SmartCity

par

Tano Iannetta

Loan Lassalle

Luana Martelli

Wojciech Myszkorowski

Camilo Pineda

Jérémie Zanone

sous la direction du Professeur René Rentsch

Semestre de Printemps 2017

# Table des matières

[Table des matières 2](#_Toc483824081)

[1 Introduction 4](#_Toc483824082)

[1.1 But 4](#_Toc483824083)

[1.2 Objectifs 4](#_Toc483824084)

[1.3 Description 4](#_Toc483824085)

[1.4 Organisation 5](#_Toc483824086)

[1.5 Répartition des tâches 5](#_Toc483824087)

[2 Conception – Architecture 6](#_Toc483824088)

[2.1 Systèmes d’exploitation et outils logiciels 6](#_Toc483824089)

[2.2 Fonctionnalités principales 7](#_Toc483824090)

[2.2.1 Cas d’utilisations 7](#_Toc483824091)

[2.2.2 Gestion des requêtes effectuées à la base de données 8](#_Toc483824092)

[2.2.3 Gestion d'un agenda 8](#_Toc483824093)

[2.2.4 Gestion des requêtes faites par les utilisateurs 8](#_Toc483824094)

[2.2.5 Filtrage des requêtes 8](#_Toc483824095)

[2.2.6 Ajout d'évènements de la ville 8](#_Toc483824096)

[2.2.7 Implémentation d'une carte interactive 8](#_Toc483824097)

[2.2.8 Génération de documents PDF 9](#_Toc483824098)

[2.3 Technologies utilisées 9](#_Toc483824099)

[2.3.1 Java 9](#_Toc483824100)

[2.3.2 Git – GitHub 10](#_Toc483824101)

[2.3.3 Apache Maven 11](#_Toc483824102)

[2.3.4 Carte interactive 12](#_Toc483824103)

[2.3.5 Base de données 15](#_Toc483824104)

[2.3.6 Génération de documents PDF 17](#_Toc483824105)

[2.3.7 Interface graphique utilisateur (GUI) 18](#_Toc483824106)

[3 Implémentation 19](#_Toc483824107)

[3.1 Carte interactive 19](#_Toc483824108)

[3.1.1 Tuile 19](#_Toc483824109)

[3.1.2 Fournisseur de tuiles 20](#_Toc483824110)

[3.1.3 Interface graphique utilisateur 22](#_Toc483824111)

[3.2 Base de données 23](#_Toc483824112)

[3.2.1 Modèle de données relationnel 25](#_Toc483824113)

[3.2.2 Diagramme des classes 26](#_Toc483824114)

[3.3 Génération de documents PDF 27](#_Toc483824115)

[3.4 Interface graphique utilisateur (GUI) 28](#_Toc483824116)

[4 Tests de l’application 32](#_Toc483824117)

[4.1 Description générale de l’environnement de test 32](#_Toc483824118)

[4.1.1 Matériel 32](#_Toc483824119)

[4.1.2 Systèmes d’exploitation et outils logiciels 32](#_Toc483824120)

[4.2 Tests unitaires 33](#_Toc483824121)

[4.2.1 Carte interactive 33](#_Toc483824122)

[4.2.2 Base de données 34](#_Toc483824123)

[4.2.3 Génération de documents PDF 35](#_Toc483824124)

[4.3 Tests d’intégrations 36](#_Toc483824125)

[4.4 Tests des cas d’utilisation 36](#_Toc483824126)

[5 Conclusion 37](#_Toc483824127)

[5.2 Problèmes rencontrés 37](#_Toc483824128)

[5.3 Propositions d’améliorations 38](#_Toc483824129)

[5.3.1 Interface graphique de l’utilisateur 38](#_Toc483824130)

[5.3.2 Formulaire des évènements 38](#_Toc483824131)

[5.3.3 Génération de documents PDF 38](#_Toc483824132)

[6 Annexes 39](#_Toc483824133)

[6.1 Cahier des charges 39](#_Toc483824134)

[6.2 Journaux de travail 40](#_Toc483824135)

[6.3 Planification initiale et son évolution 46](#_Toc483824136)

[6.4 Rapport de tests 46](#_Toc483824137)

[7 Sources – Bibliographies 47](#_Toc483824138)

[7.1 Ouvrages 47](#_Toc483824139)

[7.2 Sites Internet 47](#_Toc483824140)

[7.3 Vidéos en ligne 47](#_Toc483824141)

[8 Glossaire 48](#_Toc483824142)

# Introduction

## But

Aujourd’hui, la plupart des citoyens ont la possibilité de communiquer très rapidement avec les services privés ou publics. Cette proximité leur permet de partager leurs avis et conseils afin d’améliorer les services qui leur sont fournis par la ville de Lausanne.

Notre projet a consisté à l’élaboration d’une application nommée « SmartCity » qui assure l’archivage et la gestion de requêtes de différents acteurs de la société actuelle, l’affichage du lieu correspondant aux requêtes et la production de documents PDF des données sélectionnées. Cette réalisation s’est faite dans le cadre du module « Projet » de la Haute École d’Ingénierie et de Gestion du Canton de Vaud (HEIG-VD).

Ce programme permet :

* de gérer les requêtes venant des citoyens, de l’administration de la ville de Lausanne ou de « personnes de confiance »
* de localiser de manière claire et précise le lieu correspondant à une requête
* d’obtenir un résumé des différentes informations archivées

Dans le cadre de ce rapport, nous exposerons les technologies qui nous ont permis de réaliser l’application. Nous décrirons les différentes fonctionnalités implémentées afin de mieux comprendre l’utilité de notre application. Les problèmes rencontrés ainsi que les changements de conception seront également abordés afin d’expliquer nos choix quant au produit final.

Il est à noter que nous réalisons la partie administrative de l'application, c'est-à-dire que notre projet sera utilisé par la ville et non par les citoyens de la ville.

## Objectifs

* Organiser les requêtes venant des différents acteurs identifiés (citoyens, administration de la ville de Lausanne, « personnes de confiance »).
* Afficher les lieux correspondants aux requêtes sur une carte interactive.
* Produire des documents PDF pour fournir un résumé des informations archivées.

## Description

Après la présentation du premier projet, nous avons décidé de créer une application permettant à l’administration d’une ville (dans notre cas Lausanne) de gérer les requêtes venant de ses concitoyens, des « personnes de confiance » ou de ses propres services.

L’application s’articule autour de quatre actions principales.

La première action consiste à la consultation d’évènements ou de requêtes recueillis au sein d’une base de données. Ces évènements peuvent être transmis par les citoyens au travers d’un site Internet ou d’une application mobile (non réalisés dans le cadre du projet). Des entreprises ou des personnes bénéficiant de la confiance de l’administration de la ville obtiennent le statut de « personnes de confiance ». Elles peuvent, sans l’acceptation de l’administrateur, ajouter des évènements à la base de données. Ce droit est aussi accordé à l’administrateur de l’application.

La deuxième action est caractérisée par la gestion des requêtes stockées au sein de la base de données. Chaque requête créée par un citoyen lambda doit être validée par l’administrateur. Une fois validée, il est possible de consulter la requête au travers de différentes rubriques et de visualiser, au moyen d'une carte interactive, les lieux concernés. Pour une meilleure visibilité, chaque rubrique est associée à un filtre, rendant plus aisée la lecture sur la carte.

La carte interactive permet à l’administrateur de se déplacer et d’agrandir le plan de la ville au besoin. La sélection d’une date lui offre le moyen d’affiner ces critères d’affichage. En ajoutant une priorité aux évènements, l’administrateur se verra notifier des requêtes possédant la priorité la plus haute dans le but de traiter le plus rapidement celles-ci.

La dernière et principale action de l’application est de générer des documents PDF contenant des informations relatives aux évènements et aux rubriques que l’administrateur aura précédemment choisis. Cela permettra d’obtenir une trace écrite pour un meilleur partage d’informations au sein de l’administration de la ville de Lausanne.

## Organisation

Chef de projet :

* Tano Iannetta : [tano.iannetta@heig-vd.ch](mailto:tano.iannetta@heig-vd.ch)

Chef en second :

* Wojciech Myszkorowski : [wojciech.myszkorowski@heig-vd.ch](mailto:wojciech.myszkorowski@heig-vd.ch)

Membres du projet:

* Loan Lassalle : [loan.lassalle@cpnv.ch](mailto:loan.lassalle@cpnv.ch)
* Luana Martelli : [luana.martelli@heig-vd.ch](mailto:luana.martelli@heig-vd.ch)
* Camilo Pineda : [camilo.pineda@heig-vd.ch](mailto:camilo.pineda@heig-vd.ch)
* Jérémie Zanone : [jeremie.zanone@heig-vd.ch](mailto:jeremie.zanone@heig-vd.ch)

## Répartition des tâches

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Phases/Membres | Tano Iannetta | Wojciech Myszkorowski | Loan Lassalle | Luana Martelli | Camilo  Pineda | Jérémie Zanone |
| Dossier final | 15% | 20% | 20% | 10% | 25% | 15% |
| Analyse | 10% | 10% | 15% | 15% | 10% | 10% |
| Conception | 10% | 10% | 10% | 15% | 15% | 15% |
| Réalisation | 30% | 20% | 30% | 30% | 10% | 30% |
| Administration | 10% | 10% | 5% | 5% | 10% | 5% |
| Tests | 10% | 20% | 10% | 10% | 10% | 15% |
| Documentation de code | 15% | 10% | 10% | 15% | 15% | 10% |

# Conception – Architecture

## Systèmes d’exploitation et outils logiciels

Lors du choix des outils de conception, il était impératif de sélectionner des logiciels multiplateformes. Cette condition devait être remplie pour la bonne raison que les membres du projet ne possèdent pas le même système d’exploitation. La plupart des outils utilisés possèdent une licence qui a été contractée personnellement ou au travers des programmes de licences fournis par l’HEIG-VD.

* Microsoft Windows 10 Familiale 64 bits
* Microsoft Windows 10 Education 64 bits
* GNU/Linux Linux Mint 64 bits
* IntelliJ IDEA Ultimate 64 bits
* Java SE Development Kit 8
* Java SE Runtime Environment 8
* Eclipse Neon 2 64 bits
* Windows Builder 4.6
* Microsoft Office Professional Plus 2016 64 bits
* MySQL Workbench 6.3
* Inkscape 0.48.5 64 bits
* StarUML 2.8.0

## Fonctionnalités principales

### Cas d’utilisations

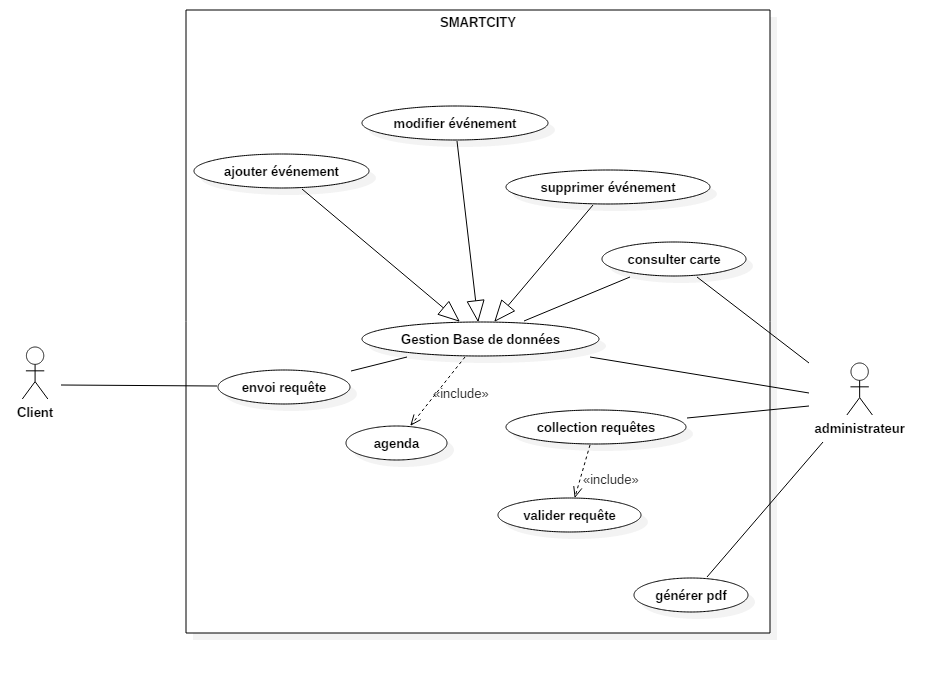


Figure 1 : Diagramme des cas d’utilisations

L’administrateur est la personne qui utilisera l’applicatif. Cette personne a la possibilité d’effectuer plusieurs actions.

