RAPPORT

OUTIL D'IMPORT OPENSTREETMAP
PROJET CHINOOK

Conception

Novembre 2015

Memoris
INGENIERIE GÉOGRAPHIQUE

Rédigépar :	Florent RICHARD	Visa
Validépar :	Julien MARGAIL	Visa
Approuvépar :	Christophe CHANTRAINE	Visa

Dated'émission : 05/11/2015	Version 1.0

ÉTATDESÉDITIONSETDESRÉVISIONS			
N° ÉditionsetRévisions	Dated'émission	Raisondel'évolution	Descriptiondelamodification
1.0	05/11/2015	Version initiale (Architecture et Partie Import)	

Sommaire

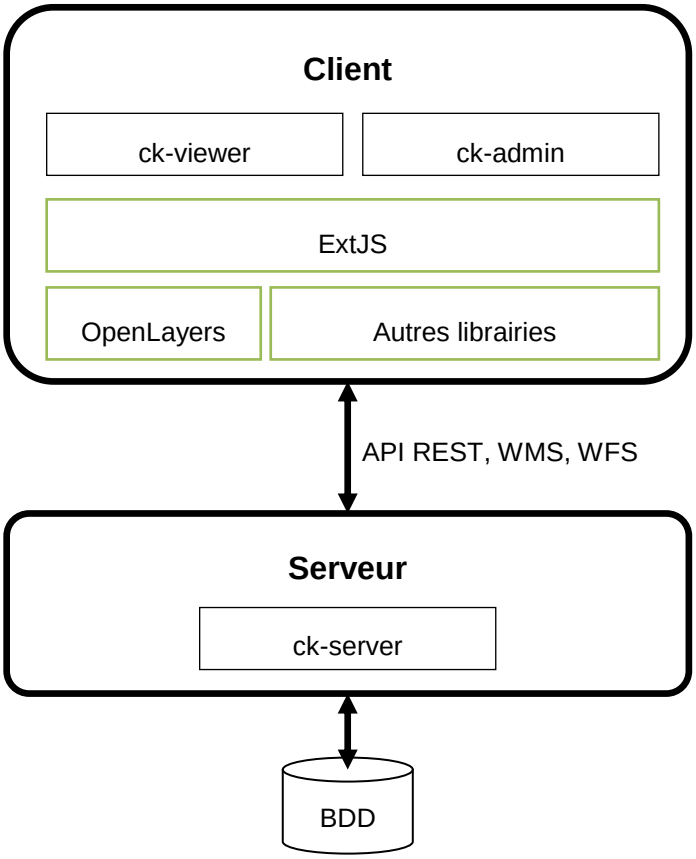
1 Introduction

Ce document présente la conception de l'outil d'import de données OpenStreetMap dans la plateforme Chinook2. Il reprend l'architecture de ck-viewer (Chinook2).

2 Architecture

2.1 Architecture Chinook2

Chinook2 se décompose de la manière suivante :



Il se décompose en 2 parties :

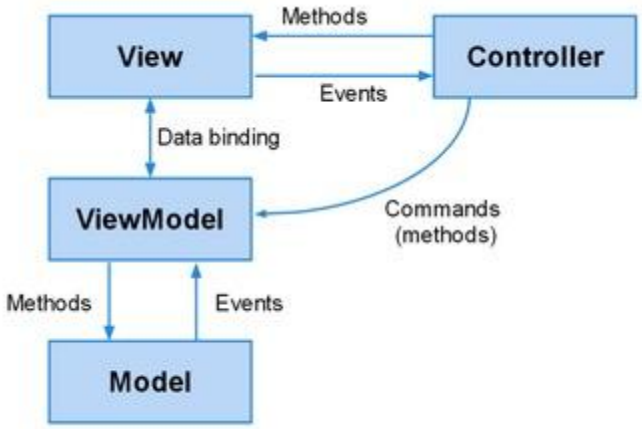
- **Client** :
  - ck-viewer est l'interface utilisateur permettant en particulier d'afficher des cartes et des données. L'outil d'import OSM sera intégré à ck-viewer.
  - ck-admin est l'interface d'administration de l'application
- **Serveur** :
  - ck-server. Le serveur expose une API REST pour échanger des données simples avec le client. Il expose également des services WMS (Web Map Service) et WFS (Web Feature Service) pour les données géographiques.

