	BATIMENT0000000029068909	1,5	1	Autre	7	33,3	36,1
7	MULTIPOLYGON (((306019.266946...	BATIMENT0000000029068928	1,5	1	Autre	4	26,1	26,5
8	MULTIPOLYGON (((305852.022428...	BATIMENT0000000029068941	1,5	1	Autre	10	20,8	24,9
9	MULTIPOLYGON (((306032.970076...	BATIMENT0000000029068937	1,5	1	Autre	6	23,5	25,1
10	MULTIPOLYGON (((305845.995731...	BATIMENT0000000029069413	1,5	1	Autre	4	21,4	21,4
11	MULTIPOLYGON (((305722.689453...	BATIMENT0000000029069412	1,5	1	Autre	4	32,1	33,3
12	MULTIPOLYGON (((306053.616698...	BATIMENT0000000029069410	1,5	1	Autre	16	32,1	40,3
13	MULTIPOLYGON (((306388.973177...	BATIMENT0000000029069404	1,5	1	Autre	6	34	35,1
14	MULTIPOLYGON (((306332.989677...	BATIMENT0000000029069403	1,5	1	Autre	7	31,7	33,4
15	MULTIPOLYGON (((306023.410091...	BATIMENT0000000029069639	1,5	1	Autre	5	20,6	21,8
16	MULTIPOLYGON (((306377.497264...	BATIMENT0000000029069868	1,5	1	Autre	5	31,5	31,7
17	MULTIPOLYGON (((306214.398347...	BATIMENT0000000029069876	1,5	1	Autre	3	22,5	22,9
18	MULTIPOLYGON (((305849.857251...	BATIMENT0000000029069970	1,5	1	Autre	3	28,1	28,2
19	MULTIPOLYGON (((305748.567104...	BATIMENT0000000029070006	1,5	1	Autre	4	29,4	30,6
20	MULTIPOLYGON (((306074.238184...	BATIMENT0000000029070004	1,5	1	Autre	12	17,1	23,1
21	MULTIPOLYGON (((305867.683419...	BATIMENT0000000029070021	1,5	1	Autre	4	21,6	21,7
22	MULTIPOLYGON (((305886.701691...	BATIMENT0000000029070115	1,5	1	Autre	7	31,5	31,9
23	MULTIPOLYGON (((305849.348236...	BATIMENT0000000029070164	1,5	1	Autre	4	28	28,3
24	MULTIPOLYGON (((305998.239081...	BATIMENT0000000029070300	1,5	1	Autre	6	27,1	27,7
25	MULTIPOLYGON (((306081.633684...	BATIMENT0000000029070546	1,5	1	Autre	3	17	17,1
26	MULTIPOLYGON (((306011.169693...	BATIMENT0000000029070708	1,5	1	Autre	7	29	29,3
27	MULTIPOLYGON (((305786.217264...	BATIMENT0000000029070705	1,5	1	Autre	4	29,8	29,9
28	MULTIPOLYGON (((305809.085504...	BATIMENT0000000029068932	2,5	2,5	BDTopo	3	19,2	20,3
29	MULTIPOLYGON (((305911.268002...	BATIMENT0000000029068945	2,5	2,5	BDTopo	5	17,2	19,5
30	MULTIPOLYGON (((306406.248570...	BATIMENT0000000029062467	2,5	2,5	BDTopo	4	26,3	26,4
31	MULTIPOLYGON (((306406.518790...	BATIMENT0000000029062466	2,5	2,5	BDTopo	5	32,6	33,7
32	MULTIPOLYGON (((306430.401021...	BATIMENT0000000029063027	2,5	2,5	BDTopo	4	20,5	21,6
33	MULTIPOLYGON (((306394.466826...	BATIMENT0000000029063025	2,5	2,5	BDTopo	5	30,4	30,4

Figure 11 : affichage des attributs sous forme de tableau

Solene-Microclimat, le logiciel de simulations climatiques, prend en entrée plusieurs fichiers. Tout d'abord des fichiers XML qui référencent les caractéristiques d'un matériau donné (comme sa conductivité, sa capacité thermique ou sa masse volumique), mais également des fichiers .val, .cir, et .shp. Les fichiers VAL sont des fichiers de données attributaires, ils affectent une valeur à certaines variables (on peut dire, par exemple, que le matériau « béton » aura la valeur 3), les fichiers CIR quant à eux servent à l'exécution des simulations climatiques. Voici, ci-dessous, un schéma représentant la chaîne de transition des données, de l'application UrbApp jusqu'aux simulations de Solene-Microclimat :

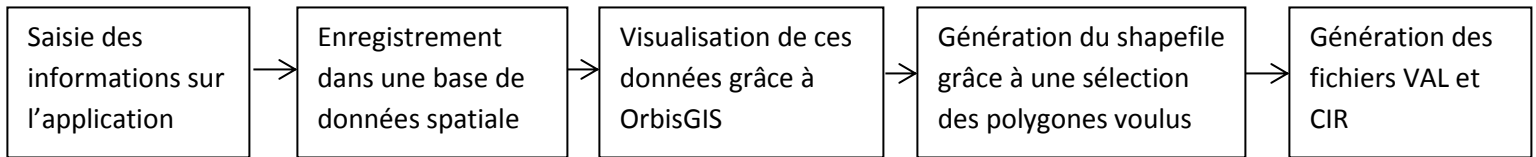


Figure 12 : Chaîne de transition des données de l'application jusqu'à Solene-Microclimat

L'objectif final du projet UrbApp est que la partie de visualisation des données ne soit pas obligatoire que la génération des différents fichiers soit automatique.

Avant de pouvoir commencer à reprendre ce projet, il m'a fallu d'abord apprendre à développer sous Android. J'ai donc suivi des tutoriels en ligne afin d'en apprendre plus sur ce type de programmation. Après avoir fait cela, il me fallait faire toutes les étapes qu'un management de projet demande, à savoir l'établissement d'un cahier des charges (disponible en annexe), d'un planning ainsi que la définition de priorité quant aux tâches à effectuer. Voici le planning qui en a résulté :

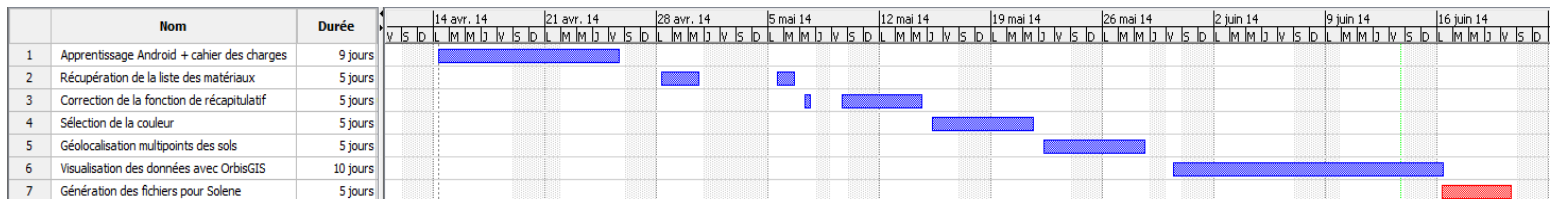


Figure 13 : Planning des tâches

La priorité a été mise tout d'abord sur les correctifs et améliorations de l'application. Cependant, après avoir passé plusieurs jours sur certains problèmes que je détaillerai après, l'amélioration de la sélection de la couleur, qui avait pour but de pouvoir caractériser la couleur d'une façade automatiquement grâce à une lecture des pixels de l'image, a été laissée en retrait afin de passer plus de temps sur OrbisGIS.

6.1) Export de la liste des matériaux

Définition du besoin :

Une des premières tâches demandées fût la récupération de la liste des matériaux de la base de données. Avant de faire cela, il me fallait tout d'abord savoir quel format serait le plus adapté. J'ai donc eu une réunion avec un des membres de l'IRSTV, Benjamin Morille. Les données liées aux matériaux qui sont envoyées au logiciel Solene-Microclimat sont des fichiers XML sous la forme suivante :

```
<matériaux>

  <matériau>

    <nom> béton </nom>

    <conductivite> 1.75 </conductivite>

    <capacite_thermique> 1000 </capacite_thermique>

    <masse_volumique> 2300 </masse_volumique>

  </matériau>

  ...

</matériaux>
```

Solution :

Afin de créer ce fichier XML en Java, j'ai utilisé la classe `DocumentBuilder` qui utilise l'API DOM. DOM est un parseur XML, il permet de lire un document XML et d'en extraire des informations. Seulement, ici on ne veut pas lire un fichier mais en créer un, il faut donc utiliser `DocumentBuilderFactory`. Grâce à cela, on construit l'arbre XML nœud par nœud, avec évidemment la possibilité de définir des attributs. Pour ajouter cela, j'ai placé le code de création du fichier XML dans la classe `HomeFragment`, qui correspond à l'affichage de la page de démarrage de l'application. J'ai dû également ajouter un bouton pour permette son exécution.

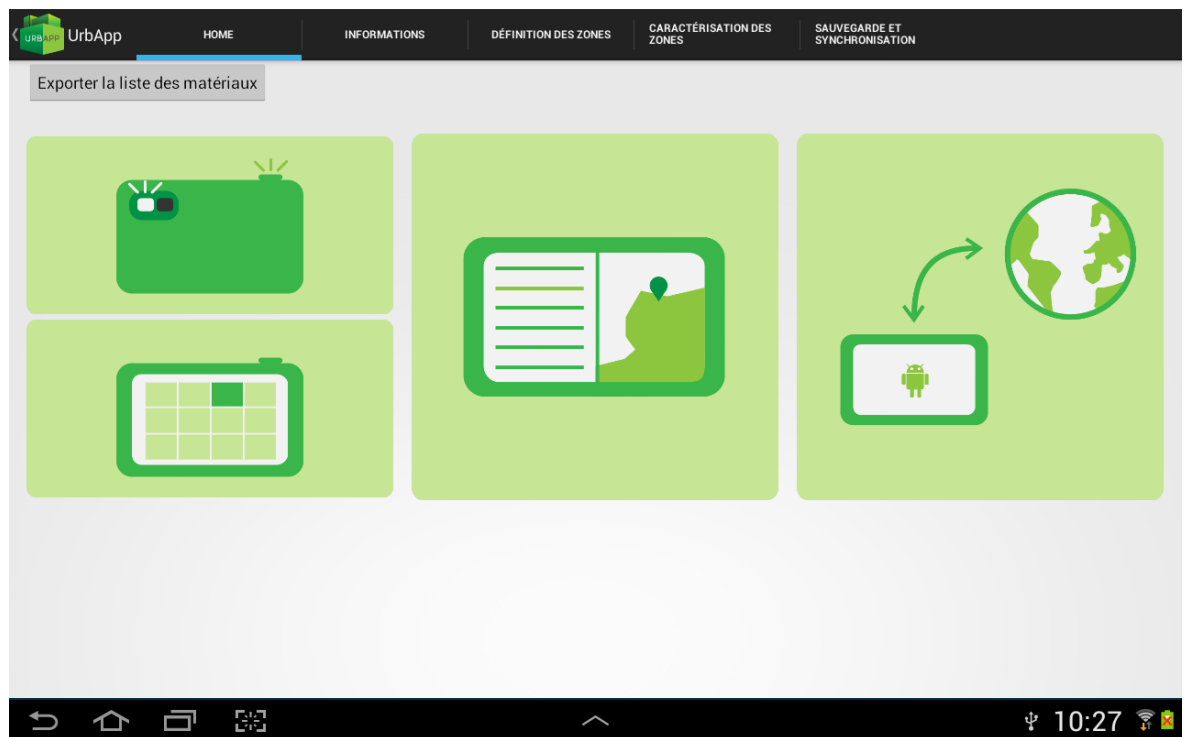


Figure 14 : Fenêtre de démarrage de l'application avec le nouveau bouton

Problème rencontré :