Elle peut tout d’abord valider des requêtes qui figurent dans une base de données. L’administrateur peut rajouter des évènements dans la base de données et modifier ceux qui y figurent. Chaque évènement a une date de début et de fin, ce qui peut servir d’affichage. Une fois les requêtes validées, il est possible de les consulter sur la carte interactive.

Finalement, l'administrateur peut générer des documents PDF qui contiennent les informations selon les rubriques d'évènements sélectionnés.

Dans notre cas, le client ne peut pas envoyer de requête, la base de données contient un certain nombre d’évènements en attente de validation pour simuler ce cas d'utilisation.

### Gestion des requêtes effectuées à la base de données

Différents types de requêtes sont exécutées par la base de données. L’ajout, la modification et la suppression d’évènements sont des requêtes produites par l’administrateur. Il est le garant du bon fonctionnement de la base de données et le gérant des requêtes en attente des citoyens.

Lorsqu’un citoyen fait une proposition d’évènement, l’administrateur peut choisir de la valider ou de la refuser. Une fois valider, l’évènement sera visible sur la carte interactive suivant ces dates de début et de fin. Si toutefois, l’évènement est refusé, la base de données lui affecte un statut précis. Il ne sera plus visible au sein de l’application.

### Gestion d'un agenda

Chaque requête a une date de début et une date de fin. L’organisation de la carte interactive selon des rubriques et de la date (granularité au jour principalement).

### Gestion des requêtes faites par les utilisateurs

Les utilisateurs de l'application peuvent effectuer plusieurs types de requêtes, classées selon les rubriques suivantes :

* Accidents
* Travaux
* Constructions
* Rénovations
* Manifestations
* Doléances

### Filtrage des requêtes

L'administrateur s’occupe de filtrer les requêtes faites par les utilisateurs. Pour certains comptes privilégiés, il n'y aura pas besoin de l'intervention de l'administrateur.

* Gestion des évènements (acceptation de requêtes, refus)

### Ajout d'évènements de la ville

Les utilisateurs peuvent émettre une localisation d'une nouvelle manifestation qui sera approuvée ou non par l’administrateur.

### Implémentation d'une carte interactive

L'application comporte une carte avec laquelle il est possible d'interagir. Cette carte référence, à l'aide de « pins », la localisation des évènements dans la ville de Lausanne. Ces évènements sont filtrés par leur date de début, leur date de fin et leur rubrique.

### Génération de documents PDF

La génération de documents PDF est réalisée selon une rubrique d’évènement choisie.

Ces documents PDF sont destinés aux départements correspondant aux rubriques. Ils permettent de garder un historique détaillé des évènements au sein de la ville. On peut imaginer qu'une fois générés, ils sont envoyés aux personnes concernées.

## Technologies utilisées

### Java

Java est un langage de programmation informatique orienté objet, développé par Sun Microsystems. Nous avons décidé d’utiliser ce langage pour différentes raisons.

La première est que nous l’avons appris récemment en cours. Ce langage était encore frais dans nos esprits et son usage a l'avantage d'imposer une manière de coder claire mais non pas laborieuse. De plus, Javadoc permet de réaliser une documentation claire et complète de codes.

Java permet aussi d'avoir un très haut niveau d'abstraction à la machine. Sa portabilité contribue à l’exécution de programmes sur tout type de machine. Ainsi, elle offre une plus grande liberté aux utilisateurs.

En ce qui concerne la sécurité, Java est considéré comme un langage fiable, stable, dû à l’utilisation d’une machine virtuelle Java. Un programme ne peut en aucun cas compromettre l’intégrité de cette dernière.

Par ailleurs, ce langage permet l’exécution « simultanée » de plusieurs tâches pour obtenir de meilleures performances. Lors de l’utilisation d’interfaces graphiques, il est indispensable d’utiliser ce mécanisme.

Pour terminer, les deux principales raisons de notre choix ont été, la disponibilité d’un grand nombre de librairies tierces et les améliorations constantes apportées à Java.

### Git – GitHub

Pour tout projet, il est primordial d’utiliser un système de versions de fichiers pour récupérer ou archiver des documents. Ceci permet de suivre l’évolution des fichiers d’un projet, d’administrer les versions successives de plusieurs documents et de maintenir à jour une copie de ces fichiers à différents endroits ou avec les différents membres du projet.

Dans ce projet, nous avons utilisé Git pour sa rapidité, sa taille extrêmement réduite, sa gestion des branches et sa décentralisation. Multiplateforme, il a permis à chacun de travailler dans son environnement préféré.



Figure : Les différentes « zones » de Git et les commandes correspondantes

En complément, nous avons utilisé la plateforme GitHub pour héberger et gérer le développement de logiciels. Elle nous a permis de mettre facilement et efficacement le travail de chacun en commun, d’obtenir une vue globale de l’évolution du projet et d’avoir un échange d’informations plus rapide et à intervalle régulier.

L’atout majeur de GitHub est la possibilité d’accéder à une ressource en même temps qu’une autre personne. Lorsque deux personnes ou plus travaillent sur la même ressource en même temps et décident d’envoyer au serveur distant, les modifications effectuées vont lever un conflit lors de la fusion des fichiers du projet. À ce moment, il faudra décider quelles sont les parties à garder de celles qui ont été modifiées.

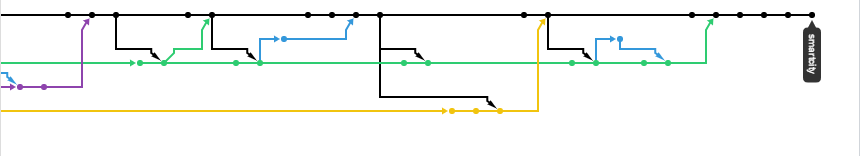


Figure : Les branches du projet par rapport à une date de départ

### Apache Maven

Nous avons utilisé Maven pour la conception architecturale de notre projet.

Maven est un outil pour la gestion et l’automatisation de production Java en général et Java EE en particulier. L’objectif est de produire un logiciel à partir des sources du programme, en optimisant les tâches réalisées et en garantissant le bon ordre de fabrication.

Maven utilise un paradigme connu sous le nom de Project Object Model (POM) qui décrit un projet logiciel, ses dépendances avec des modules externes et l'ordre à suivre pour sa production. Il est livré avec un grand nombre de tâches prédéfinies, comme la compilation de code Java ou encore la modularisation. Nous y trouvons entre autres :

* Le nom du projet
* Le numéro de version
* Les dossiers contenant le code source et les fichiers de ressources
* Les bibliothèques nécessaires à la compilation
* Les dépendances vers d’autres projets
* Les noms des contributeurs

Maven 2 se concentre sur le principe de favoriser l'utilisation des conventions plutôt que de configurer son projet. Ainsi, si l'on respecte certaines conventions définies par Maven 2, il devient inutile de préciser certaines informations dans son pom.xml. Par exemple, Maven 2 préconise l'utilisation du répertoire src/main/java pour stocker les fichiers sources du projet. En respectant ceci, il devient alors inutile de spécifier à Maven 2 où se trouvent les sources Java, ce qui allège d'autant l'écriture du pom.xml.

Avec ce simple pom.xml, il devient possible de réaliser tout le processus de construction d'un projet avec Maven 2, pour :

* Gérer les fichiers de ressources
* Compiler les sources Java
* Compiler et exécuter les tests unitaires
* Créer le fichier JAR du projet
* Déployer le JAR

### Carte interactive

Dans notre projet, nous avons besoin d’une carte interactive qui permet de visualiser avec précision où ont lieu les évènements. Cependant, cette carte n’utilise pas l’adresse des évènements comme référence.

Pour représenter un point géographique, nous avons utilisé deux systèmes de coordonnées :

* Un système de coordonnées sphériques nommé WGS84, qui utilise la latitude et la longitude. Ce système est employé par défaut sur la majorité des GPS.
* Un système de coordonnées cartésiennes, utilisé par les principaux outils de géolocalisation en ligne comme OpenStreetMap.

#### Système WGS84

Système géodésique mondial le plus courant, il permet de géolocaliser des positions sur un plan ou une carte à partir de ses coordonnées géographiques. Nous avons décidé de l’utiliser pour localiser les différents évènements sur une carte.

Chaque localisation a une représentation tridimensionnelle :

* **La longitude** et **la latitude** sont des coordonnées géographiques représentées par des valeurs angulaires, expression du positionnement est-ouest, respectivement nord-sud d'un point sur Terre.
* **L’altitude** est l'élévation verticale d'un lieu ou d'un objet par rapport à un niveau de base.

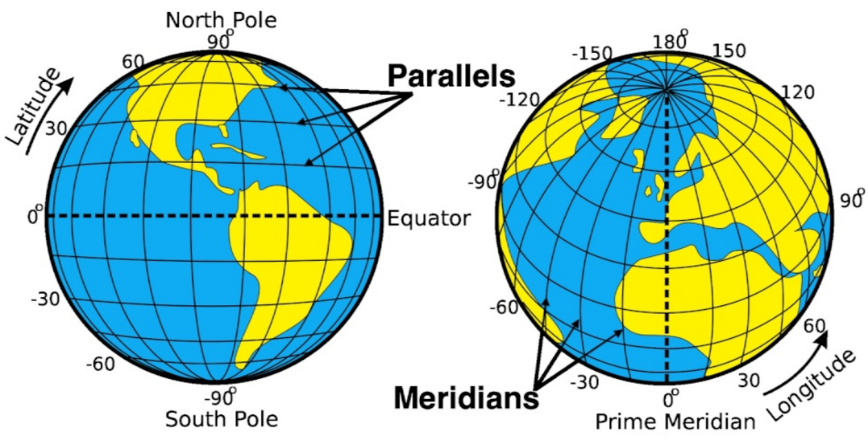


Figure : Représentation du système WGS84

Le méridien de référence est celui de Greenwich. La longitude est comprise entre -180o et + 180o : les valeurs positives se trouvent à l’est et les valeurs négatives à l’ouest. La latitude se situe entre -90o et + 90o, les valeurs positives se trouvant au nord et les valeurs négatives au sud de l’équateur.

Dans ce projet, nous n’avons pas tenu compte de l’altitude puisque c’est une donnée qui ne nous était pas utile. Seules nous importaient les informations concernant la position au sol.

Ce système comporte plusieurs représentations des coordonnées. Le système le plus connu étant les degrés minutes et secondes (DMS). L’API de Google Maps et d’OpenStreetMap utilise notamment la représentation en degrés décimaux (DD). Nous l’avons choisi pour des questions de simplification.

#### Système cartésien

Le système cartésien dans le monde cartographique permet de localiser un endroit sur une image d’une carte du monde, sous la forme d’une projection de Mercator qui projette les pôles à l’infini. Cette image est définie par une échelle ou zoom grâce aux axes x et y dont l’unité est le pixel. Si l’échelle change alors les coordonnées également.