Ck-viewer utilise divers framework et librairies :

- Sencha ExtJS V6.0.0 (Licence GPL) : librairie JavaScript permettant de construire une application web interactive de manière structurée
- OpenLayer V3.10.1 (Licence BSD) : librairie JavaScript permettant de gestion de cartes multicouches en utilisant différentes projections
- Diverses autres librairies

2.2 Architecture ck-viewer - MVVM

A l'aide de ExtJS, l'architecture de ck-viewer est basée sur le design pattern MVVM (Model – View – ViewModel). Un Controller étant également présent sous la forme d'un ViewController (un ViewController étant instancié pour chaque instance de vue).

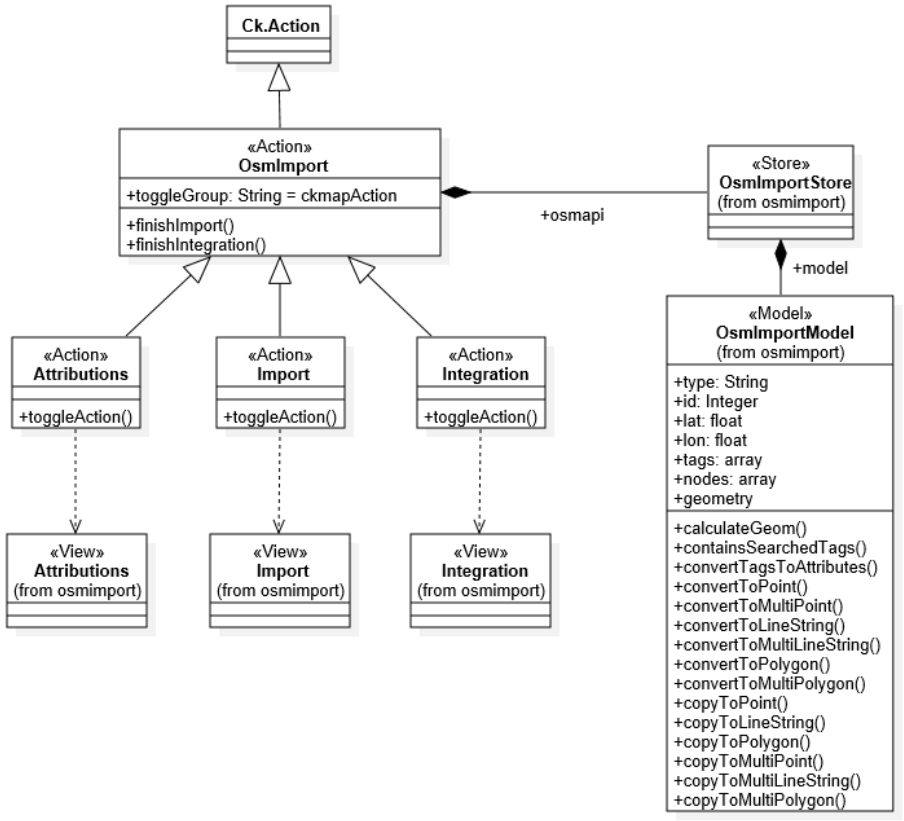


Ck-viewer définit également quelques classes de bases pour simplifier le développement, parmi lesquelles :

- Ck : permet de récupérer des objets relatifs au projet tels que la carte
- Ck.Controller : définit notamment des méthodes et évènements pour la gestion des cartes
- Ck.Action : définit des méthodes pour simplifier la gestion des actions (dans les boutons)

3 Barre d'outils

Le schéma suivant présente le diagramme des classes utilisé pour la gestion de la barre d'outils, pour la partie OSM Import.



Dans la barre d'outils, chaque bouton est une classe Action.

La classe OsmImport représente le bouton de menu et est utilisée comme classe mère pour toutes les autres actions qui elles représentent les sous-menus.