Lors de ma réunion avec Monsieur Morille, ce dernier m'a envoyé des fichiers XML d'exemple utilisé par Solene-Microclimat. Dans la base de données distante de départ, il n'y avait que la colonne « matériel » mais pas ses caractéristiques, je les ai donc ajoutées. Seulement, après avoir modifié les classes en conséquence, l'application ne fonctionnait plus. La raison de cela serait due aux curseurs utilisés pour parcourir la base de données, mais même sachant cela je n'ai pas réussi à ce que ces colonnes soient prises en compte. Le résultat obtenu est donc le suivant :

```
<matériau id="1">
  <nom>Acier</nom>
  <conductivite>0.0</conductivite>
  <capacite_thermique>0</capacite_thermique>
  <masse_volumique>0</masse_volumique>
</matériau>
```

Figure 15 : Aperçu du fichier XML créé

6.2) Correction du récapitulatif

Définition du problème :

Sur l'application que j'ai récupérée, une fonction de récapitulatif était disponible. Celle-ci permettait de savoir si des zones n'étaient pas entièrement caractérisées. Cependant, cette fonction ne fonctionnait pas, elle renvoyait soit toutes les zones alors que celles-ci étaient caractérisées, soit plusieurs fois la même zone.

Solution :

L'erreur de cela était due aux conditions posées dans la boucle `if`. Les anciens étudiants vérifiaient si, pour une zone donnée, le matériau ou le type d'élément (façade, sol, toit) était renseigné, et donc si l'identifiant du matériau (ou du type d'élément) était égal à 0. Cela posait problème car, si le matériau n'est pas renseigné, l'identifiant du matériau de la zone a une valeur inconnue, qui peut très bien être égale à 0 ou non. Pour remédier à cela, il faut vérifier directement s'il y a une valeur dans le champ `material_id`, et donc si celui-ci est égal à `null` ou non. Or, la variable `material_id` est un entier, et un entier ne peut pas être égal à `null`.

J'ai donc utilisé une méthode la classe `String`, `String.valueOf()`, qui permet de convertir un entier en une chaîne de caractère et ainsi savoir s'il était égal à `null` ou non. La classe qui est concerné par cette modification est la classe `SummaryDialogFragment` du package `dialogs`.

Si l'on clique sur un des boutons, l'élément désigné est automatiquement sélectionné.

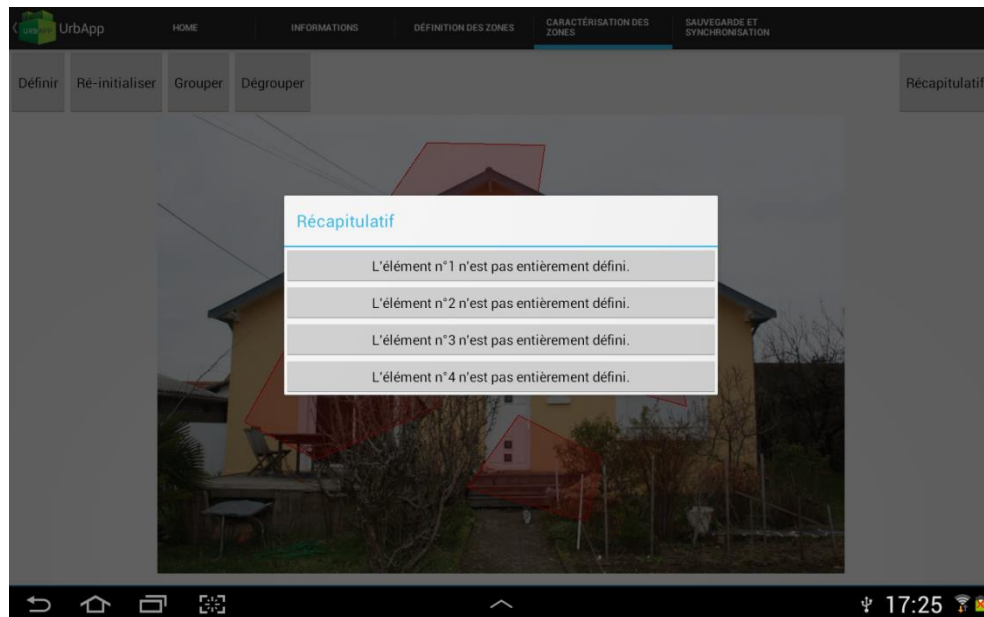


Figure 16 : Fenêtre de récapitulatif des zones

6.3) La géolocalisation d'un sol

Définition du problème :

Pour la géolocalisation d'un sol, les élèves de l'année dernière utilisaient juste quatre points que l'on plaçait sur la carte. Or, un sol peut nécessiter beaucoup plus que quatre points, ou alors juste trois. Il fallait donc enlever cette limite de quatre points et définir d'autres conditions quant à l'apparition des messages « Vous n'avez pas renseigné assez de points » et « Vous avez placé le nombre de points maximal » mais toujours garder la distinction entre la géolocalisation d'une façade et d'un sol.

Solution :

Le nombre de points que l'on doit placer dépend du choix que l'on fait au démarrage du projet, si nous étudions une façade ou alors un sol. Pour une façade, le nombre de points est de strictement deux, puisqu'il s'agit d'un mur, mais pour un sol il est de au moins de trois pour avoir un polygone. Dans le code, ce nombre limite de points que l'on place est défini par une variable, `nbPoints`, qu'y changeait de valeur selon le choix de la façade ou du sol. Une solution aurait été de mettre un nombre assez grand dans cette variable si l'on choisissait l'option sol, mais cela n'était pas très robuste et évolutif. J'ai donc d'abord essayé de réutiliser la partie du code concernant la création des différentes zones sur une image car on pouvait mettre autant de points que l'on voulait, mais les classes utilisées pour ces points n'étaient pas les mêmes, de plus, j'ai finalement trouvé une solution plus simple.

Afin de faire la différence entre le choix d'une façade ou d'un sol, j'ai d'abord créé une variable `selected` et l'ai initialisé à 0, puis cette variable vaut 1 si l'on choisit sol, et 2 si l'on choisit façade. Si j'ai choisi d'utiliser un entier et non un booléen, c'est dans un souci d'évolution de l'application. En effet, le fait d'utiliser un entier rend l'application plus extensible puisqu'il permet d'ajouter d'autres types de sélection dans le futur (par exemple toit qui vaudrait alors 3, etc). Une fois cela fait, il suffisait de mettre dans le code correspondant à l'ajout de points une condition comme quoi si `selected` vaut 1 ou 2. S'il vaut 1, et donc si c'est un sol, on ajoute des points sans se soucier du nombre de points déjà placé, s'il vaut 2, on peut ajouter des points tant qu'il n'y en a pas plus de 2 déjà placé.

Enfin, il fallait ajouter des conditions lors de la validation de la géolocalisation, en effet, on ne peut pas valider la géolocalisation d'un sol s'il n'y a que deux points de placés, voire même aucun. Pareil si c'est une façade, mais il faut ajouter une condition qui dit que s'il y a déjà deux points, on ne peut plus en placer.

Les classes que j'ai dû donc modifier sont les classes `GeoActivity`, du package `activities`, et `NbPointsGeoDialog` du package `dialogs`.