(0,0)

(256,0)

(0,256)

axe x

axe y



Figure : Exemple d’un système cartésien au niveau de zoom 0

Le zoom 0 correspond au zoom le plus éloigné et nous permet de voir la carte en entier, dans une image carrée de 256 pixels de côté. À l’inverse, le zoom 20 permet d’obtenir une image détaillée et centrer sur une zone très précise. Cependant, les dimensions de l’image augmentent, puisqu’à chaque niveau de zoom la carte est deux fois plus grande que l’image précédente.

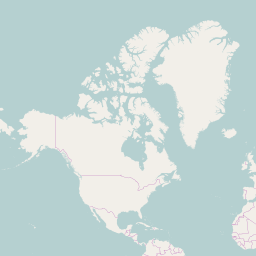
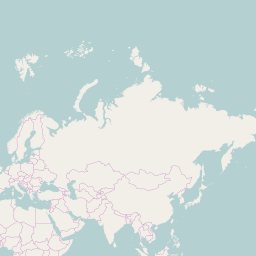
#### Conversion du système WGS84 en système cartésien

Pour passer d’un système tridimensionnel à un système bidimensionnel, nous utilisons une projection cartographique. Il existe une variété de projections qui permettent de passer de l’un à l’autre. Nous avons utilisé la projection de Mercator, utilisée par les services cartographiques tels que Google Maps et OpenStreetMap. Cette projection désigne la terre comme une sphère parfaite ce qui n’est pas le cas dans la réalité. Elle fait correspondre le point de coordonnées WGS84 (0°,0°) avec le centre de la carte.

#### Carte en tuiles

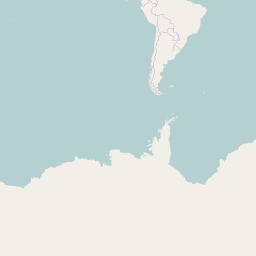
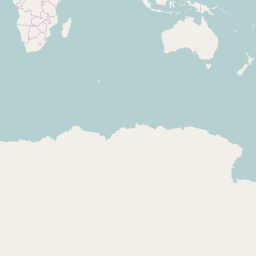
Dans ce projet, nous utilisons les services cartographiques proposés par OpenStreetMap qui met à disposition des serveurs fournissant des images carrées, appelées tuiles de 256 pixels (cf. Système cartésien). Elles représentent des parties de la carte du monde à un zoom défini.

Ce petit format est très utile, car il demande peu de ressources pour son utilisation. Si le format des fichiers était plus conséquent, le transfert via le réseau Internet serait limité par la bande passante. De plus, ce format contient la quantité d’information nécessaire à nos besoins. Les tuiles de petite taille sont plus pertinentes pour notre utilisation. Mises côte à côte, elles forment l’image de la section désirée.

(0,0)

(1,0)

(1,1)

(0,1)

Figure : Utilisation des tuiles de zoom de niveau 1

Les tuiles sont caractérisées par 3 paramètres : le niveau de zoom et les coordonnées x et y. L’accès au serveur s’effectue à l’aide de l’URL : <http://a.tile.openstreetmap.org/17/67960/46219.png>. Les trois derniers numéros de l’adresse Internet représentent les paramètres cités précédemment.

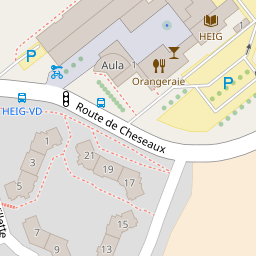


Figure : Tuile représentant l’HEIG-VD sur le site de Cheseaux au niveau de zoom 17

L’utilisation de ces images est gratuite selon des conditions à respecter. Dans le cadre de notre projet, qui n’est pas destiné à la commercialisation, l’utilisation des images est autorisée.

### Base de données

L’utilisation d’une base de données est indispensable pour l’enregistrement, l’organisation et le tri d’une importante quantité de données. Bien structurée et indexée, une base de données doit répondre aux principales formes normales existantes. La normalisation des modèles de données permet ainsi de vérifier la robustesse de sa conception pour améliorer la modélisation et faciliter la mémorisation des données en évitant, par exemple, les problèmes de redondances, de mise à jour ou de cohérence.

L’accès à une base de données peut être entrepris à travers le réseau ou directement sur la machine hôte. L’application SmartCity accèdera à la base de données située en local. La création et l’utilisation d’une base de données au sein du langage Java seront facilitées par l’utilisation du framework Hibernate.

Il sera nécessaire à l’utilisateur d’installer un service tiers tel que MySQL pour fournir un gestionnaire de bases de données relationnelles. Pour ce qui est de la structure de la base de données, nous avons décidé d’utiliser la technologie de MySQL, car nous étions habitués à l’utiliser.

#### Modèle de données entité-association

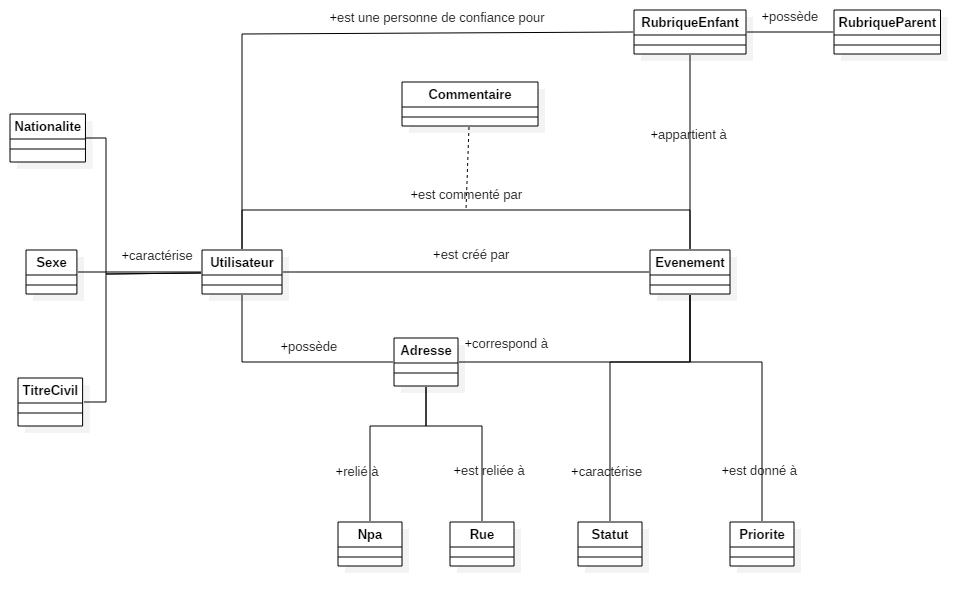


Figure : Modèle de données entité-association

#### MySQL

MySQL est un système de gestion de bases de données relationnelles (SGBDR) qui nous a permis de créer et de mettre en place la base de données. Pour plus de simplicité, le logiciel MySQL Workbench a été utilisé.

Pour pouvoir utiliser l’application, il a fallu insérer des données. Celles-ci bien que fictives ont été insérées à travers un script. Cela permet de proposer une application prête à l’emploi.

#### Hibernate

Pour faciliter la gestion des données au sein de l’application, nous avons utilisé le concept d’objet persistant, permettant de stocker des objets Java en toute simplicité. Pour ceci, nous avons utilisé le framework Hibernate.

Ce dernier apporte le moyen de traduire les objets Java en données stockables dans la base. Il a fallu créer toutes les tables de la base de données en tant que classe Java et définir des fichiers de mapping pour que Hibernate fasse correspondre le code Java et la structure de la base de données.



Figure : Logo du framework Hibernate

### Génération de documents PDF

Les évènements sont stockés dans la base de données. Cependant, il est utile de pouvoir retrouver ces informations sous un autre forme. Nous avons décidé de les archiver dans des documents PDF. Ainsi, les utilisateurs de l'application retrouvent plus facilement les informations souhaitées.

De plus, chaque document contient des statistiques qui permettent d’obtenir une représentation visuelle des données.

Pour générer le document PDF, nous avons choisi d’utiliser le kit de développement iText version 7. Ce support offre plusieurs librairies qui permettent de créer facilement des documents PDF et laisse un grand degré de liberté sur la gestion de la mise en page.



Figure : Logo du kit de développement iText

Pour la partie graphique, nous avons utilisé JfreeChart. Ce kit permet de créer toutes sortes de graphiques. Dans le cadre du projet, nous avons choisi deux graphes en particulier, le diagramme en barres et le diagramme circulaire.



Figure : Logo du kit de développement JFreeChart

### Interface graphique utilisateur (GUI)

Pour l'interface graphique, nous avons décidé d'utiliser la technologie Swing.

Nous avions déjà travaillé avec cette librairie par le passé lors du semestre précédent. Comme nous avions peu travaillé avec des interfaces graphiques, nous avons préféré utiliser cette libraire connue, car notre application nécessitait une interface graphique conséquente.

De plus, Swing propose tous les composants nécessaires à la réalisation de notre projet. Les composants principaux étant des panels, des boutons, des champs de saisie, des boites de sélection et un calendrier.

En revanche, le design de Swing n'est pas des plus épurés, mais nous avons ciblé les fonctionnalités plutôt que l'esthétique pour ce projet.

Swing impose une syntaxe de code lourde et sa relecture en est plus laborieuse qu'à l'accoutumée. Cependant, nous avons fait en sorte de séparer les éléments graphiques du reste afin d'avoir un code structuré.

En évaluant ses points positifs et négatifs, nous avons donc fait le choix d'utiliser cette libraire pour implémenter l'interface graphique de notre application.

# Implémentation

Deux choix importants ont été faits lors de l’implémentation de l’application. Le premier concerne l’affichage de message en différentes langues dû au plurilinguisme de la Suisse.

Étant donné qu’au niveau fédéral les langues officielles sont l’allemand, le français et l’italien. Nous avons souhaité mettre en place la gestion de messages en plusieurs langues afin de faciliter l’utilisation de notre application dans d’autres villes que Lausanne. Nous avons utilisé des fichiers ResourceBundle.properties pour adapter la langue affichée au système exécutant l’application.

Le second fut porté sur le suivi des erreurs survenues au sein de notre application. Nous avons décidé de mettre en place un système de journalisation qui permet de suivre en temps réel ou après coup les différentes erreurs survenues. Ces fichiers de log sont enregistrés dans le dossier SmartCity/Logs.

## Carte interactive

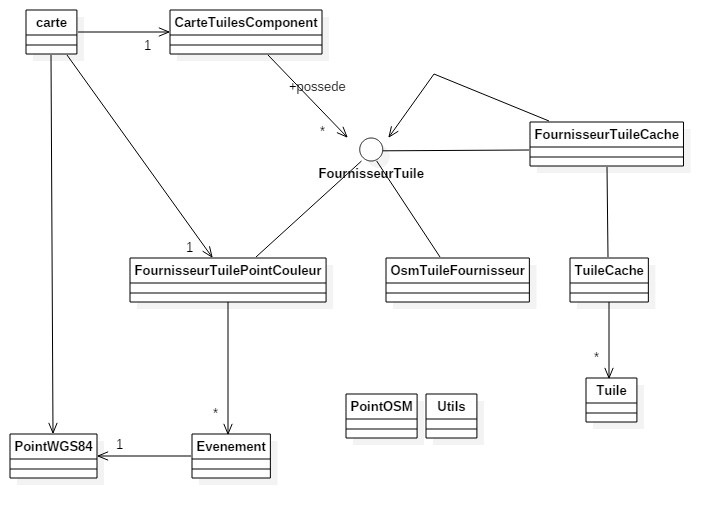


Figure : Schéma de la structure des classes utilisée pour construire la carte interactive

### Tuile

Les tuiles sont des images représentées par des instances de la classe BufferedImage de la bibliothèque Java. Les attributs de cette classe sont les paramètres de la tuile soit le niveau de zoom et les coordonnées x et y (cf. Carte en tuiles).