La classe OsmImport contient des éléments communs aux différentes parties de l'outil :

- **toggleGroup** : définition du groupe pour afficher et masquer les sous-menus
- **Store OsmImportStore :** ce store permet de récupérer les données depuis OpenStreetMap puis de les stocker en vue de leur intégration. Il utilise le **Model OsmImportModel** pour ces données. Comme le panneau d'import et le panneau d'intégration partagent ce store, les données importées sont accessibles aux deux.
- **finishImport() :** cette méthode permet de signifier la fin de la phase d'import du workflow. L'outil se place donc en phase d'intégration.
- **finishIntegration() :** cette méthode permet de signifier la fin de la phase d'intégration du workflow. L'outil se place donc en phase d'import.

Détails des actions de sous-menus :

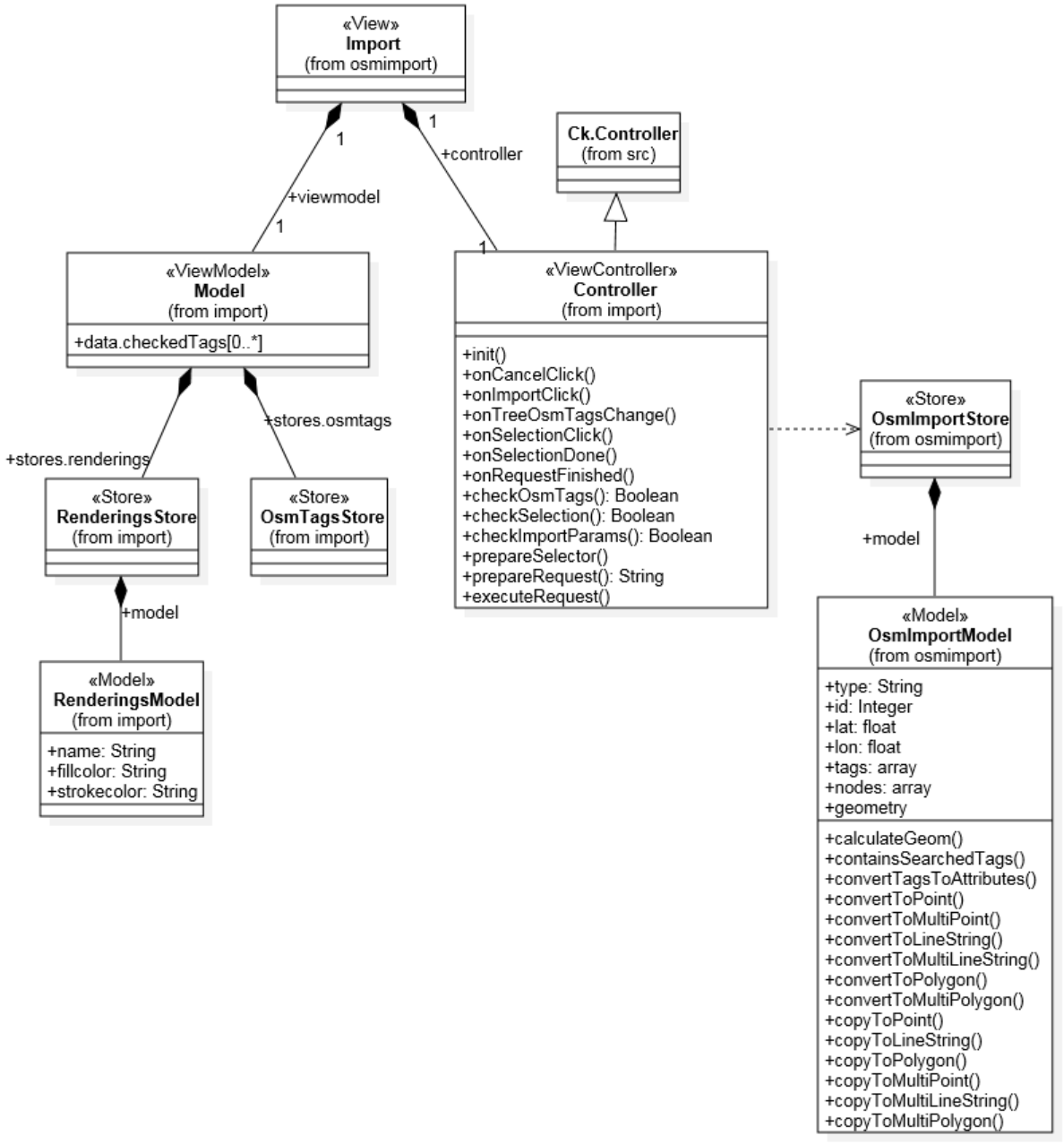
- **Action Attributions** : ce sous-menu affiche une vue Attributions.
- **Action Import** : ce sous-menu affiche une vue Import.
- **Action Integration** : ce sous-menu affiche une vue Integration