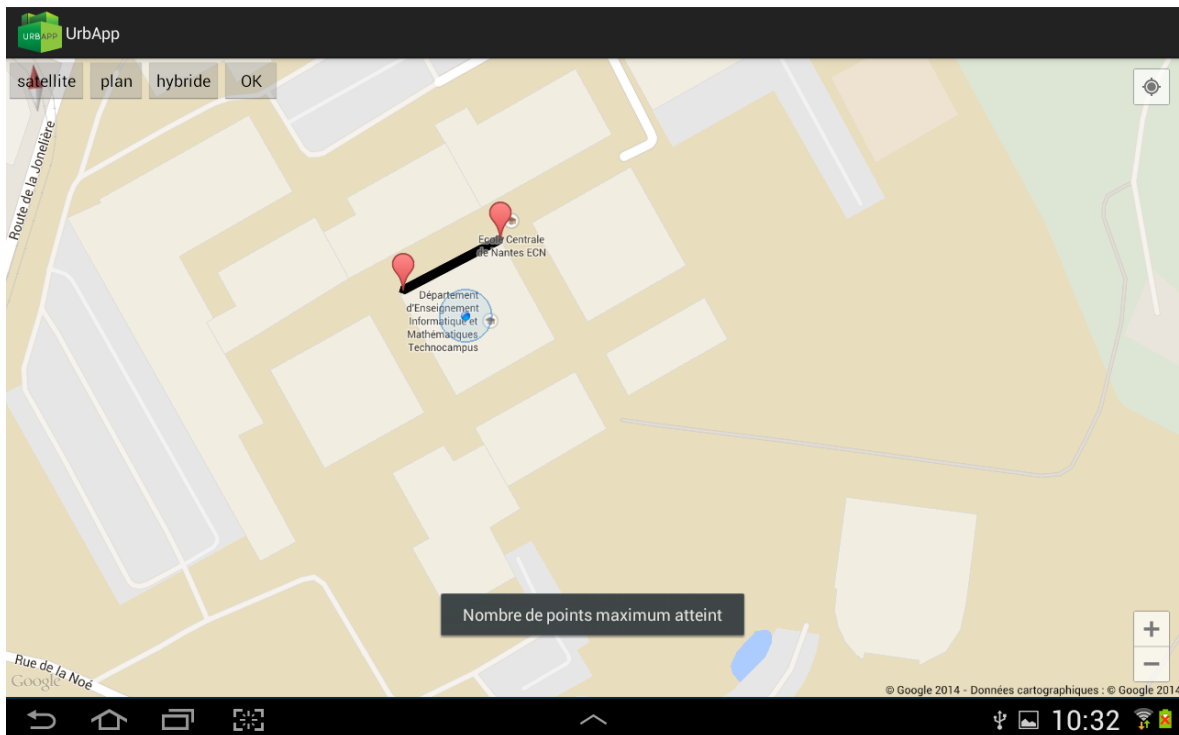


Figure 17 : Géolocalisation d'une façade avec message d'avertissement

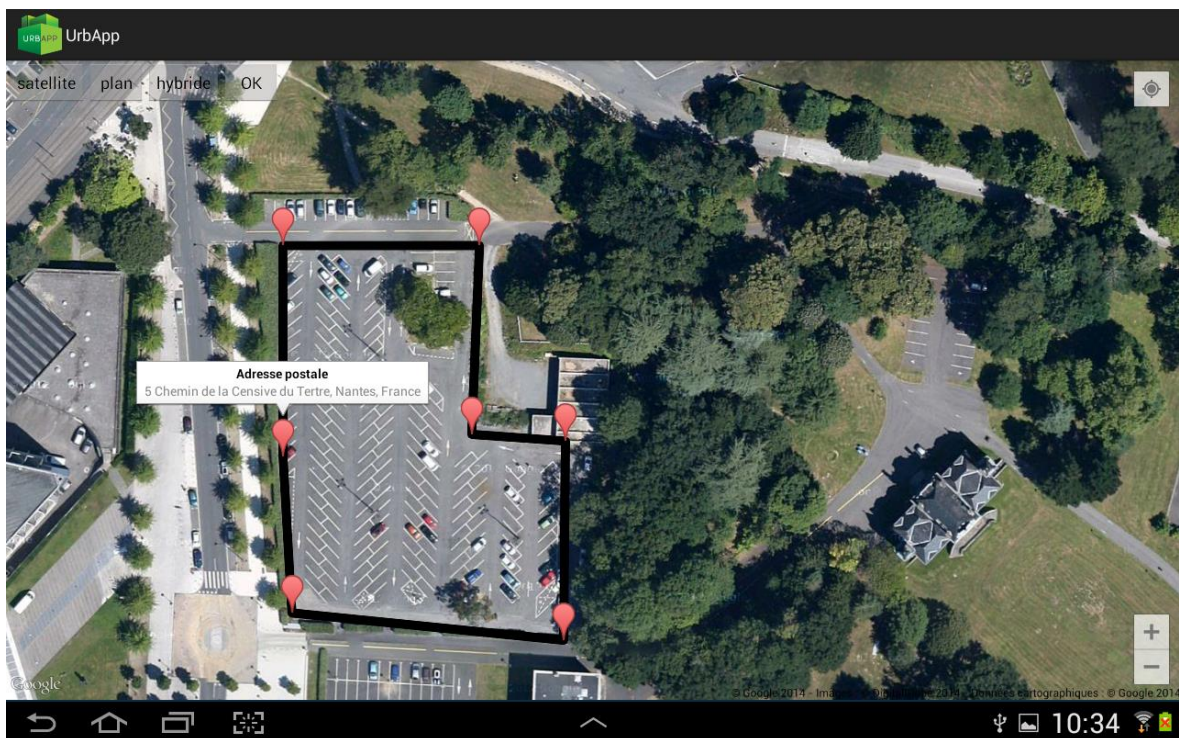


Figure 18 : Géolocalisation d'un sol avec plus de quatre points

On peut voir sur le schéma suivant les endroits de l'application où je suis intervenu (indiqué en rouge) :

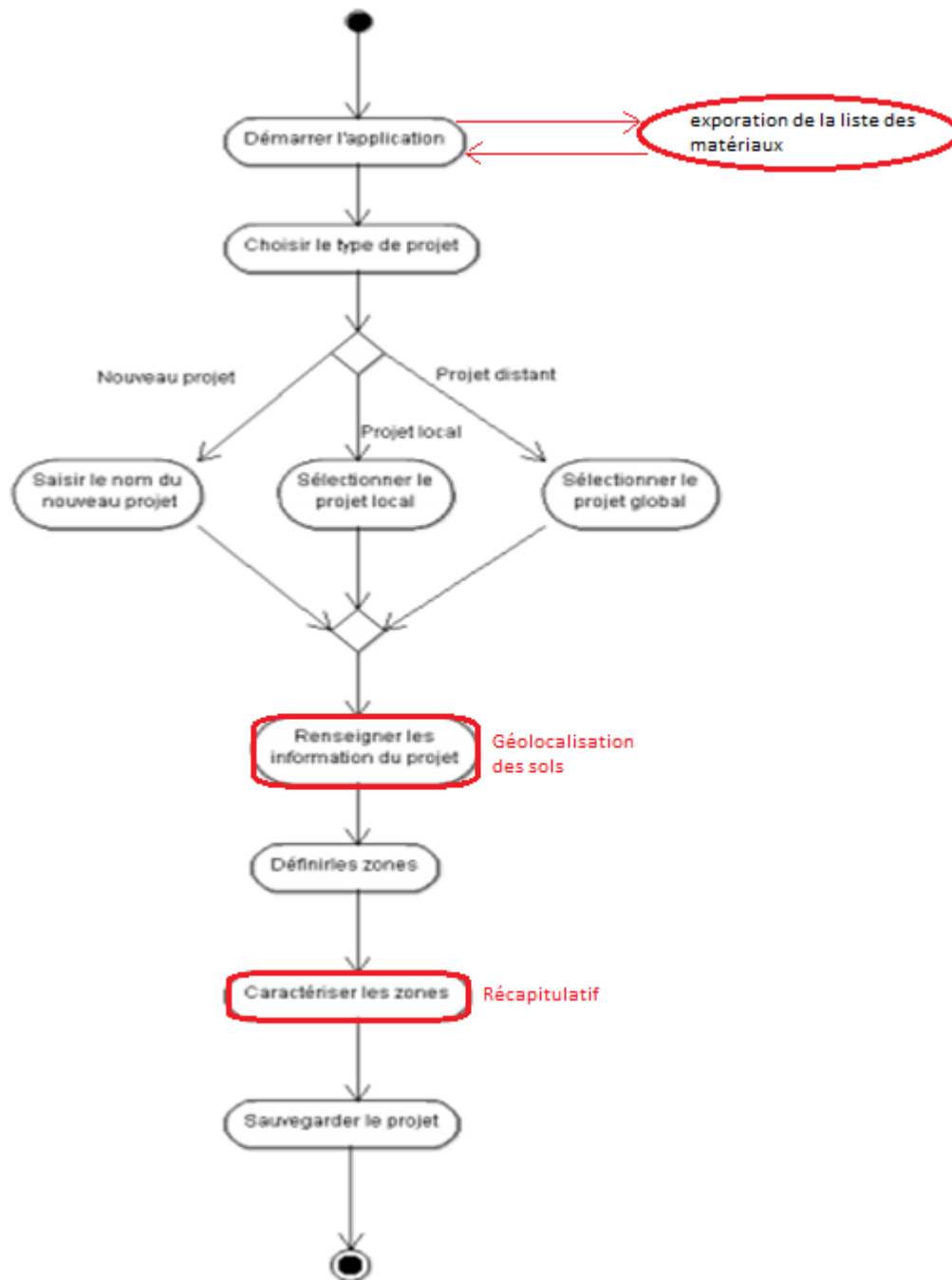


Figure 19 : Schéma d'activité de l'application avec les modifications apportées

Après cette phase de correctif de l'application, je suis passé sur la partie visualisation des données avec OrbisGIS. Cependant, j'ai rencontré un problème qui a retardé grandement le début de cette étape, la connexion à la base de données distante. En effet, OrbisGIS permet de se connecter à une base de données afin d'utiliser les tables pour les représenter. Cette base de données distante était hébergée sur un des serveurs de l'Ecole Centrale de Nantes, les anciens élèves qui ont commencé le projet ont laissé les identifiants de connexion, mais ceux-ci ne fonctionnaient pas. J'ai donc tout d'abord testé la connexion via un script Java, et il s'est avéré que le mot de passe que j'avais n'était pas le bon. J'ai contacté un des anciens élèves ayant travaillé sur ce projet pour avoir le bon mot de passe. Après avoir réessayé avec le bon mot de passe, la connexion n'était toujours pas possible. En effet, le port qui était utilisé avait été fermé. Une demande a donc été faite pour son ouverture. Une fois le port ouvert, la connexion au serveur a pu être établie, mais la base de données qu'il contenait était vide. La bonne base de données était en fait sur un autre port, il a donc fallu échanger les deux bases de données.

Une fois tout cela fait, j'ai pu finalement utiliser OrbisGIS avec les données contenues dans la base, et donc les données qui étaient saisies avec l'application.

6.4) Visualisation des données et génération des différents fichiers

Définition de la tâche :

Ce qu'il fallait faire maintenant était, dans un premier, d'arriver à visualiser les données contenues en base de données, puis de faire les opérations nécessaires pour pouvoir générer les différents fichiers utilisés par Solene-Microclimat. A cause des différents problèmes décrits avant, cette tâche a été la dernière abordée, et n'est pas finie à ce jour.

Avant de pouvoir extraire les fichiers voulus, il fallait tout d'abord visualiser les données afin d'avoir une idée de comment elles étaient construites et s'il est était utilisables. Pour cela, il suffisait de se connecter à la base de données, puis de sélectionner les tables que l'on voulait représenter. Il n'y que deux tables qui sont représentable par OrbisGIS sous forme de couche 2D, les tables GpsGeom et PixelGeom. Comme dit plutôt, GpsGeom contient les géolocalisations, et PixelGeom polygones des zones que l'on définit. Pour faire des tests, l'IRSTV m'a envoyé une base de données topographique sur l'Ecole Centrale, visible sur les images de présentation d'OrbisGIS. J'ai donc fait des relevés avec l'application sur les bâtiments de l'Ecole Centrale afin de pouvoir recouper ces deux bases.

Une fois mes relevés effectués, il faut afficher la table GpsGeom dans OrbisGIS, sélectionner ce qui correspond à notre relevé (GpsGeom étant les géolocalisation des projets, sa représentation est sous forme de lignes ou de polygones suivant si elle contient des sols et/ou des façades). A partir de cette sélection, on peut créer une nouvelle source grâce à une option d'OrgisGIS. Une fois la nouvelle source créé, il suffit de mettre les deux couches que l'on veut recouper dans le cadre Toc.

Pour ce qui est de la génération des fichiers, le shapefile peut se générer facilement à l'aide d'un clic droit/Export, cependant, pour les fichiers CIR et VAL, il faut utiliser un script.

Problèmes rencontrés :

J'ai eu beaucoup de problèmes avec OrbisGIS. Le premier a été la création de la nouvelle source. En effet, quand je voulais créer une nouvelle source, soit cela me faisait une erreur, soit elle était créée mais elle était vide. J'ai donc demandé de l'aide Monsieur Petit sur ce logiciel, qui m'a redirigé vers un forum d'aide où je posais mes questions. A l'heure actuelle, la génération des fichiers CIR et VAL n'a pas été encore abordée. De plus, lors de l'export du shapefile, une erreur est obtenu dont je ne connais pas encore l'origine.