### Fournisseur de tuiles

Un fournisseur de tuile fournit une tuile selon ses coordonnées et son niveau de zoom. Ils implémentent l’interface FournisseurTuile qui possède une seule méthode qui permet de fournir une tuile selon les paramètres donnés (zoom et coordonnées).

#### OSM

Ce fournisseur obtient les tuiles à partir d’un serveur distant appartenant à OpenStreetMap. Son attribut est l’URL de base du serveur (cf. Carte en tuiles). L’image est récupérée grâce à une instance de la classe URL et la méthode statique read de la classe ImageIO. Un objet Tuile est alors créé. En cas d’erreur de connexion ou de faiblesse du serveur, une tuile alternative est affichée.



Figure : Tuile d’erreur

#### Point couleur

Le but principal de la carte est de pouvoir afficher les différents évènements en cours aux bons endroits.

Les tuiles sont des images, il est donc aisé d’en faire des modifications. Cependant, le procédé de modification est lourd, car à chaque changement du filtre d’évènements il faut reconstruire des tuiles OSM entièrement.

À ces fins, nous avons créé un autre fournisseur de tuiles (fournisseurTuilePointCouleur) qui possède en attribut une liste des évènements récupérés à partir de la base de données. Donc, lors d’un accès à une tuile, une nouvelle image est créée avec le nom et l’ID de l’évènement : ils apparaissent directement sur l’emplacement de celui-ci.

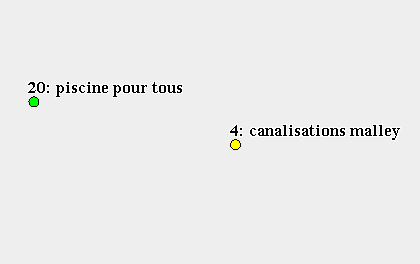


Figure : Exemple d’une tuile fournie depuis le fournisseurTuilePointCouleur

Par la suite, les tuiles marquées de points de couleur vont se superposer aux tuiles de la carte, grâce à un fond transparent. Lors d’un changement d’affichage, seules ces tuiles sont modifiées.

#### Cache

Le fournisseur de tuile en cache est un transformateur qui enveloppe un fournisseur de tuile standard. Tous les nouveaux appels à une tuile sont stockés dans un objet TuileCache qui est une table associative de tuiles, LinkedHashMap.

La clé pour retrouver la tuile est un objet de type Long et construit à partir des coordonnées de la tuile. La classe LinkedHashMap offre la méthode removeEldestEntry qui permet de fixer une taille maximum de données, et une fois atteinte, s'assure que les plus anciennes sont effacées.

Si la tuile s’y trouve déjà, il n’y a pas besoin de la reconstruire. Un simple appel à la table associative est suffisant. Ce procédé n’améliore pas la rapidité lors du premier chargement, mais apporte de la fluidité une fois que les tuiles ont été chargées dans le cache.

### Interface graphique utilisateur

La classe CarteTuilesComponent (sous-classe de JComponent) est un composant Swing capable d'afficher une carte en tuiles fournies par un ou plusieurs fournisseurs de tuiles. Elle possède deux attributs, le niveau de zoom actuel de la carte et la liste des fournisseurs de tuiles.

Le zoom et la liste des fournisseurs de tuiles sont fournis lors de la construction d’un objet CarteTuilesComponent.

Le niveau de zoom, compris entre 12 et 19 inclus, est modifiable par la suite lors d’une interaction avec la molette de la souris. Tout comme le zoom, la liste des fournisseurs de tuiles est modifiable grâce à la méthode setZoom respectivement ajoutFournisseurTuile.

Pour garder le même comportement d’utilisation du composant graphique JComponent, plusieurs méthodes de cette classe doivent être redéfinies.

Premièrement, getPreferredSize retourne la dimension du composant. Dans notre implémentation, cette taille est celle de la carte du monde au niveau du zoom courant.

Puis paintComponent, est appelée à chaque fois que le composant doit être affiché. C’est elle qui sera chargée de dessiner les tuiles des différents fournisseurs les unes par-dessus les autres, dans l'ordre des fournisseurs. Le premier fournisseur transmet le fond de la carte avec des tuiles opaques, tandis que le fournisseur de points couleur superpose des tuiles transparentes.

Le système de coordonnées de CarteTuilesComponent est le système de coordonnées cartésien au niveau de zoom courant, ce qui peut vite devenir gigantesque selon le zoom utilisé (cf. Système cartésien). Cependant la taille du composant n’est pas un problème puisque les parties invisibles ne sont jamais dessinées et aucune mémoire ne leur est allouée.

La partie visible du composant est obtenue à l’aide la méthode getVisibleRect, qui retourne le rectangle visible du composant dans son propre système de coordonnées. C’est dans ce rectangle que l’on va déterminer l'ensemble des tuiles à dessiner.

Lors d’un changement de niveau de zoom ou lorsque la liste des fournisseurs de tuiles est mise à jour, il faut redessiner le composant. Un appel à la méthode repaint est alors effectué.

Concernant l’interaction du déplacement de la carte avec la souris, elle est gérée grâce à un objet JViewport positionné sur une partie de la zone dessinée par CarteTuilesComponent. La position de cette vue bouge et suit les mouvements de la souris.

## Base de données

En premier lieu, nous avons travaillé sur l’implémentation de la base de données. Puis, nous avons implémenté les fichiers de mapping pour que Hibernate puisse connaître la structure de la base de données. Des classes représentant les tables de la base de données ont été créées pour utiliser le code Java afin d’élaborer des requêtes vers la base de données.

Avec l’utilisation du framework Hibernate, la mise en place de requête à effectuer au niveau de la base de données a été simplifiée et accélérée. Nous avons décidé de mettre en place un système de transaction permettant de ne pas altérer la base de données si une requête n’aboutissait pas. N’importe quelle requête renvoie à une liste d’objets. Lors d’une erreur, cette liste n’existe pas (valeur à null), s’il n’y a pas d’objet qui correspond à la requête effectuée la liste est vide.

Une requête est définie par des critères de sélection qui permettent à la base de données de récupérer les enregistrements correspondants.

Prenons l’exemple d’une requête de sélection d’un statut.

Pour exécuter cette requête, il est indispensable d’ouvrir ou de récupérer la session de travail correspondant à la liaison avec la base de données. Puis, pour pallier aux erreurs qui surviendraient lors de l’exécution, une transaction est ouverte.

Dans le cas de notre application, un système de notification des évènements existe. Dû à cela, plusieurs threads peuvent accéder simultanément à cette session de travail. Il est donc indispensable d’en protéger l’accès.



Figure : Ouverture d'une session de travail et d'une transaction

Ensuite, il faut définir la table de la base de données ciblée en utilisant la classe correspondante. Pour définir des critères de sélection à la requête, il est nécessaire de créer des predicats. Ces prédicats sont construits à partir d’un opérateur de comparaison (égal, inférieur, supérieur, etc.), de l’attribut de la table et du critère de sélection fournit par l’utilisateur.

Pour que Hibernate comprenne quel attribut de la table est utilisé lors de la comparaison, il est obligatoire de créer des Metamodels (cf. Figure 16). Ils permettent de connaître le type de l’attribut et de forcer l’utilisateur à fournir un paramètre de type semblable pour que la comparaison ait lieu.

Pour une utilisation standard des fonctions définissant les requêtes, nous avons décidé de gérer les valeurs atypiques des critères de sélection. Si la valeur de celui est null ou vide (pour une chaîne de caractères) alors la requête ne possèdera aucun critère de sélection. Elle récupèra tous les enregistrements de la table ciblée.

Metamodel



Figure : Définition des critères de sélection de la requête

Une fois les critères de sélection définis et transmis à la requête, elle est exécutée. Si tout s’est bien passé, la transaction sera validée et la fonction retournera la liste des enregistrements correspondant aux critères. Sinon une exception sera levée.



Figure : Exécution de la raquête et validation de la transaction

Il est possible d’ajouter des enregistrements à la base de données en créant de nouveaux objets Java et en les transmettant à la base de données. Il faut savoir que chaque nouvel objet créé est un nouvel enregistrement pour la base de données.

Ainsi, pour utiliser un objet contenu dans la base de données, il faut d’abord le récupérer. Pour sa suppression au sein de la base de données, il faut être vigilant aux références de cet objet, car il peut être contenu dans d’autres enregistrements. Si c’est le cas, la suppression pourra avoir lieu selon les règles définies dans la base de données.

Dans notre projet, il n’est pas possible de supprimer des événements car le référencement de ceux-ci ne le permet pas. Ils figurent dans la table des commentaires. En fait, l’action de suppression était tout à fait possible, mais nous avons décidé de ne pas permettre la suppression de commentaires, car nous voulions garder la traçabilité de tous les évènements.

Pour ce qui concerne la mise à jour des objets, Hibernate analyse tous les attributs de l’objet pour déceler la moindre différence entre l’enregistrement et l’objet.

### Modèle de données relationnel

Nous avons pris le parti de mettre en place des triggers et des vues pour apporter une première vérification des données lors de l’enregistrement de celles-ci. Ceci a été pensé dans l’idée de réutiliser la base de données dans une autre application.

Des tables ont été créées pour stocker différents éléments tels que la nationalité, le sexe, l’état civil afin que chaque objet puisse avoir les mêmes valeurs. Dans cette même optique, l’adresse a été décomposée en 3 tables (Rue, Npa, Adresse).

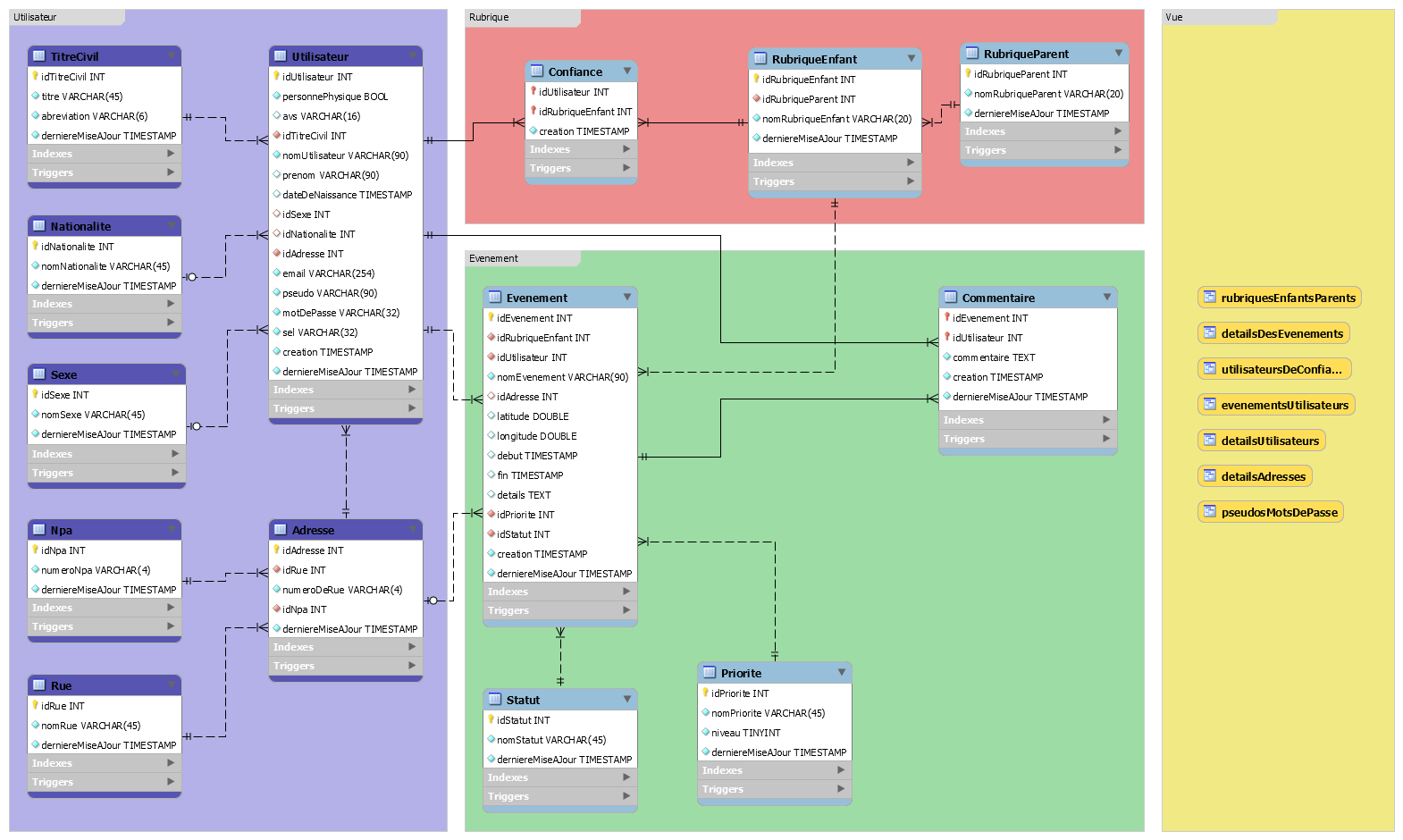


Figure : Modèle de données relationnel

### Diagramme des classes

De nombreuses classes et fonctions ont été implémentées dans le but de proposer une application complète et apte à évoluer sans difficulté.