Les boutons sont utilisés en « toggle », c'est-à-dire qu'ils permettent alternativement d'afficher puis de masquer la vue.

4 Partie 1 : Import des données

4.1 Diagramme de classes

Le schéma suivant présente le diagramme des classes utilisé pour la partie import des données :



On retrouve le design pattern MVVM associé à un ViewController. Le détail de ce schéma est présenté dans chacune des sous-parties.

Pour les parties communes, la méthode **init()** du ViewController est exécutée lors de l'instanciation de la Vue et de son Controller.

4.2 Sélection des Tags OSM

Diagramme des classes

Le diagramme est disponible au §4.1.

Le ViewModel utilise un TreeStore (OsmTagsStore) pour récupérer les données depuis un fichier JSON et afficher ces données dans une arborescence. Aucun Model n'est associé à ce TreeStore.

Dans le ViewModel, on définit les data suivantes :

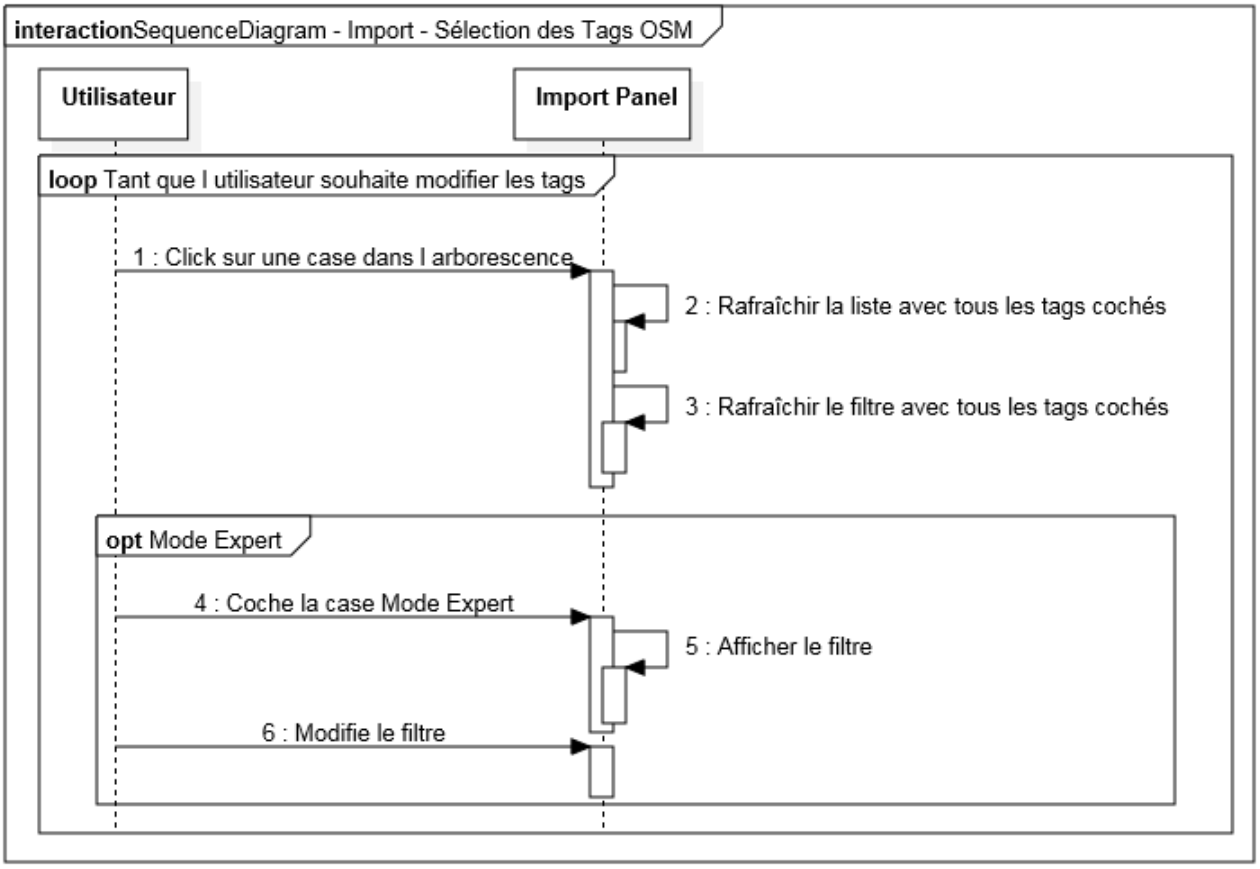
- **data.checkedTags :** liste des tags sélectionnés.