J'ai quand même voulu recouper mes données avec la base topographique de l'Ecole Centrale sans passer par une sélection, et donc en utilisant toutes les données contenues dans GpsGeom, cependant, ces deux sources ne s'affichaient pas dans le même référentiel. La raison de cela est due au référentiel utilisé lors de la projection. En effet, pour représenter les données spatiales, un champ appelé the_geom est utilisé. Ce champ correspond à des latitudes et longitudes, mais il faut utiliser un système de projection pour les représenter, et il en existe une multitude.

A l'aide de la console SQL d'OrbisGIS, j'utilise des fonctions pour pouvoir connaître quel système de projection, et convertir les valeurs de the_geom de ma table GpsGeom vers le même format utilisé dans la base topographique. A ce jour cela n'est toujours pas résolu mais j'ai demandé de l'aide sur le forum que Monsieur Petit m'a envoyé.

7) Travail restant et évolutions possibles

Même si l'application fonctionne, il reste encore plusieurs choses qui peuvent être amélioré ou qu'ils restent à faire.

La première tâche serait de finir la chaîne de traitement qui va de la saisie des informations jusqu'au formatage des données pour Solene-Microclimat. En effet ceci est pour moi ce qu'il faut finir en premier puisqu'à l'heure actuelle la transition entre l'application et Solene-Microclimat n'est pas encore automatique.

En deuxième, il faudrait corriger l'exportation de la liste des matériaux afin que celle-ci prenne en compte les valeurs contenues dans conductivité, capacité thermique et masse volumique.

Enfin, une dernière tâche qui n'a pas pu être abordé faute de temps est la sélection de la couleur pour caractériser une zone. Actuellement, la sélection de la couleur se fait via une sélection dans un nuancé de couleurs. L'optimal serait que cette sélection se fasse automatique à l'aide d'une lecture des pixels de la zone.

Pour ce qui est des évolutions possibles, on pourrait imaginer la prise en compte de la hauteur lors l'acquisition des données sur les façades et sols. En effet, la coordonnée z n'est pas encore incorporée dans l'application.

III) Apports du stage

Ce stage d'une durée de dix semaines m'a permis d'enrichir mes connaissances informatiques. En effet, pendant dix semaines j'ai pu m'impliquer totalement dans un objectif, reprendre et améliorer une application Android.

Ce stage m'a permis d'apprendre de nouvelles choses, tout d'abord la programmation sous Android, qui est maintenant bien développé à travers le monde et peut être un grand plus de la connaître pour mes prochaines années d'études et d'entreprise. J'ai également une meilleure capacité d'adaptation puisque j'ai dû m'adapter à Android mais aussi à OrbisGIS.

Ce stage m'a aussi permis d'être plus rigoureux dans mes démarches et mon travail afin d'établir un plan de travail ou savoir par où commencer, rédaction d'un planning, estimation du temps de travail sur telle tâche etc.

Ce stage m'a également permis d'améliorer mon autonomie. Malgré le fait que l'informatique est un domaine où le travail d'équipe est presque indispensable, il faut néanmoins être capable d'être autonome afin d'être le plus efficace et le plus performant possible. Cette situation m'a obligé à approfondir mes recherches quand je rencontrais une erreur ou un problème, au sein d'une équipe, j'aurais tout d'abord exposé mon problème afin de voir si un des membres a une solution pour y remédier.

Finalement, je peux affirmer que ce stage m'a conforté dans mon choix d'orientation : faire un métier lié à la conception et à la programmation logicielle et pourquoi pas plus spécialement dans les applications Android.

IV) Bibliographie/Webographie

Programmation Android (Dernière utilisation : 08/05/14) :

<http://fr.openclassrooms.com/informatique/cours/creez-des-applications-pour-android>

PostGIS (Dernière utilisation : 02/05/14) :

<http://postgis.net/docs/manual-2.1/>

OrbisGIS (Dernière utilisation : 20/06/14) :

<http://www.orbisgis.org/documentation/>

<http://orbisgis.3871844.n2.nabble.com/>

<http://www.h2gis.org/docs/dev/home/>

GitHub (Dernière utilisation : 20/06/14) :

<https://github.com/>

Autres (Dernière utilisation : 28/05/14) :

<http://stackoverflow.com/>

V) Annexes

BERNARD Alexis



Cahier des charges (seulement la partie concernant mon stage)

7. Reprise du projet

7.1. Contexte du stage

Le stage consiste à reprendre l'application développée l'année dernière et y apporter des améliorations, ajouter de nouvelles fonctionnalités ainsi que diverses modifications si cela est demandé.

Ce qui a abouti de l'année dernière est une application déjà fonctionnelle.

- On peut prendre ou charger une photo depuis la galerie de la tablette.
- On peut charger un projet déjà commencé pour le consulter ou le compléter.
- On peut géo localiser la photo prise.
- Dire si c'est une façade ou un sol.
- Tracer des zones et les caractérisés (type, matériau, couleur).
- Sauvegarder son projet.
- Se synchroniser avec la base de données distante.

L'interface graphique est quant à elle satisfaisante. Le travail restant tourne autour de la base de données. Il faudrait ajouter des champs de caractérisation supplémentaires pour plus de précision. Mettre en place une extraction de ces données dans un format défini au préalable.

7.2. Fiche signalétique

Étudiants

Bernard Alexis alexis.bernard@ec-nantes.fr

Équipe pédagogique

Vincent Tourre Encadrant stage vincent.tourre@ec-nantes.fr

Myriam Servières Encadrant stage myriam.servieres@ec-nantes.fr

7.3. Spécifications

7.3.1. Visualisation des données

Une des premières choses à faire est la visualisation des données enregistrées dans la base de données distante. Ces données étant des données géométriques, il faut utiliser un Système d'Information Géographique (SIG). Pour ce stage, nous allons utiliser le logiciel de l'IRSTV, futur clients de l'application, à savoir OrbisGIS. OrbisGIS sert à visualiser des données topographiques en 2D ou 3D. Il utilise les fichiers shapefile, ou fichier de forme en français. La première étape consiste donc à faire transiter les données vers ce format.

7.3.2. Conversion vers Solene-Microclimat

Rappelons-le, le but final de cette application est d'utiliser les données entrées pour faire des tests en terme d'éclairage, de thermiques ou d'énergie. Ces tests sont effectués par le logiciel Solene-Microclimat. C'est donc la deuxième étape, faire transiter les données d'OrbisGIS vers Solene-Microclimat. Solene utilise plusieurs fichiers en XML pour les matériaux par exemple ainsi que des fichiers au format spécifique pour les positions ou les formes.

7.3.3. Partie debug de l'application

L'application est fonctionnelle, mais plusieurs choses peuvent être améliorés ou corrigés. Tout d'abord la sélection de la couleur. Lors de la caractérisation d'une façade, quand on sélectionne une zone, on nous demande la couleur avec une sélection via un nuancier de couleurs. Une solution optimale serait de prélever directement la couleur via la photo.

Il y a aussi la gestion de la hauteur. En effet, pour l'instant l'application ne gère pas la 3D, on ne sait donc pas précisément la position des fenêtres sur une façade par exemple, seulement une projection au sol.

Une fonctionnalité de l'application permet de récapituler les zones d'une photo qui sont caractérisées ou non, mais celle-ci ne fonctionne pas, ils font donc la corriger.

L'application permet également de faire des projets sur des sols, mais quand on sélectionne cette options, nous avons le droit à seulement 4 points à placer sur la géolocalisation, or un sol peut en nécessiter plus.

Il pourra aussi être bien de pouvoir récupérer la liste des matériaux que l'on peut choisir.

Enfin, permettre à un projet d'avoir plusieurs photos, comme par exemple la façade nord, est, ouest et sud d'une maison.

7.4. Planning

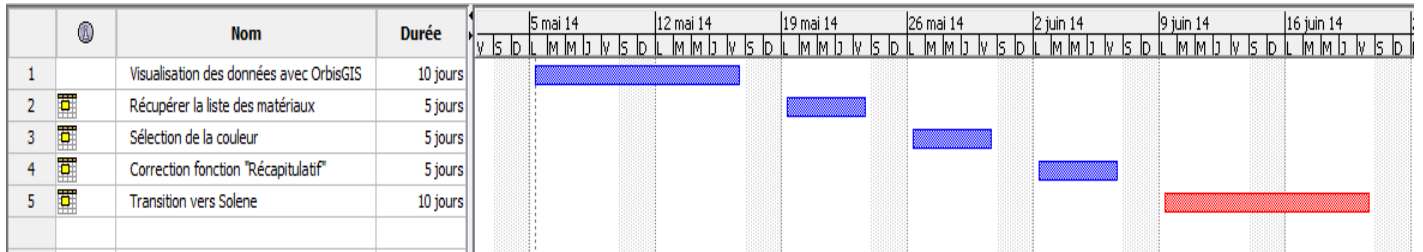


FIGURE 20 : Planning indicatif



GUIDE DE L'UTILISATEUR

Guide réalisé par les élèves de l'année dernière, j'y ai ajouté l'explication sur
l'exportation ainsi que le récapitulatif

SOMMAIRE

Préambule	2
Lancement de l'application.....	4
Charger/prendre une photo.....	5
Charger un projet local.....	5
Charger un projet distant	5
Charger une photo à partir d'un projet.....	6
Charger les matériaux & types du serveur	6
Informations	7
Géolocalisation	7
Définition des éléments.....	9
Qu'est-ce qu'un élément ?	9
Comment définir des éléments ?	9
Créer une zone	9
Conditions de validation.....	10
Editer une zone	11
Opérations d'édition	12
Annulation de modification.....	12
Supprimer une zone	12
Superposition de zones	13
Caractérisation des zones	14
Définir la ou les zones sélectionnées	14
Ré-initialiser la ou les zones sélectionnées	15
Grouper les zones sélectionnées.....	15
Récapitulatif des caractérisations	15
Sauvegarde et synchronisation.....	16
Synchronisation.....	16
Index	17

LANCEMENT DE L'APPLICATION

Au démarrage d'UrbApp vous arrivez sur la page d'accueil. Quelques remarques d'ordre général :

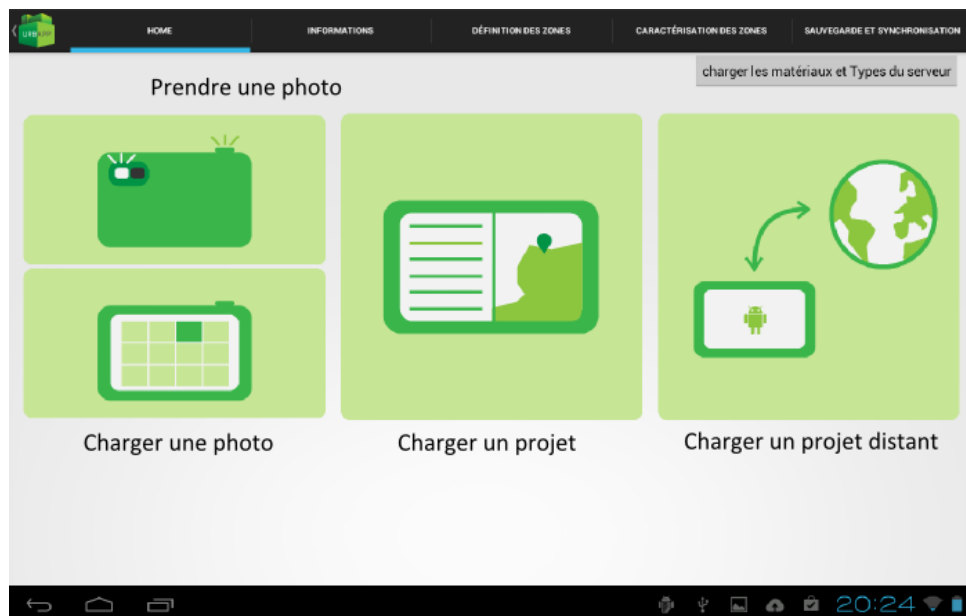
Si votre appareil n'est pas connecté à Internet l'application vous demandera d'y remédier car une connexion est Indispensable pour que votre travail soit enregistré.