La structure des classes permettant la création des requêtes a été factorisée au maximum. Une classe a été créée pour chacune des tables afin d’effectuer des requêtes selon un type d’objet. La plupart des fonctions de ces classes appellent des fonctions de la classe DataBaseAccess. Ceci est dû à sa généricité. La récupération d’un élément suivant son type a été implémentée différemment selon chaque classe. Cela s’avérait indispensable pour créer des requêtes correspondant aux critères des objets.

Figure : Diagrammes des classes principales

## Génération de documents PDF

Dans notre projet, il nous paraît important de garder une traçabilité des données. Pour ce faire, nous avons décidé de produire des documents PDF qui contiennent les informations relatives à la rubrique choisie. Ce document est composé de deux parties majeures.

La première donne des informations importantes sur les évènements passés (lieu, dates). La deuxième partie est comporte des statistiques concernant l’ensemble de la rubrique choisie.

Une classe principale, GenerateurPDF, est appelée lorsque l'utilisateur clique sur le bouton PDF de la fenêtre principale. Cette classe crée un document et le remplit avec les éléments présents dans la base de données. Les données collectées servent à réaliser un graphique.

À chaque création de documents PDF, un graphe en barres ou un graphe circulaire est généré de manière aléatoire. Ces graphiques sont créés à partir de la classe GenerateurGraphique qui génère une image.

Dans le cas où l'utilisateur sélectionne plusieurs rubriques, plusieurs documents PDF sont générés. Ils sont tous regroupés dans le dossier SmartCity/PDF du répertoire de l'utilisateur. Ils sont identifiés par la date et le nom de la rubrique sélectionnés.

Si, pour une date donnée, il n'y a pas d'évènements, la première partie du document PDF sera vide. La deuxième partie qui contient les statistiques générales reste telle quelle.

Les documents doivent pouvoir interagir avec la base de données pour récupérer toutes les informations nécessaires à leur création. Pour ce faire, nous avons créé une classe qui permet de se connecter à la base de données et de lier les requêtes SQL avec des fonctions Java.

In fine, un document PDF est généré à partir du bouton PDF de la fenêtre principale. La création d’un ou plusieurs document PDF entraîne l'ouverture du dossier contenant les nouveaux documents. À noter que, si l'utilisateur génère un document PDF et l'ouvre, puis qu'il rédocument depuis l'application, il n'est pas régénéré. Il faut d'abord fermer le document. Ceci est dû au fait que le nom des documents est identique et que le fichier est donc en conflit avec deux applications (le lecteur de document PDF et l'application SmartCity). Dans ce cas, d'un point de vue visuel pour l'utilisateur, il ne se passe rien de spécial.

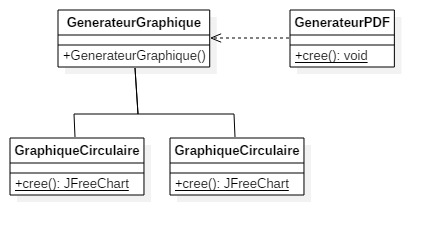


Figure : Schéma de la structure des classes pour la génération d’un document PDF

## Interface graphique utilisateur (GUI)

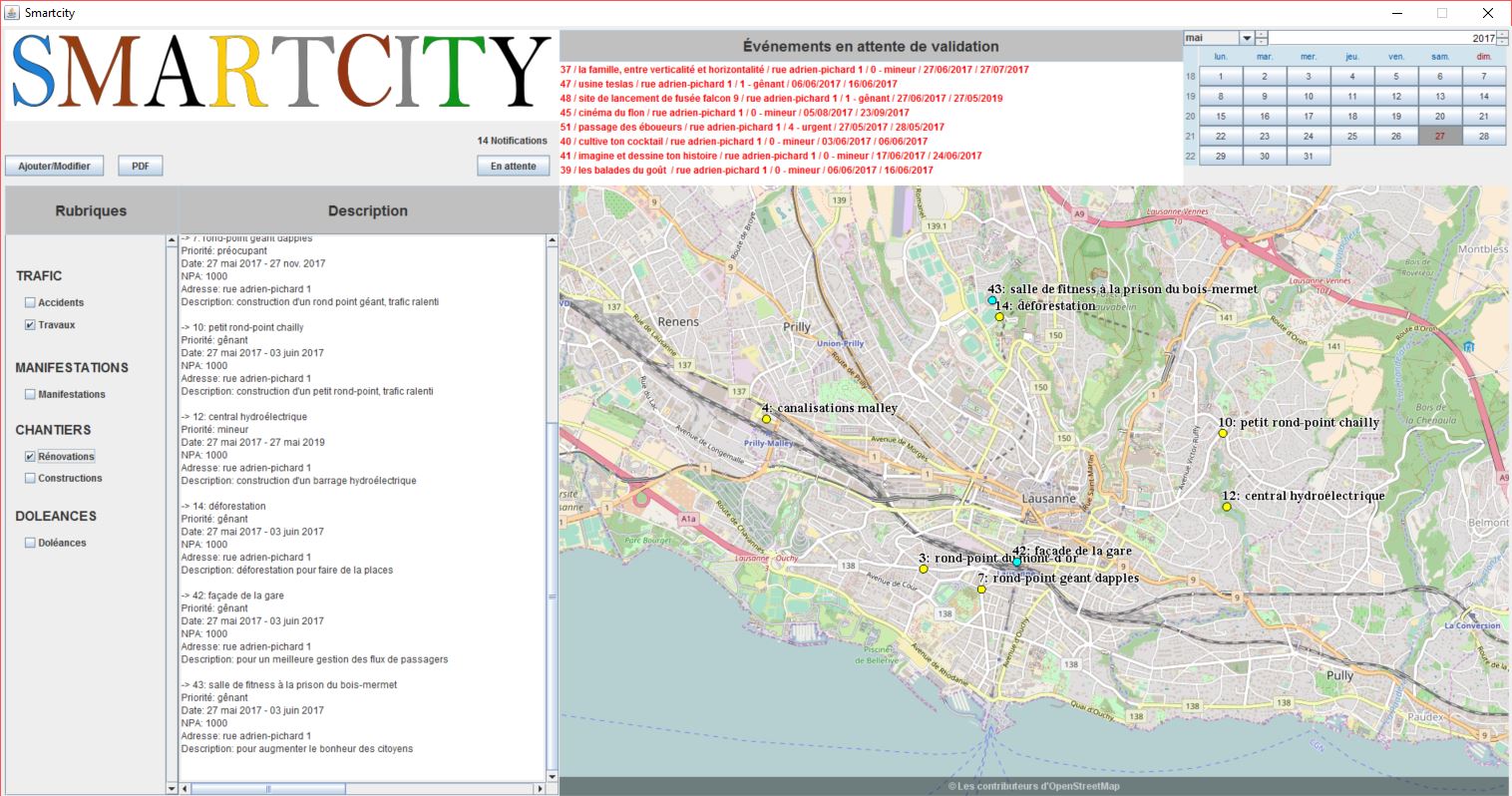
L'application est modélisée par deux fenêtres.

La première fenêtre est la fenêtre "principale" de l'application. Elle permet de voir l'affichage des évènements sur la carte. Les cases de sélection (checkboxes) et le calendrier agissent comme filtres sur les évènements affichés.

Cette fenêtre contient également d’autres boutons offrant les fonctionnalités suivantes : Ajout et modification des évènements, validation des évènements en attente et génération d'un document PDF.

La visualisation des évènements est facilitée par leur positionnement sur la carte et l’affichage de leur description.

Une liste composée d’évènements en attente de validation par l’utilisateur se met à jour automatiquement à intervalle régulier.



1

2

3

8

9

6

7

10

5

4

Figure : Fenêtre principale de l’application

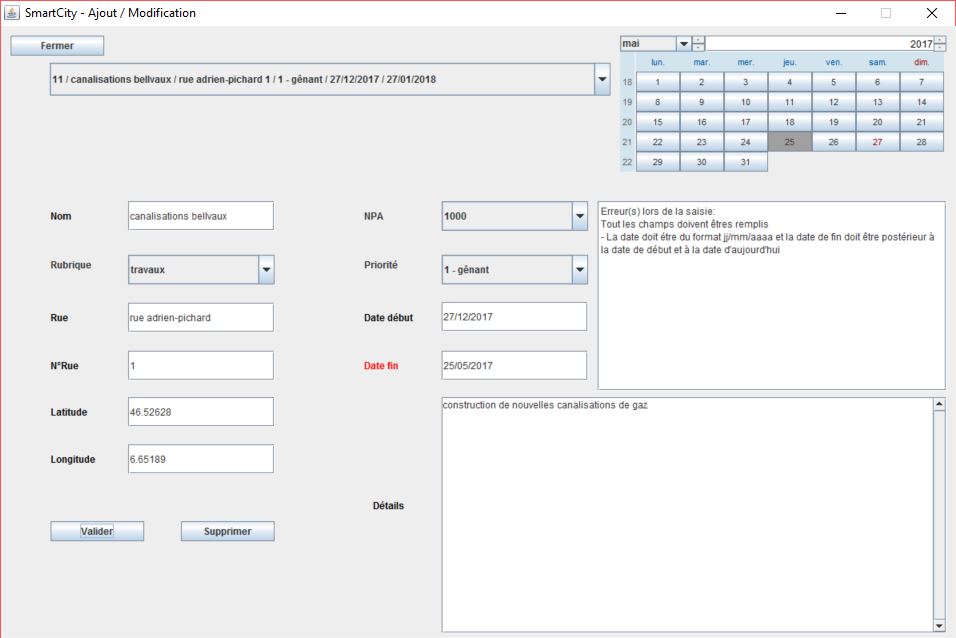
Description des composants de la fenêtre principale :

|  |  |
| --- | --- |
| **Éléments / zone** | **Description** |
| 1. Logo | Représente le logo de l'application |
| 1. Bouton Ajouter/Modifier | Ce bouton ouvre sur la fenêtre d'ajout/modification d'évènements. |
| 1. Bouton PDF | Ce bouton permet de générer un document PDF comprenant les informations relatives à la rubrique choisie (grâce aux checkboxes). |
| 1. Notifications | Affiche le nombre d'évènements en attente de validation |
| 1. Bouton en attente | Ouvre sur la fenêtre permettant de valider ou refuser en évènement en attente. |
| 1. Checkboxes Rubriques | Les cases cochées agissent comme filtre sur les évènements à afficher sur la carte interactive. |
| 1. Description | Affiche les descriptions complètes des évènements filtrés |
| 1. Liste évènements en attente | Liste déroulante affichant les évènements en attente de validation. |
| 1. Calendrier | Agit comme filtre par la date sur les évènements à afficher |
| 1. Carte interactive | Carte interactive affichant la position des évènements filtrés |

La deuxième fenêtre de l’application apparaît dans deux contextes différents. Lorsque l'on veut ajouter ou modifier un évènement de la base de données. La distinction entre ces deux actions est faite par la liste déroulante. Celle-ci propose l’option d’ajouter un nouvel évènement ou d’en sélectionner un et de le modifier. Ces évènements sont définis comme actifs suivant leur date de fin.