L'application fonctionne en mode paysage, inutile de tourner votre appareil en mode portrait. Pour votre confort veuillez toutefois autoriser dans les paramètres d'Android la rotation de l'écran, dans le cas contraire vous devrez tourner votre appareil pour prendre ou charger des photos.

L'application est optimisée pour l'utilisation sur les tablettes Samsung Note et Acer de 10".

Vous avez le choix entre prendre une photo via un capteur de votre appareil, charger une photo présente dans la mémoire de votre appareil, charger un projet local (de votre appareil) ou distant (du serveur).

Il est impossible de passer à un autre onglet avant d'avoir sélectionné une photo.



La page d'accueil, avec les différentes actions légendées et la barre d'actions en haut.

CHARGER/PRENDRE UNE PHOTO

Charger ou prendre une photo sont des opérations qui font appel à des applications spécifiques de votre appareil ou sa version d'Android, veuillez-vous référer à leur documentation propre pour plus de détails.

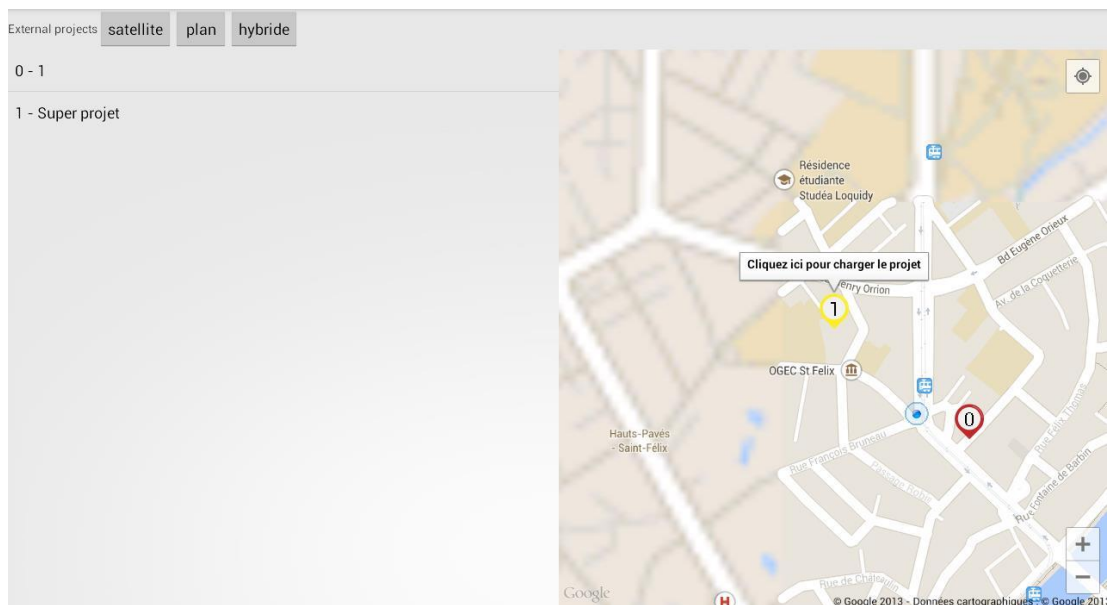
Attention : le loader peut éventuellement vous proposer des images qui ne sont pas présentes sur l'appareil (qui se trouvent sur un dossier distant Picasa par exemple) et ne peuvent pas être utilisées par UrbApp.

CHARGER UN PROJET LOCAL

Cette page affiche tous les projets présents sur votre appareil, ainsi que leur localisation sur une carte, vous permettant d'avoir un aperçu rapide du travail déjà effectué. Vous pouvez vous géolocaliser en cliquant sur l'icône en haut à droite, par défaut, c'est la France qui est affichée.

CHARGER UN PROJET DISTANT

Cette page affiche tous les projets présents sur le serveur, ainsi que leur localisation sur une carte, vous permettant d'avoir un aperçu rapide du travail déjà effectué. Vous pouvez vous géolocaliser en cliquant sur l'icône en haut à droite, par défaut, c'est la France qui est affichée.

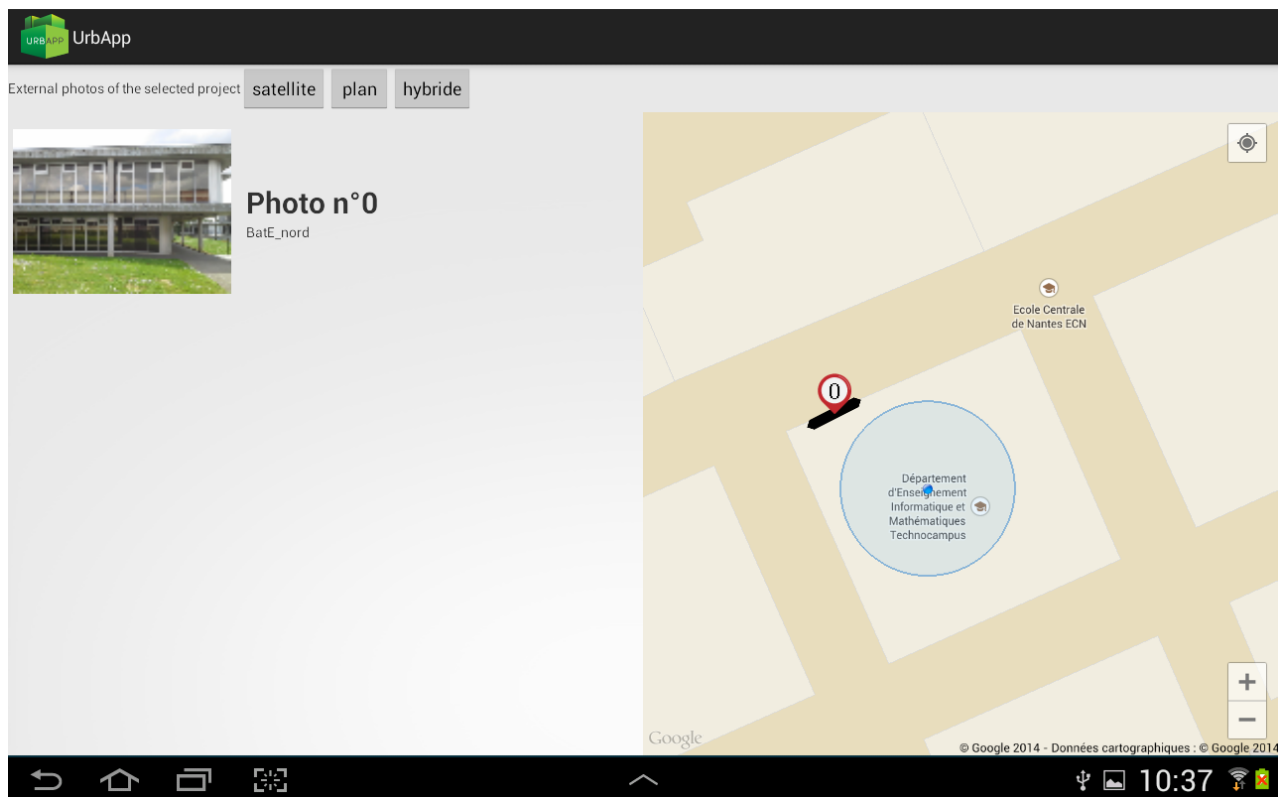


Charger un projet

CHARGER UNE PHOTO A PARTIR D'UN PROJET

Pour ces deux fonctionnalités, il faut cliquer sur le marqueur de la carte et valider le projet en cliquant sur le texte s'affichant.

Il convient alors de choisir la photo sur laquelle on souhaite travailler de la même manière.



Charger une photo depuis un projet

EXPORTER LA LISTE DES MATERIAUX

Ce bouton sert à exporter la liste des matériaux sous un format XML. Le fichier est créé sous le nom de Liste_matériaux.xml et se trouvera dans la racine de votre appareil.

INFORMATIONS

Vous arrivez sur cet onglet après avoir chargé une image. Il s'agit maintenant de renseigner l'application sur cette image. Toutes les informations demandées sont indispensables pour enregistrer le travail mais peuvent être renseignées ultérieurement.

Auteur : Team UrbApp
Description de l'image : Exemple
Projet : Nom de projet
Adresse : Adresse *
Complément *
Aucune Géolocalisation

GEOLOCALISATION

Une des informations que nous vous demandons de renseigner est la géolocalisation de l'image. Une fenêtre dédiée permet d'obtenir des coordonnées relativement précises à l'aide d'une carte, plus précises et fiables que celle d'un simple localisateur GPS.