Si le choix se porte sur un évènement présent dans la liste déroulante, les champs sont remplis avec les attributs de l’évènement sélectionné. En modifiant la valeur de ces champs, il est possible de modifier l'évènement sélectionné.

Cette fenêtre propose une validation du contenu des champs. Si une valeur n'est pas conforme, le label du champ est affiché en rouge et une note détaille le format du champ à respecter. Cette vérification est effectuée lorsque l’on clique sur le bouton valider et elle se base sur les caractères utilisés et la longueur du contenu grâce à des expressions régulières (regex).



1

2

3

6

7

5

4

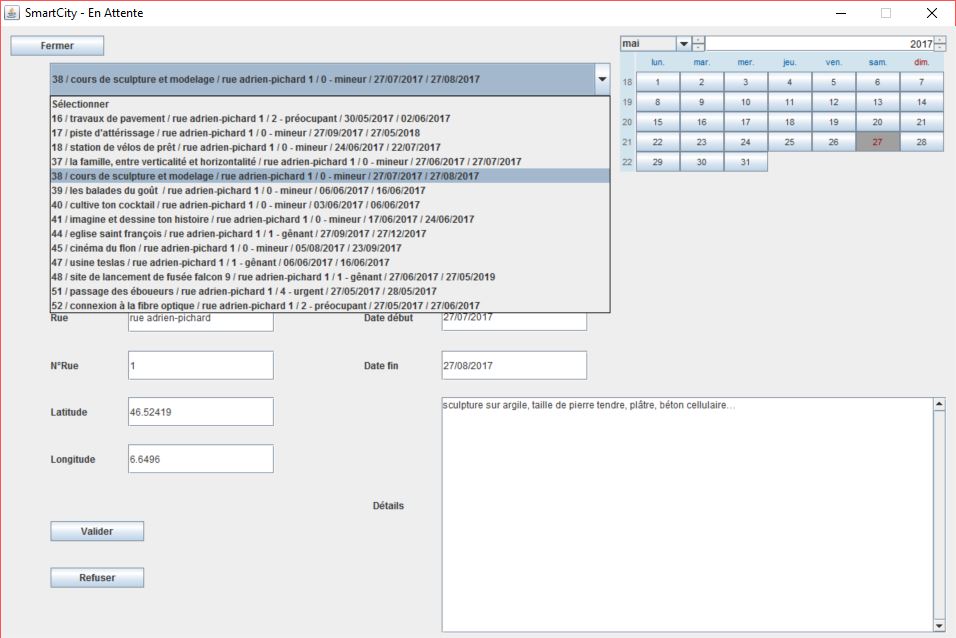
Figure  : Fenêtre d'ajout et de modification des évènements avec notification d’une erreur

Description des composants de la fenêtre d'ajout/modification :

|  |  |
| --- | --- |
| **Éléments / zone** | **Description** |
| 1. Liste déroulante | Permet de choisir si l'on veut ajouter un évènement ou modifier un élément déjà présent dans la base de données |
| 1. Champs | Champs éditables pour définir les attributs d'un évènement |
| 1. Calendrier | Permet de remplir les champs de date de début et de date de fin en interagissant avec les boutons du calendrier |
| 1. Notification d'erreurs | Indique à l'utilisateur le champ incomplet et la forme attendue |
| 1. Bouton fermer | Ferme la fenêtre courante |
| 1. Bouton valider | Contrôle les données saisies, et si elles sont valides, ajoute l'évènement à la base de données, ou met à jour l'évènement si l'utilisateur avait choisi un évènement déjà présent dans la base de données |
| 1. Bouton Supprimer | Supprime l'évènement sélectionné de la base de données |

La même fenêtre permet la validation d'évènements en attente. Dans ce cas, l’utilisateur peut valider ou refuser un évènement.

La liste déroulante permet de sélectionner un de ces évènements. Sa sélection remplira les champs de la fenêtre. L’utilisateur pourra ensuite décider de le modifier ou non avant de le valider.



1

2

3

Figure  : Fenêtre de validation des évènements en attente avec sa liste des évènements déroulée.

Description des composants de la fenêtre de validation :

|  |  |
| --- | --- |
| **Éléments / zone** | **Description** |
| 1. Liste déroulante | Permet de sélectionner un évènement en attente de validation |
| 1. Bouton valider | Valide l’évènement |
| 1. Bouton refuser | Refuse l’évènement |

# Tests de l’application

## Description générale de l’environnement de test

Lors du développement de l'application, nous avons procédé à différents tests. Nous avons tâché d'isoler au mieux les différentes parties telles que la génération de document PDF et la carte interactive afin de mieux déceler pour chacun, ce qui ne fonctionnait pas.

L’application créée étant basée sur des fonctionnalités, il nous parut nécessaire de tester celles-ci afin de valider les objectifs en lien.

En ce qui concerne la sécurité de l’application, nous nous sommes concentrés uniquement sur les saisies de l’utilisateur. La base de données et la communication avec l’applicatif font partie des éléments critiques à protéger, mais nous avons fait le choix de ne pas y consacrer du temps.

Deux machines virtuelles étaient dédiées aux tests de validation. Elles tournaient sous Microsoft Windows 7 SP1 64 bits et Microsoft Windows 10 Familiale 64 bits. Elles ont permis de tester l’application dans des environnements très utilisés aujourd’hui.

Des tests unitaires et d’intégrations ont été effectués durant le développement de l’application sur la machine possédant Microsoft Windows 7. La base de données était stockée en local. Les données utilisées lors des tests étaient des données fictives.

Les conditions exactes des tests unitaires et d’intégrations du projet étaient :

* Microsoft Windows 7 64 bits
* Une connexion au réseau Internet et Intranet de l’HEIG-VD d’Yverdon-les-Bains
* L’archive JAR du projet située à la racine de son disque dur système

Une campagne de validation des cas d’utilisation a été effectuée sur les deux machines virtuelles par deux personnes extérieures au projet, à la fin du développement de l’applicatif. Ceci nous a permis d’obtenir des avis objectifs sur la correspondance entre l’applicatif et les cas d’utilisations annoncées. Ainsi, nous avons pu valider le bon fonctionnement de l’application.

### Matériel

* Deux machines virtuelles de test
  + Intel® Core™ I7 4770 3.40GHz
  + 1 coeur physique, 2 coeurs logiques
  + 2 Go de RAM
  + Connexion réseau : Host-only
  + Taille du disque dur : 40.0 Gb

### Systèmes d’exploitation et outils logiciels

* Microsoft Windows 7 Entreprise 64 bits SP1
* Microsoft Windows 10 Familiale 64 bits

## Tests unitaires

### Carte interactive

Afin de tester les méthodes telles que les constructeurs de points, nous avons utilisé des tests unitaires à l'aide du framework Junit. Il permet d'insérer des tests dans l'application et de référencer des méthodes présentes dans le corps du projet.

Cette méthodologie offre la possibilité de regrouper plusieurs tests à la suite sur un même objet comme un point par exemple. Dans le cas de la carte, nous devions tester la validité des conversions entre les points WGS84 et les points OSM.

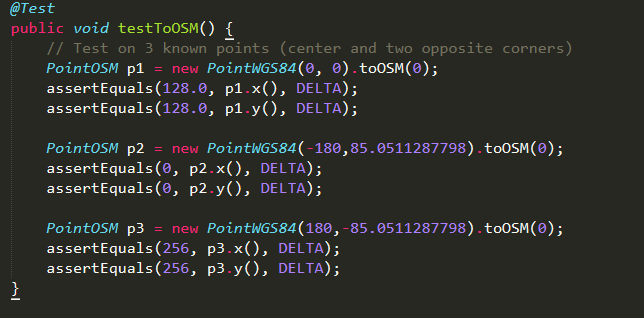


Figure : Test unitaire pour tester la conversion des points WGS84 en OSM

Nous avons aussi effectué des tests sur le zoom qui ne peut pas dépasser certaines valeurs. Dans le cadre de notre application, nous avons décidé de borner le zoom. Celui-ci n'est pas utile au-delà de certaines valeurs et rend les informations sur la carte illisible. Grâce au test unitaire, nous pouvons tester une série de points et vérifier que le point reste valide.

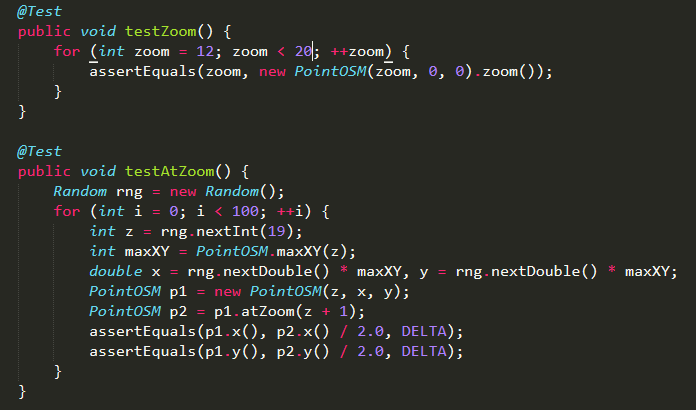


Figure : Test unitaire sur le zoom de la carte

### Base de données

### Génération de documents PDF

Nous avons essayé de générer un document PDF dans des conditions inhabituelles, afin de nous assurer que tout fonctionnait. En voici quelques-unes :

* Si la base de données est vide : le document PDF généré contiendra le logo d'entête, le titre et la date ainsi qu'un message indiquant qu'il n'y a pas de données.
* S'il n'existe aucun évènement à la date sélectionnée, la première partie du document PDF sera vide, mais la deuxième partie contiendra les statistiques.
* Si l'utilisateur crée deux fois le même document PDF (même rubrique, même date) : le deuxième document PDF écrasera le premier. Nous avons fait le choix de ne pas conserver le document précédent pour la bonne raison que le document PDF créé est plus récent.
* Si plusieurs rubriques sont sélectionnées : plusieurs documents PDF qui seront générés.

## Tests d’intégrations

Les tests d’intégrations ont été effectués lors du regroupement des différents composants de l’application.

L'intégration de la carte dans l’application n'a pas nécessité de tests puisqu’elle n’a pas d’incidence sur les autres composants. Suite à la création de tuiles avec des points de couleurs, le test de l’affichage de la localisation des évènements s’est déroulé rapidement.

L'intégration de la base de données a dû être testée directement. Nous avons dû vérifier que les méthodes mises à disposition de la base de données permettaient par exemple de renvoyer une liste d'évènements actifs. Il fallut par la suite ajouter des méthodes qui n'avaient pas été prévues initialement.

La génération de document PDF n'a pas été testée lors de son insertion dans l'application. Celle-ci a été testée séparément pour la mise au point de la structure de la page (titre, type de graphique, entêtes, etc.). Puis, elle a été testée lorsque la base de données contenait suffisamment de données.

## Tests des cas d’utilisation

Nous avons testé séparément les différentes parties de l’application, telles que la génération de document PDF et la carte interactive afin de mieux déceler ce qui ne fonctionnait pas. Après le regroupement des parties, nous avons commencé à tester les cas d'utilisations.

Les premiers tests ont concerné la récupération d'évènements de la base de données afin que ceux-ci apparaissent correctement dans la fenêtre de modifications. On peut se reporter la Figure 17 encadré 1 et voir que les évènements apparaissent. Nous avons ensuite vérifié que ces évènements correspondaient à ceux présents de la base de données.

Nous avons ensuite testé la modification des évènements présents dans la base de données et vérifié que la modification se répercute bien sur les différentes fenêtres et dans la base de données.

* L'ajout de requête en attente se rajoute automatiquement dans la liste des évènements en attente de validation. N'ayant pas fait d'application qui permette d'envoyer des évènements directement dans la base de données, nous avons simulé l'ajout et celui fonctionne correctement. De plus si la liste dépasse le nombre d'affichages elle fait une mise à jour et affiche un mélange entre ceux déjà affichés et ceux qui ne le sont pas.

Un autre test fut la sélection des évènements par dates.

* Si l’utilisateur sélectionne une catégorie d'évènement à une date précise alors ceux de cette catégorie s'affichent correctement sur la carte.
* Si l’utilisateur choisit de cocher plusieurs rubriques, seuls les évènements actifs à cette date s’affichent sur la carte.

Il est possible de valider les évènements en attente. Ceux-ci apparaissent dans la liste des évènements modifiables et disparaissent de la liste des notifications.

Les tests sur la génération des documents PDF ont porté sur la sélection d’une ou plusieurs rubriques et d’une date. La vérification des documents PDF s’est basée sur la similarité de leur contenu et de celle de la base de données.

# Conclusion

#### État des lieux

Au vu des délais à respecter, nous n'avons pas pu implémenter les fonctionnalités suivantes :

1. La gestion de plusieurs comptes administrateur
2. Dessins et coloriages sur la carte selon des critères de secteurs (cercles, lignes, texte)
3. Ajout d'une capture d'écran dans le document PDF de l'état de la carte
4. Calcul de chemin le plus court en évitant les zones accidentées ou en travaux

Les fonctionnalités principales décrites dans le cahier des charges au début du projet sont présentes. Lors de la réalisation du projet, quelques points ont été modifiés. Par exemple la date de fin, qui était optionnelle au départ, est devenue obligatoire. Elle est toutefois modifiable pour pouvoir allonger ou raccourcir la durée d'un évènement.

De plus, une rubrique a dû être simplifiée. Initialement, « Doléances » devait regrouper plusieurs sous-rubriques créées dynamiquement en fonction des données soumises par les citoyens. Ainsi, nous avons décidé de transformer la rubrique « Doléances » en une rubrique simple.

Les fonctionnalités supplémentaires ne sont pas indispensables pour l'utilisation de l'application, mais elles peuvent toutefois améliorer le fonctionnement de cette dernière.

C’est pourquoi nous avons décidé de ne pas les développer, celles-ci requérant également une charge de travail trop conséquente (calcul du chemin le plus court).

L'application ne possède pas de fonctionnalité d'authentification. Dans le cahier des charges, il s’agit d’une fonctionnalité supplémentaire qui aurait dû permettre la gestion de plusieurs comptes administrateurs. Ainsi, nous n’avons pas de moyen d’authentification et considérons l'application utilisable uniquement sur le poste d'un administrateur.

À l'heure actuelle, nous pouvons appliquer plusieurs opérations sur la carte interactive, comme déplacer la vue, zoomer et dézoomer. Mais il n'est pas possible de dessiner sur la carte.

Pour la génération de document PDF, il est possible de générer des documents en deux parties : une première partie contenant les informations des évènements du jour et une seconde partie contenant des statistiques générales sur la rubrique choisie. Une fonctionnalité supplémentaire proposée était de mettre une capture d'écran de l'état de la carte dans le document PDF. Cette fonctionnalité n'a pas été réalisée.

## Problèmes rencontrés

Nous avons rencontré plusieurs problèmes auxquels nous n’étions pas préparés et nous avons sous-estimé la charge de travail. Lors de l'élaboration initiale du projet, nous avions conscience de la difficulté de certaines tâches, mais nous ne savions pas précisément quelles technologies nous allions utiliser pour développer l’application et ces différents composants.

Tout d’abord, la mise en place de la base de données fut beaucoup plus longue que prévu. Ce qui a retardé l’accès et l’intégration des données au projet.

Composante phare de notre application, la carte interactive affichait des performances en deçà de nos attentes. Il a fallu trouver un moyen de pouvoir charger et afficher rapidement les tuiles de la carte. Après avoir effectué des recherches, nous avons compris qu’il était possible de mettre en cache les tuiles. Ce processus permet de les afficher plus rapidement au sein de l’application.

Une autre difficulté rencontrée fut de rassembler les principaux composants (carte, base de données et génération de document PDF). Au début de l’élaboration des composants, nous avons décidé de partir sur des sous-projets Java sans utiliser Apache Maven. Mais lors de l’intégration des composants, nous avons eu des difficultés en raison de l’utilisation de différentes librairies pour chacun des sous-projets. Pour remédier à ce problème, nous avons décidé de convertir les composants en sous-projet Maven afin d’avoir plus de facilité lors de l’intégration.

## Propositions d’améliorations

### Interface graphique de l’utilisateur

Lors du choix de la librairie de l'interface graphique, nous ne connaissions que JavaSwing. Elle suffisait à l'implémentation de notre application. Mais nous avons appris l'existence de JavaFX, qui à l'instar de Swing permet d'implémenter une interface graphique. Toutefois, JavaFX s'organise différemment et permet d'avoir un code beaucoup plus organisé et facile à lire. De plus, le design est plus agréable.

Une amélioration possible de notre application aurait été de passer de Swing à JavaFX. Nous n'avons pas pu effectuer cette modification dans les délais étant donné qu'une grosse partie de notre application concerne la partie graphique et qu’elle représentait une charge de travail trop importante pour être terminée à temps.

De plus, nous pourrions améliorer l'aspect général de l'application. Pour le moment, la taille de la fenêtre principale n'est pas modifiable. Pour ajouter du confort à l'utilisation, il faudrait rendre l’application « responsive design ».

Toujours dans un souci de design, nous pourrions créer et utiliser nos propres composants graphiques pour un « look » unique. Toutefois, comme nous avons mis l'accent sur les fonctionnalités, ces améliorations ne sont pas prioritaires.

### Formulaire des évènements

Une amélioration possible du formulaire serait de changer le contrôle de la saisie du nom de la rue d'un évènement. À l'heure actuelle, nous contrôlons que les caractères soient conformes à un nom de rue, mais nous ne pouvons pas garantir son existence dans la ville de Lausanne.

Nous pourrions ajouter un contrôle de vérification. Pour ce faire, une liste proposerait les noms des rues possibles en fonction des premières lettres tapées par l'utilisateur.

### Génération de documents PDF

Nous pourrions imaginer une amélioration au niveau des statistiques. Avec des évènements sur plusieurs années, les statistiques fournies seraient plus complètes. Il serait ainsi possible de voir une évolution des chantiers dans la ville de Lausanne ou encore de pouvoir repérer les meilleurs lieux pour des manifestations culturelles.

# Annexes

## Cahier des charges

## Journaux de travail

Tano Iannetta

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Semaine | Activités | | | |
| Tâche planifiée | Résultat du travail effectué | Heures prévues/  effectives |
| Semaine 1 | Recherche de sujet et discussion de groupe | Choix du 1er sujet | 3/3 |
| Semaine 2 | Mise en place github  Rédaction readme  Canevas | Travaille sur le cahier des charges du 1er sujet  Recherche d'un 2eme sujet | 6/8 |
| Semaine 3 | Choix 2eme sujet  Début interface graphique | Choix du 2eme sujet  Cahier des charges | 9/6 |
| Semaine 4 | Avance interface graphique | Spécification de l’interface graphique des deux fenêtres de l'application (avances générales) | 5/5 |
| Semaine 5 | Avance de la fenêtre principale de l'interface graphique, Maintenance github | Spécification et implémentation de l’interface graphique des deux fenêtres de l’application (avances générales) | 9/8 |
| Semaine 6 | Finalisation de la fenêtre principale | Implémentation de la fenêtre de modification des évènements et du remplissage des champs | 7/9 |
| Semaine 7 | Communication avec la base de données, Apprentissage Java pour la communication avec une base de données | Contrôle de saisie de la fenêtre de modification + intégration de la carte dans la fenêtre principale | 8/7 |
| Semaine 8 | Continuation de la communication entre base de données et fenêtre principale | Préparation de la présentation  Tests et corrections de la fenêtre de modification  (valeurs fictives, sans base de données) | 4/4 |
| Semaine 9 | Gestion interface graphique avec la carte interactive | Intégration de la base de données avec l'interface graphique (ajout et suppression d'évènements.) | 8/7 |
| Semaine 10 | Implémentation de la fenêtre d'ajout/modification des évènements | Intégration de la base de données avec l'interface graphique (ajout et suppression d'évènements.) | 4/7 |
| Semaine 11 | Fusion des fonctionnalités | Adaptation du code et corrections de divers bugs de la base de données et l'interface graphique  Implémentation de la fonctionnalité de modifier un évènement | 5/8 |
| Semaine 12 | Merge des fonctionnalités | Tests et corrections de la fenêtre de modification avec la base de données | 5/5 |
| Semaine 13 | Tests de l'application | Finalisation du code de l'interface graphique, Tests de l'application | 5/13 |

Loan Lassalle

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Semaine | Activités | | | |
| Tâche planifiée | Résultat du travail effectué | Heures prévues/  effectives |
| Semaine 1 | Constitution des groupes, choix du sujet et discussions | Nomination du chef de projet et du second, proposition de projet : coffre-fort numérique | 3/4 |
| Semaine 2 | Commencement du cahier des charges, feedback du professeur | Début du cahier des charges, recherche d’un nouveau sujet en raison du refus du précédent sujet, proposition de projet : gestionnaire de requêtes citoyennes, rédaction du cahier des charges du nouveau projet | 3/6 |
| Semaine 3 | Cahier des charges du projet avec planning | Finalisation du cahier des charges et du planning | 5.5/5.5 |
| Semaine 4 | Conception de la base de données | Conception de la base de données, recherche des technologies adaptées au projet pour la liaison avec la base de données | 6/8 |
| Semaine 5 | Conception de la base de données | Choix de la technologie utilisée : HIBERNATE, compréhension du fonctionne et de son implémentation | 6/10 |
| Semaine 6 | Réalisation de la base de données, implémentation de la liaison avec la base de données | Modification et mise en place de la base de données, implémentation de la traduction de code Java en requête SQL | 6/10 |
| Semaine 7 | Réalisation de la base de données | Finalisation de l’implémentation de fonctionnalité | 7/8 |
| Semaine 8 | Intégration de la base de données avec l’interface graphique | Mise en commun avec la carte, simplification de l’implémentation | 7/7 |
| Semaine 9 | Intégration de la base de données avec l’interface graphique | Correction de la structure de la base de données (trigger), intégration de la liaison de la base de données avec l’interface graphique | 7/7 |
| Semaine 10 | Intégration de la base de données avec l’interface graphique | Mise en place de l’ajout d’évènements au sein de l’interface graphique, simplification de fonctionnalités | 7/7 |
| Semaine 11 | Intégration de la base de données avec l’interface graphique, rédaction du rapport final | Mise en place de la modification d’évènements au sein de l’interface graphique, mise en place du rapport, début de rédaction | 5.5/7 |
| Semaine 12 | Rédaction du rapport final | Rédaction du rapport final, finalisation des commentaires dans le code | 5.5/7 |
| Semaine 13 | Rédaction du rapport final | Finalisation du rapport et journal de bord | 5.5/7 |