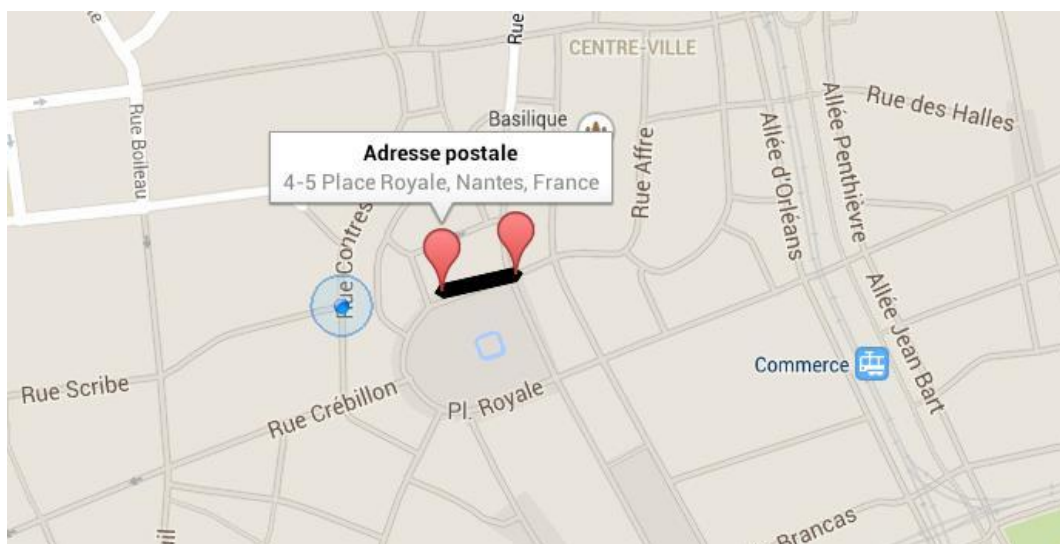
L'objectif est de relier l'image à des lieux réels. Il s'agit de donner plusieurs points pour repérer les extrémités d'un élément vertical (2 points) ou horizontal (4 points). Le nombre de points est demandé au lancement de la géolocalisation.



La carte de géolocalisation (GoogleMaps)

GUIDE D'UTILISATION - URBAPP

Différents types de cartes sont disponibles dans le coin supérieur gauche pour pouvoir choisir le type de carte idéal pour repérer le contexte de l'image. Dans le coin supérieur droit un bouton permet de centrer la carte sur votre position GPS, si la localisation GPS est activée dans votre appareil. Vous pouvez réduire le zoom en pinçant la carte ou l'augmenter en écartant vos doigts, ou encore cliquer respectivement sur les boutons - et + dans le coin inférieur droit.



Exemple de localisation d'une façade (2 points). L'adresse affichée en infobulle sera utilisée pour renseigner l'adresse de l'image.

Pour effectuer la géolocalisation touchez les points d'extrémités de l'image. Vous pouvez éditer ces points en appuyant longuement dessus puis les faisant glisser tout en maintenant votre doigt appuyé.

Pour valider la géolocalisation cliquez sur **OK** dans le coin supérieur gauche. Une fois la géolocalisation effectuée le bouton change d'aspect mais il reste possible de l'effectuer à nouveau en cliquant sur le bouton "Projet géolocalisé". Une fois le projet géolocalisé, un bouton "==" apparaît vous renvoyant à la prochaine étape.

Auteur :	<u>Team UrbApp</u>
Description de l'image :	<u>Exemple</u>
Projet :	<u>Nom de projet</u>
Adresse :	<u>2 Rue des Vignes, <u>Caillouet-Or</u></u>
	<u>Complément *</u>
	Projet Géolocalisé
	==>

DEFINITION DES ELEMENTS

Maintenant que vous avez renseigné les informations à propos de l'image, il s'agit de représenter des éléments.

QU'EST-CE QU'UN ELEMENT ?

Un élément est une zone de la photo, délimitée par un polygone. C'est ce polygone qui est défini dans l'onglet **DEFINITION DES ZONES**. Des informations sur la nature de l'élément découpé par la zone seront apportées dans l'onglet de **CARACTERISATION DES ZONES**.



Exemple d'élément : délimitation d'une fenêtre

COMMENT DEFINIR DES ELEMENTS ?

L'onglet **DEFINITION DES ZONES** d'UrbApp vous permet de créer, éditer, supprimer des zones. Ces opérations sont décrites ci-dessous.

CREER UNE ZONE

Débuter la création de zone se fait par un appui *court* sur l'image. Le premier point de la zone est alors créé. Appuyer à nouveau sur l'image crée un nouveau point. Ce nouveau point est ajouté à la suite du dernier point, représenté en jaune.

Le bouclage potentiel est représenté par un trait en pointillés jaune.

GUIDE D'UTILISATION - URBAPP



Exemple : le bouclage potentiel. La zone est prête à être validée.



Exemple : le bouclage potentiel. Si on valide maintenant la zone ne délimitera pas la fenêtre.

Il est possible d'éditer le tracé d'une zone en pleine création en utilisant les options d'édition, décrites dans le paragraphe Opérations d'édition.

Faire un appui *court* sur le premier point (celui qui est relié au dernier point par un pointillé jaune) permet, si les conditions sont remplies, de valider le bouclage. Le bouton **VALIDER** remplit le même rôle.

La zone est maintenant affichée, avec une couleur par défaut tant que vous ne l'aurez pas définie dans l'onglet **CARACTERISATION DES ZONES**.

CONDITIONS DE VALIDATION

UrbApp ne permet pas l'enregistrement de zones ne remplissant pas les conditions suivantes :

- Au moins 3 points, pour avoir un polygone.

GUIDE D'UTILISATION - URBAPP

- Pas d'intersection entre les segments du polygone. Dans le cas d'une intersection, les segments sont représentés en bleu.



Exemple : intersection entre deux segments. Le bouton validation est inaccessible.

Les opérations d'édition décrites dans le paragraphe éponyme vous permettront de corriger le tracé.

EDITER UNE ZONE

Débuter l'édition d'une zone se fait par appui *long* sur zone. Les points de construction de la zone apparaissent à nouveau, mais plus de "dernier point" ou de "bouclage potentiel" comme en création.



Exemple : édition de zone

Les opérations décrites ci-dessous sont également réalisables lors de la création d'une zone.

OPERATIONS D'EDITION

Pour déplacer un point appuyez dessus, puis faites glisser votre doigt tout en le maintenant appuyé. Le tracé de la zone est actualisé au cours du déplacement. Relâchez pour terminer le déplacement.

Pour insérer un point entre deux autres il s'agit de déplacer un point miniature.

Pour supprimer un point sélectionnez un point par un appui **long**, sans déplacement, puis appuyez sur le bouton **SUPPRIMER**. Un point sélectionné est représenté plus gros.



Exemple : Le point dans le bord supérieur droit est sélectionné. Les points les plus proches, qui ne sont pas dans les coins, sont des points d'insertion. Les autres sont des points "normaux".

Pour désélectionner un point touchez à nouveau l'image.

ANNULATION DE MODIFICATION

Il est possible d'annuler une action n'ayant pas été validée en appuyant sur le bouton **ANNULER DERNIERE ACTION**.

Un appui sur le bouton **QUITTER** entraîne l'annulation de toutes les actions non validées. Sélectionner une autre zone (appui long) que celle en cours d'édition annule également les modifications.

SUPPRIMER UNE ZONE

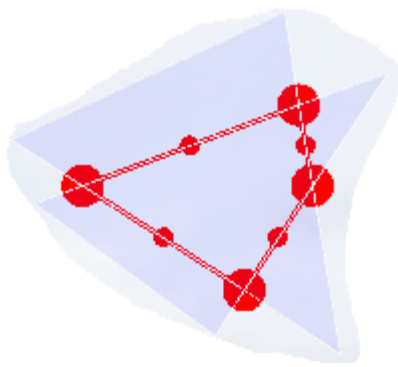
Pour supprimer une zone il s'agit de sélectionner une zone par un appui **long** et cliquer sur **SUPPRIMER**.

Attention ! Aucune confirmation ne vous sera demandée.

SUPERPOSITION DE ZONES

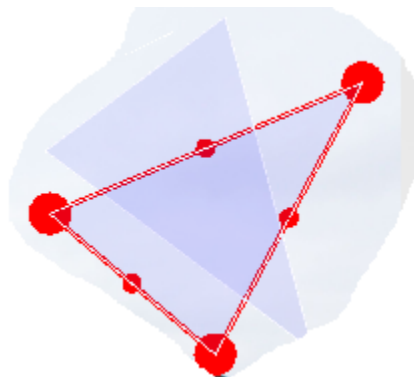
Au cours de la création ou de l'édition d'une zone vous pouvez être amené à la faire croiser une autre zone ou même à faire qu'une zone couvre une autre. Après validation UrbApp vous proposera deux alternatives :

- découper les zones de telle sorte que les intersections forment de nouvelles zones et soient amputées des autres zones. Cette option est recommandée si les deux zones caractérisent des éléments qui sont dans le même plan. Elle permet d'éviter des erreurs dans le calcul des surfaces lors du traitement de ces informations.



Exemple : les deux triangles ont été découpés et une nouvelle zone est apparue au milieu.

- Laisser les zones se superposer. Cette fonction est à utiliser dans le cas d'un balcon par exemple qui vient s'ajouter devant la façade mais n'est pas recommandée pour la caractérisation d'une façade simple.



Exemple : pas de découpe

- En cliquant hors de la boîte de sélection vous pouvez annuler la validation.

A l'aide de cette option vous pouvez créer des zones encerclant un espace, en créant une petite zone à l'intérieur d'une grande ou une grande zone autour d'une petite. La petite peut être supprimée, la grande encerclera du vide.

CARACTERISATION DES ZONES

Une fois les zones délimitées il s'agit de renseigner les informations qui y sont liées : type d'élément circonscrit (toit, façade ou sol) ainsi que le matériau qui le compose. Dans cet onglet, contrairement au précédent, vous pouvez sélectionner plusieurs zones simultanément.



Exemple : sélection de toutes les fenêtres (en rouge foncé)

Sélectionnez tout d'abord une ou plusieurs zones en les touchant. Vous pouvez annuler la sélection d'une zone en la touchant une nouvelle fois. Choisissez ensuite l'une des actions suivantes en cliquant sur le bouton associé.

DEFINIR LA OU LES ZONES SELECTIONNEES

Il s'agit de décrire le type (toit, façade ou sol), le matériau et d'attribuer une couleur à l'élément délimité par les zones sélectionnées. S'effectue en cliquant sur le bouton **DEFINIR**.

les types de zones et matériau sont chargés à partir du serveur, si vous voulez en rajouter, il faudra donc demandé à l'administrateur de rajouter un nouveau type d'élément (respectivement un nouveau matériau)

Choix des caractéristiques des zones sélectionnées

Type de la zone : Façade

Matériau de la zone : Verre

Couleur de la zone :

Valider

Exemple : définition des fenêtres

RE-INITIALISER LA OU LES ZONES SELECTIONNEES

Réinitialise les informations à propos des zones sélectionnées.

Attention ! Aucune confirmation ne vous sera demandée !

GROUPER LES ZONES SELECTIONNEES

Vous pouvez lier les zones sélectionnées de manière plus ou moins profonde :

- liaison forte : les zones sont fusionnées et n'en forment plus qu'une. L'opération est irréversible.
Attention ! Lors d'une liaison forte toutes les informations sur toutes les zones groupées (types, matériaux et couleurs) sont perdues. Pensez donc à les lier avant de les caractériser.



*Exemple : édition d'une fenêtre après application d'une liaison forte dans l'onglet **Définition des zones**. Toutes les fenêtres ne constituent plus qu'une zone.*

- liaison faible : les zones sont liées pour la sélection, mais restent séparées, au contraire de la liaison forte. Sélectionner une des zones dans l'onglet Caractérisation des zones sélectionne également les zones liées, mais pas dans l'onglet Définition des zones.

RECAPITULATIF DES CARACTERISATIONS

Appuyer sur ce bouton liste les zones qui n'ont pas été entièrement définies. La liste est cliquable et vous permet de sélectionner les zones en question.

SAUVEGARDE ET SYNCHRONISATION


Une fois les informations à propos de l'image (2nd onglet) et les zones définies (onglet 3) et caractérisées (onglet 4) il s'agit d'enregistrer son travail.

SYNCHRONISATION

Une fois le travail réalisé, ce fragment vous offre la possibilité de sauvegarder votre travail.

Pour pouvoir sauvegarder il faut que les informations du projet soient rentrées (cf onglet information) ainsi qu'au moins un élément soit créé et défini. Si ces informations ne sont pas rentrées, un message vous invitant à le faire sera retourné.

En cas de succès, la sauvegarde s'effectue vers la base de données locale et distante. Un message de succès vous informera que la synchronisation s'est déroulée sans encombre.

- 
- + Caractériser une zone : 14
 - + Charger une photo : 5
 - + Charger un projet : 5
 - + Créer une zone : 9
 - + Editer une zone : 11
 - + Élément : 9
 - + Enregistrer le travail : 16
 - + Façade : 14
 - + Fusionner des zones : 15
 - + Géolocalisation : 7
 - + Grouper des zones : 15
 - + Informations : 7
 - + Intersections : 10
 - + Matériaux : 14
 - + Prendre une photo : 5
 - + Projet : 5
 - + Récapitulatif des zones : 15
 - + Sauvegarder le travail : 16
 - + Sol : 14
 - + Supprimer un point : 12
 - + Supprimer une zone : 12
 - + Toit : 14
 - + Type de zone : 14
 - + UrbApp : 2
 - + Valider une zone : 10
 - + Zone : 9

PROJET ANDROID D'EXTRACTION DE CARACTERISTIQUES DE FACADES ET SOLS SUR SITE A PARTIR D'IMAGES

Guide du développeur



Membres Projet
Alexis BERNARD

Équipe pédagogique
Myriam SERVIERES
Vincent TOURRE

Table des matières

1. Licence de l'application	3
2. Compilation	3
3. Organisation du code	4
4. Détails concernant le fonctionnement du code source	4
4.1. Utilisation de la Google Maps API	4
4.2. Géolocalisation.....	5
4.3. Base de données	5
4.3.1. Fonctionnement général	5
4.3.2. Modification de la base de données.....	6
4.4. Fragment Home	6
4.5. Fragment Information	6
4.6. Fragment Zone.....	7
4.6.1. Chargement d'image	7
4.6.2. Gestion des évènements.....	7
4.6.3. Procédure d'affichage	7
4.6.4. Annulation des actions effectuées.....	8
4.6.5. Validation des modifications	8
4.6.6. Gestion des intersections entre zones	8
4.6.7. Gestion des polygones croisés	8
4.7. Fragment Characteristics.....	9
4.7.1. Définition des caractéristiques d'un élément.....	9
4.7.2. Regroupement de PixelGeom	9
4.8. Fragment Synchronization	9
5. Script de connexion à la base de données.....	10
6. Schémas de la base de données	11

Guide réalisé par les élèves en charge du projet l'année dernière

1. Licence de l'application

Cette application Android est **distribuée sous la licence libre CeCILL** (Plus d'informations : <http://www.cecill.info/>). Il s'agit d'une licence libre compatible avec la licence GNU GPL et adaptée au droit français.

Vous pouvez donc librement utiliser, modifier et redistribuer cette application. Le code de cette application **utilise certaines librairies externes** :

- Android Color Picker (<http://code.google.com/p/android-color-picker/>)
 - Distribuée sous la licence Apache 2.0.
 - Utilisée pour l'affichage de la boîte de dialogue de choix des couleurs.
- GSON (<https://code.google.com/p/google-gson/>)
 - Distribuée sous la licence Apache 2.0.
 - Utilisée pour transformer des objets Java en leur représentation JSON.
- Java Topology Suite (<http://www.vividsolutions.com/jts/JTSHome.htm>)
 - Distribuée sous la licence GNU LGPL.
 - Utilisée pour le calcul d'intersection et d'union des Geometry (PixelGeom), ainsi que pour les convertir dans leur représentation WKT.
- GoogleMaps Android API v2 (<https://developers.google.com/maps/documentation/android/>)
 - Utilisée pour l'affichage de carte et la géolocalisation des projets et photos sur cette carte.

2. Compilation

Cette application a été développée et compilée avec **les versions 18 puis 19 de l'API Android** (Android 4.3 et 4.4). Cependant elle est compatible avec les versions d'Android supérieures à la version Android 4.0.3 (API 15).

Comme l'application utilise une Google Maps API, elle ne peut pas être utilisée directement, il faut d'abord obtenir une clé API pour autoriser l'application à utiliser cette API. Référez-vous à la section 4.1 pour plus de détails.

3. Organisation du code

Les différentes classes de l'application ont été réparties dans **plusieurs paquets** pour une meilleure lisibilité et compréhension :

- *Activities* : Contient toutes les activités de l'application ;
- *Db* : Contient toutes les classes décrivant la base de donnée locale ;
- *Dialogs* : Contient l'implémentation de toutes les boîtes de dialogue ;
- *Fragments* : Regroupe les fragments de l'application ;
- *Listener* : Contient la classe définissant le *listener* pour la barre de menu principale ;
- *SyncToExt* : Contient la classe permettant la synchronisation distante ;
- *Utils* : Contient la plupart des fonctions utilitaires ;
- *Utils.ColorPicker* : Contient les fonctions permettant l'affichage de la boîte de dialogue de choix des couleurs ;
- *Zones* : Contient les fonctions permettant la définition des PixelGeom par l'utilisateur (position des points délimitant les zones, fonction utilitaire d'intersection et d'union, enregistrement des types et matériau...).

Pour une meilleure compréhension du code, nous vous invitons à consulter **le rapport ainsi que les commentaires et la Javadoc** présente dans le code de l'application.

4. Détails concernant le fonctionnement du code source

4.1. Utilisation de la Google Maps API

Voici la marche à suivre pour faire fonctionner la Google Maps API :

- 1) Télécharger le Google Play Service library depuis l'Android SDK manager (dans extra).
(Plus d'infos sur cette page : <http://developer.android.com/google/play-services/setup.html>)
- 2) Une fois téléchargé, ajouter la bibliothèque (dans votre dossier <android-sdk>/extras/google/google_play_services/libproject/google-play-services_lib/) comme un import d'un projet android en faisant bien attention à COPIER la bibliothèque dans votre workspace (en cochant une petite case dans la fenêtre).
- 3) Vérifier en faisant un clic droit sur la bibliothèque, propriétés, que « Is Library » est bien coché (dans l'onglet android, qui doit être lancé par défaut normalement)
- 4) Clic droit sur le projet, propriétés, rajouter la library Google Play service en cliquant sur le bouton add. Vérifier si le project Build Target est bien sur Google APIs et non pas Android 4.x.

Maintenant, il faut récupérer une clé API depuis le compte gmail du groupe :

urbapp.ecn@gmail.com

Mot de passe : lr86DgQs

Pour cela il faut rajouter votre clé SHA1 sur cette page :

https://cloud.google.com/console?redirected=true#/project/165657011840/apiui/app?show=a_llapp

Rajouter une registered app, en cliquant sur +.

Pour connaître la clé SHA1, suivez ce tutoriel :

https://developers.google.com/maps/documentation/android/start#display_your_apps_certificate_information (cliquer sur Displaying the debug certificate fingerprint)

Là, vous aurez votre clé SHA1 et il faudra lui rajouter le nom du package `com.ecn.urbapp` lorsque vous la rajouterez aux autres clés acceptées.

Normalement, vous n'aurez pas à changer la clé API dans l'application et celle-ci fonctionnera normalement.

4.2. Géolocalisation

L'activité `GeoActivity`, dans le package `activity` est appelé à la fois en tant qu'activité, lors de la demande d'ajouts des points de la zone/façade, depuis le fragment `information` mais aussi sous la forme d'une classe plus classique dans toutes les activités de charger d'un projet/photo local ou depuis la base de donnée externe.

La variable `nbPoints` définit le nombre de marqueurs que l'utilisateur doit renseigner, exactement, elle n'est utilisée que pour les types façade car un sol n'a pas de limite de points.

4.3. Base de données

4.3.1. Fonctionnement général

La base de données locale est définie dans le package `com.ecn.urbapp.db`.

Les informations stockées dans les champs *static* correspondent au cache de l'application, on enregistre ces informations dans la base de données seulement lors de la synchronisation (en utilisant le `Fragment Save`).

On peut aussi exporter ces informations vers la base de données distante.

La classe `mysqliteHelper` déclare l'architecture de la base de données : nom des tables, types et noms des colonnes. Elle permet également la création de la cette base de données via des requêtes *create*.

La classe `Localdatasource` permet l'accès à la base de données et l'exécution de requête de contenu comme *update*, *insert* ou *select*.

Les classes héritant de la classe `DataObject` simulent le fonctionnement d'une base de données objet en instanciant les objets dans le code. Une fois toutes les modifications effectuées, c'est-à-dire une fois que tous les objets seront instanciés et stockés dans les champs `static` de `MainActivity`, on utilise ces éléments et on enregistre leurs attributs dans la base de données locale via des requêtes pré-écrite en utilisant la méthode `saveToLocal` et la classe `LocalDataSource`.

4.3.2. Modification de la base de données

Cas de l'ajout d'une table :

Classe *SQLiteOpenHelper*

- ❑ ajouté une nouvelle chaîne contenant le nom de la table :
 - o `public static final String TABLE_NOM= "Nom";`
- ❑ ajouté des chaînes pour les colonnes. Il faut autant de chaîne que de colonne, sans re-déclarer les clés étrangères :
 - o `public static final String COLUMN_COLONNE1 = "nom_colonne1";`
- ❑ ajouté la requête de création de la table et les éventuelles nouvelles clés étrangères dans les tables existantes.

Classe *LocalDataSource*

- ❑ ajouter le tableau contenant l'ensemble des colonnes de la table (et éditer les autres tableaux dans le cas de l'ajout de clés étrangères.
- ❑ créer les fonctions nécessaires pour accéder aux données de la nouvelle table dans cette classe.

Classe *Objet* :

- ❑ créer une nouvelle classe héritant de *DataObject* portant le nom de la table et ayant comme attributs ses colonnes.