Luana Martelli

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Semaine | Activités | | | |
| Tâche planifiée | Résultat du travail effectué | Heures prévues/  effectives |
| Semaine 1 | Constitution des groupes, choix du premier sujet et discussions | C.f. tâche planifiée | 4 / 4 |
| Semaine 2 | Cahier des charges, recherche d’un 2ième sujet | C.f. tâche planifiée, ébauche d'analyse du PDF | 6 / 6 |
| Semaine 3 | Cahier des charges du 2ième projet avec planning | C.f tâche planifiée, ébauche d'analyse des statistiques | 6 / 6 |
| Semaine 4 | Recherches / analyse génération de PDF | Recherche de libraires / JDK, première ébauche de PDF | 6 / 6 |
| Semaine 5 | Recherches / analyse génération de graphiques | Ajout des statistiques au PDF | 5 / 4 |
| Semaine 6 | Implémentations | Données écrites en brut, commentaires, nettoyage du code | 6 / 6.5 |
| Semaine 7 | Implémentations | Structure du code, continuation du PDF | 6 / 4 |
| Semaine 8 | Implémentations | Structure du code, continuation du PDF | 5.5 / 5 |
| Semaine 9 | Implémentation | Début de la connexion avec la base de données | 5.5 / 5 |
| Semaine 10 | Liaison du PDF avec la base de données | Connexion avec la base de données / peuplement | 6 / 6 |
| Semaine 11 | Liaison du PDF avec la base de données | Fin de la connexion avec la base de données / début de la connexion avec l'interface graphique | 6 / 10 |
| Semaine 12 | Rédaction du rapport final / Connexion PDF-interface graphique | Connexion avec l'interface graphique, tests | 6 / 10 |
| Semaine 13 | Rédaction du rapport final | Rédaction du rapport | 6 / 5 |

Wojciech Myszkorowski

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Semaine | Activités | | | |
| Tâche planifiée | Résultat du travail effectué | Heures prévues/  effectives |
| Semaine 1 | Constitution des groupes, choix du premier sujet et discussions | c.f: tache planifié | 3/3 |
| Semaine 2 | Cahier des charges, recherche d’un 2ième sujet | c.f: tache planifiée | 5/6 |
| Semaine 3 | Cahier des charges du 2ième projet avec planning | c.f: tache planifiée | 5/7 |
| Semaine 4 | Mise au point implémentation de la carte | Conception théorique des points OSM 1er jet du diagramme de classe de la carte | 5/6 |
| Semaine 5 | Recherches sur les systèmes de points wgs84 et OSM  Codage des premières classes de la carte | Mise en place de classe Utils et méthodes de conversion des points. mise en place de tests pour vérifier les calculs de conversion | 5/6 |
| Semaine 6 | Implémentation de la carte  Avancement code sur les tuiles | Implémentation des classes présentes dans le diagramme de classe pour la carte | 5/6 |
| Semaine 7 | Implémentation de la carte | Ajout de la carte à l'application | 5/3 |
| Semaine 8 | Implémentation des points d'affichage sur la carte | Résolution des bugs sur la carte en ce qui concerne le zoom et le déplacement long | 7/7 |
| Semaine 9 | Résolution des problèmes d'ajouts de la carte à l'application principale | Résolution des bugs sur la carte en ce qui concerne la mise en cache des tuiles | 7/7 |
| Semaine 10 | Résolutions des bugs sur la carte | Rajout de commentaires dans les classes de la carte et mise au propre du code | 7/7 |
| Semaine 11 | Codages des dernières fonctionnalités de l'application | Rédaction du rapport final  Implémentation des notifications sur la GUI | 7/7 |
| Semaine 12 | Rédaction du rapport final  Et tests finaux sur l'application | Rédaction du rapport final  Implémentation de la mise à jour des notifications des dans la Base de données | 7/5 |
| Semaine 13 | Finalisation rapport et relecture | Rédaction du rapport final  Tests finaux pour l'application | 5/7 |

Camilo Pineda

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Semaine | Activités | | | |
| Tâche planifiée | Résultat du travail effectué | Heures prévues/  effectives |
| Semaine 1 | Constitution des groupes, choix du premier sujet et discussions | Cf : tâche planifiée | 4/3 |
| Semaine 2 | Cahier des charges, recherche d’un 2ième sujet | Cf : tâche planifiée | 6/5 |
| Semaine 3 | Cahier des charges du 2ième projet avec planning | Cf : tâche planifiée | 4/4 |
| Semaine 4 | PDF et génération de fichiers | Conception base de données | 4/3 |
| Semaine 5 | PDF et Conception de la base de données | PDF première partie, titre, infos | 6/6 |
| Semaine 6 | GUI Réalisation de la base de données | PDF début seconde partie, statistiques | 6/5 |
| Semaine 7 | Rapport, PDF |  | 6/5 |
| Semaine 8 | Documentation, GUI | Assistance GUI | 6/5 |
| Semaine 9 | PDF tests, GUI | rapport | 6/5 |
| Semaine 10 | Mise en commun code, documentation | Assistance GUI | 6/5 |
| Semaine 11 | Documentation rapport, mise en commun | Rédaction rapport et manuel utilisateur | 6/5 |
| Semaine 12 | Tests, debug | Cf : tâche planifiée + documentation : manuel utilisateur | 6/8 |
| Semaine 13 | Rapport et documentation, tests | Cf : tâche planifiée | 6/ |

Jérémie Zanone

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Semaine | Activités | | | |
| Tâche planifiée | Résultat du travail effectué | Heures prévues/  effectives |
| Semaine 1 | Constitution des groupes, choix du premier sujet et discussions | Cf: tâche planifiée | 2.5/3 |
| Semaine 2 | Cahier des charges, recherche d’un 2ième sujet | Cf: tâche planifiée | 5.5/5.5 |
| Semaine 3 | Cahier des charges du 2ième projet avec planning | Cf: tâche planifiée | 5.5/5.5 |
| Semaine 4 | Conception de la carte et implémentation de point pour la carte | Conception de la carte et implémentation de points de coordonnées pour les évènements sur la carte ainsi qu'implémentation des évènements | 6.5/7 |
| Semaine 5 | Création des tuiles pour la carte | Création des fournisseurs de tuiles et de l'interface graphique pour les tuiles de la carte | 5.5/6 |
| Semaine 6 | Création de l'interface graphique contenant les tuiles | Continuer l'interface graphique du carteTuilesComponent + tests d’affichages et résolutions de beug | 5.5/5.5 |
| Semaine 7 | Ajouter un pin de couleur sur la carte pour chaque évènement | Ajout des informations et pin selon une liste d'évènements sur la carte + ajouter la carte à la fenêtre principale | 5.5/8 |
| Semaine 8 | Avancer le code pour la présentation intermédiaire | Ajout de l'interaction de la carte avec la souris | 5.5/6 |
| Semaine 9 | Ajout de l'interaction sur la carte | Poursuite de l'interaction carte et ajout d'une classe pour garder les tuiles utilisées dans un cache (ArrayList) | 6/9 |
| Semaine 10 | Apprendre comment récupérer les informations depuis la base de données | Ajouter les points des évènements sur la carte selon les checkboxes et la date sélectionnées | 5/7 |
| Semaine 11 | Linker la base de données avec la carte | Ajouter les descriptions des évènements selon les checkboxes et la date sélectionnées | 7/6 |
| Semaine 12 | Rédaction du rapport final | Cf: tâche planifiée + amélioration du code | 8/8 |
| Semaine 13 | Rédaction du rapport final | Cf: tâche planifiée + commenter le code | 8/9 |

## Planification initiale et son évolution

Comparaison entre ce qui a été prévu et ce qui a été fait.

Lors de l'élaboration de notre planning initial, nous n'avions pas décidé de toutes les technologies que nous allions utiliser, par exemple Hibernate pour la base de données. Ceci nous a retardés dans notre planning initial. Certains choix ont provoqué des petites replanifications au cours des semaines comme par l'exemple l'intégration de la carte qui n'était pas complètement fonctionnelle en semaine 9. Il y a aussi eu le facteur de la sous-estimation de la charge de travail que certaines fonctionnalités ont requise, notamment nous voulions mettre des icônes de différents types et nous nous en sommes rendus compte qu'après un certain qu'il fallait opter pour des points de couleur simples.

Dans le cas de notre application, la base de données joue un rôle très important, toutes les fonctionnalités sont en interactions avec celle-ci. Durant notre projet, il y a plusieurs ajustements des méthodes et du comportement de la base de données qui ont parfois mené à des petits recalibrages de certaines fonctionnalités.

L'interface graphique a quant à elle bien suivi le planning initialement prévu, les derniers soucis ont été réglés comme prévu vers la fin du projet. Ceux-ci concernaient principalement de l'optimisation de code ainsi que dans la vérification de la bonne exécution des différentes tâches de l'application. Un effort supplémentaire a été fait pour préserver l'intégrité des données affichées à l'écran par rapport aux changements effectués sur la base de données.

## Rapport de tests

# Sources – Bibliographies

## Ouvrages

* SAVITCH, Walter. Absolute Java. 5ième édition, San Diego, Pearson, 2013, 1260 pages
* DELANNOY, Claude. Programmer en Java. 9ième édition, Saint-Germain, Eyrolles, 2015, 916 pages

## Sites Internet

Les sites Internet ci-dessous ont tous été consultés du 20.02.2017 au 30.05.2017.

* Développez : <http://www.developpez.com/>
* iText : <http://itextpdf.com/>
* JBoss HIBERNATE : <https://docs.jboss.org/hibernate/stable/core.old/reference/fr/html/>
* JFreeChart : <http://www.jfree.org/jfreechart/>
* Microsoft : <http://blogs.msdn.com/>
* Microsoft : <http://www.labo-microsoft.org/>
* Microsoft : <https://msdn.microsoft.com/>
* Microsoft : <http://support.microsoft.com/>
* Stackoverflow : <http://stackoverflow.com/>
* Wikipedia : <http://wikipedia.org/>
* YouTube : <https://www.youtube.com/>
* EPFL : <http://cs108.epfl.ch/archive/14/>
* Oracle: <https://docs.oracle.com/en/java/>
* OpenClassRoom: <https://openclassrooms.com>

## Vidéos en ligne

* edureka!. Hibernate Tutorial - 1 | Hibernate Tutorial for Beginners - 1 | Edureka. [Tutoriel en ligne]. <https://www.youtube.com/watch?v=iwMW\_SZPGnY>. (Consulté le 13 mars 2017).
* edureka!. Hibernate Tutorial | Hibernate Tutorial - 2 | Hibernate Tutorial for Beginners - 2 | Edureka. [Tutoriel en ligne]. <https://www.youtube.com/watch?v=VAr\_IikcYfE>. (Consulté le 13 mars 2017).
* edureka!. Hibernate - The Ultimate ORM Framework | Edureka. [Tutoriel en ligne]. <https://www.youtube.com/watch?v=xo\_HGo812DQ>. (Consulté le 13 mars 2017).

# Glossaire

API Application Programming Interface

DD Degrés Décimaux

DMS Degré Minutes Secondes

GUI Graphical User Interface

HEIG-VD Haute École d’Ingénierie et de Gestion du Canton de Vaud

NPA Numéro Postal d'Acheminement

OSM OpenStreetMap

PDF Portable Document Format

Regex Regular Expression

SGBDR Système de Gestion de Base de Données Relationnelles

WSG84 World Geodetic System 1984