Pour éditer une table, il suffit d'adapter les éléments présentés ci-dessus.

4.4. Fragment Home

Ce fragment est la page d'accueil de l'application. Elle contient quatre images cliquables qui lanceront les activités de prise d'une photo, de chargement d'une photo existante, de chargement d'un projet local et de chargement d'un projet distant.

Lors du clic sur l'image de prise de photo, la méthode fournit à l'application native le chemin absolu de sauvegarde de l'image.

4.5. Fragment Information

Ce fragment sert à définir l'ensemble des informations relatives au projet.

Les valeurs des champs "descriptions de l'image", "Auteur", "Adresse" sont sauvegardées dans l'objet `com.enc.urbapp.db.Image` correspondant. Le champ "Projet" est lui sauvegardé dans l'objet `com.ecn.urbapp.Projet` correspondant.

La sauvegarde de ces informations se fait lors que l'appel à la fonction `onStop()` qui est appelé par android lors de l'arrêt du fragment. Dans le cas où aucun projet n'est présent, un nouveau sera créé et associé à la photo.

Le chargement des informations se fait lors de l'appel à la fonction `onStart()`.

Le bouton de géolocalisation (Toggle Button) appellera lors du clic l'activité de géolocalisation. Une fois le projet géo-localisé, il changera d'état et le bouton lançant le fragment de définition des zones apparaîtra. Afin que le bouton ne change pas d'état directement lors du clic comme il devrait le faire normalement, on l'oblige à reprendre son état antérieur juste après le clic et avant de lancer la géolocalisation.

4.6. Fragment Zone

Ce fragment a pour fonction de délimiter des *Element* par leur géométrie décrite dans *PixelGeom* à l'aide de polygones construits point par point.

4.6.1. Chargement d'image

Le chargement de l'image utilise de nombreuses classes Android, et en effet il comprend de nombreuses étapes. L'image est récupérée du MainActivity sous la forme d'une chaîne de caractères indiquant le chemin absolu menant au fichier voulu. Cette chaîne est utilisée pour récupérer un fichier dans la classe BitmapLoader, qui ne provient pas de l'API d'Android mais de l'aide de Google pour les développeurs.

La classe BitmapLoader permet de récupérer une image de taille réduite à l'espace disponible dans UrbApp, entré en paramètre. La détermination de cet espace représente une gageure puisqu'il n'est possible de connaître cet espace qu'une fois l'image chargée. Il est estimé dans l'application à l'aide des dimensions de l'écran, réduit par la taille supposée des barres de menus et de boutons. L'image est chargée sous forme de Bitmap, l'opération de traduction demandant un espace mémoire non négligeable qui peut dépasser les capacités de la tablette.

Le bitmap est ensuite transformé en objet Drawable, plus exactement DrawZoneView qui étend Drawable. Cet objet forme une couche qui est ensuite apposée à l'ImageView décrite dans les fichiers de layout.

4.6.2. Gestion des évènements

Les différents évènements gérés sont les clics sur les boutons, les appuis courts et longs, les appuis déplacés, sur l'image. Tous ces évènements sont tout d'abord déclarés à l'aide de différents listeners. En fonction des évènements les appuis sur les différents boutons sont rendus possibles ou impossibles. Si Android gère les clics longs sur des boutons ce n'est pas le cas pour les touches longs, il revient donc à l'application de mesurer le temps d'appui. Les touches d'images renvoient un objet de type MotionEvent permettant de déterminer si l'évènement a consisté en un appui ou un relâcher, un déplacement, de combien, et quand, entre autres. Les évènements sont gérés en fonction des cas généraux : sélection d'une image, création de zone, édition de zone, et point sélectionné. Les touches effectués hors de l'image sont projetés sur celle-ci.

4.6.3. Procédure d'affichage

Lors des différents événements l'affichage est rafraîchi, ce qui correspond à plusieurs opérations : gérer la disponibilité des différents boutons, le passage d'un cas (sélection, création, édition de zones) à un autre, et appeler la procédure d'affichage de DrawZoneView en "invalidant" l'affichage de l'image. Celui se fait en fonction de références à une liste de zones, à une zone de travail et un point sélectionné, références qu'il ne faut pas écraser. Utiliser des setters pour les modifier.

La procédure draw consiste à une définition des différentes peintures utilisées, couleur, trait ou remplissage, largeur, style du trait... Les largeurs de traits et de points sont définies pour une image de taille donnée et sont réactualisées en fonction de la taille occupée par l'image. Vient ensuite l'affichage en lui-même : il varie en fonction du cas d'utilisation, du nombre de points, d'intersections éventuelles.

L'affichage consiste à afficher des points, des segments entre des points, des différentes listes de points composant la zone - sachant que les intersections sont également décrites par des listes de points ; seule la peinture change.

4.6.4. Annulation des actions effectuées

La zone en cours d'édition est référencée en cache pour permettre une annulation groupée des modifications.

Une classe Action, interne à la classe Zone, permet d'enregistrer les actions. Lors de l'ajout ou de la suppression d'un point celui-ci est enregistrée dans une action ; lors du déplacement d'un point l'action est initialisée et clôturée "manuellement" dans ZoneFragment.java, les points de début et de fin étant enregistrés dans l'Action.

Lors de l'appui sur Annuler dernière action l'application appelle la classe Zone pour qu'elle supprime la dernière action de sa liste, en annulant ses conséquences (insertion d'un point supprimé à son ancienne location, suppression d'un point ajouté, remplacement d'un nouveau point par un ancien).

4.6.5. Validation des modifications

Une zone ne peut être validée que si son polygone a au minimum 3 points et si le polygone n'est pas croisé. Cette validation est possible de deux manières différentes, par bouclage "intuitif" en touchant le premier point (uniquement lors de la création d'une zone) ou appui sur le bouton dédié. Lors de la validation la zone de cache et la liste d'actions, conservée pour pouvoir effectuer des annulations simplement sont supprimées.

4.6.6. Gestion des intersections entre zones

Lors de la validation, l'application gère l'intersection entre les zones, en proposant à l'utilisateur une séparation des parties recoupées ou une superposition des zones. Dans le cas d'un découpage une zone peut se retrouver avec des trous, ce qui la conduit à avoir plusieurs listes de points.

Dans le cas de l'ajout de zone sans intersection avec les précédentes zones, l'application permet de sélectionner une zone recouverte en sélectionnant en priorité la zone avec la surface la plus faible.

4.6.7. Gestion des polygones croisés

L'application vérifie que les polygones ne sont pas croisés, et empêche la validation dans le cas contraire. La vérification est effectuée liste de points par liste de points par la méthode `isSelfIntersecting`, qui vérifie par groupe de 2 segments, c'est à dire de 4 points, s'il y a intersection.

L'application vérifie chaque contour du polygone, il n'y en a qu'un si le `PixelGeom` est simple, mais si le `PixelGeom` est composé de plusieurs polygones comportant eux-mêmes plusieurs trous, il faut vérifier chaque contour indépendamment.

4.7. Fragment Characteristics

4.7.1. Définition des caractéristiques d'un élément

Pour chaque *Element*, on peut définir :

- Un type parmi une liste de type définie sur la base de données distante dans la table *ElementType*. L'application ne permet pas l'ajout direct de nouveaux types, pour cela il faut demander à l'administrateur de la base de données de le rajouter dans la base. Cette information est enregistrée dans la table *Element* grâce à une clé étrangère qui pointe sur l'*ElementType* choisi.
- Un matériau parmi une liste définie de la même façon sur la table *Material*.
- Une couleur définie avec la boîte de dialogue **Android Color Picker**. Cette information est enregistrée directement sous forme d'une chaîne de caractère représentant un entier dans colonne *element_color* de la table *Element*.

4.7.2. Regroupement de PixelGeom

Ce fragment permet également de regrouper différents *Element*. On peut le faire de deux façons :

- En utilisant l'attribut *linkedPixelGeom* qui indique pour chaque *PixelGeom* les *PixelGeom* qui lui sont liés, ainsi lorsque l'on clique sur un *PixelGeom*, on parcourt la liste de tous les *PixelGeom* pour tous les sélectionner (ou désélectionner). Cette opération est réversible en ré-initialisant la liste avec une liste vide.
- En fusionnant les *Geometries* représentée par les *PixelGeoms* en une seule *Geometry* représenté par un *PixelGeom*. Cette opération n'est pas réversible et est effectué grâce à la bibliothèque JTS.

4.8. Fragment Synchronization

Les données traitées par l'application (soit celles « intéressantes » pour la photo ou le projet sélectionné) existent à trois endroits différents :

- Dans la base de données locale, donc stockées dans l'application
- Dans des variables statiques de l'application, instanciées à chaque utilisation de l'application
- Dans la base de données externe, située sur le serveur ser-info-02

Lors du clic sur le bouton Sauvegarde, la série d'opérations suivantes est initiée.

Dans un premier temps l'application récupère les « maxID » de la base de données distante. Il s'agit des ID les plus grands pour chaque table, en effet la base distante est maître pour le

choix des IDs, si conflit il y a c'est sa valeur qui sera imposée. Les IDs récupérés sont donc utilisés en cas de conflit, ou de création d'un nouveau projet.

Les IDs sont donc enregistrés dans les variables statiques de l'application, puis les variables statiques sont enregistrées dans la base de données locale.

La synchronisation vers la base de données externe commence en même temps (thread) que l'enregistrement local. On commence par traiter les données pour les mettre au format JSON. La chaîne de caractères en JSON est alors transmise en POST à un script PHP, qui l'interprète et effectue les opérations correspondantes (ajout ou mise à jour des informations) en SQL sur la base de données. Le serveur retourne alors un message sur l'état de l'opération à l'application.

En parallèle de l'envoi de la chaîne de caractères au format JSON vers le serveur, l'envoi de la photo est fait, dans un thread différent. Les photos ne sont pas stockées dans la base de données mais dans un dossier séparé sur le serveur. Seules les adresses permettant d'y accéder apparaissent dans la base de données. La table Photo contient un champ pour la date de dernière modification de la photo. Ce champ est utilisé pour la gestion des conflits en écriture sur un projet.

La gestion des conflits se résume à la fonctionnalité suivante : à l'ouverture de l'application, en même temps que la récupération des IDs maximaux, on récupère la date actuelle du serveur (le référent en termes de temps). Au moment d'exporter les modifications vers la base de données distante, on compare la date de dernière modification à celle récupérée au début, si elles diffèrent, un message est affiché, avertissant l'utilisateur que des modifications ont été effectuées et qu'il risque de les écraser en enregistrant.

5. Identifiant de connexion à la base de données

Voici les identifiant de connexion à la base de données :

URL : urbapp.ser-info-02.ec-nantes.fr

Port : 5432

Dbname : urbapp

User : urbapp

Password : zehirmann