Le ViewController définit quelques méthodes utilisées pour la sélection des Tags OSM :

- **onTreeOsmTagsChange()** : cette méthode est appelée dès qu'une case est cochée ou décochée (event : checkchange). Elle met à jour la liste des Tags sélectionné et le filtre de mode expert
- **checkOsmTags()** : Cette méthode permet de vérifier que la sélection de tags OSM respecte les règles de gestion

Séquence d'actions

Le schéma suivant présent un diagramme de séquence décrivant les actions réalisées pour la sélection des Tags OSM, par l'utilisateur et par l'outil (Import Panel) :



Fichier de configuration

Tous les tags affichés dans l'arborescence sont définit dans un fichier JSON. Ce fichier est enregistré dans le répertoire **resource/data/tagsosm.json**. Les éléments écrits utilisent directement les types de données d'un Tree (côté vue) auxquels on ajoute une donnée « tags ».

Donnée	Type	Obligatoire	Description
text	String	OUI	Nom de la feuille ou du dossier dans l'arborescence
tags	String	NON, seulement pour les feuilles	Tags OSM associés
leaf	Boolean	NON, seulement pour les feuilles	Indique si l'élément est une feuille (true) ou un dossier (false)
checked	Boolean	NON, seulement pour les feuilles	Ajoute une checkbox non cochée à la feuille. Doit toujours être à false
children	[]	NON, seulement pour les dossiers	Contient des données enfants sous le même format (text, tags, leaf/children)

Exemple de fichier JSON associé

```
{
    children : [{
        text: 'Terrains',
        children: [{
            text: 'Résidentiel',
            tag: '[landuse=residential]',
            leaf: true,
            checked: false
        }]
    },{
        text: 'Education',
        children: [{
            text: 'Ecoles Primaires',
            tag: '[amenity=school][“school:FR”=primaire]',
            leaf: true,
            checked: false
        },{
            text: 'Bibliothèques',
            tag: '[amenity=library]',
            leaf: true,
            checked: false
        }
    }]
}
```

Règles de Gestion

Pour la sélection des tags OSM, on utilisera les règles de gestion suivante:

- Au moins 1 tag doit être sélectionné pour l'import
- Expression régulière sur le filtre de tags pour vérifier que les filtres sont écrits correctement (vérifié seulement lorsque le mode expert est activé
    - Les tags sont entre crochets []
    - Les clés-valeurs sont séparés par le signe = ou par != ou bien ~ ou !~ pour les expressions régulières
    - Une clé peut être présente sans valeur
    - Une clé ou une valeur contenant un « deux points » (:) doit être entourée par des guillemets « ». Il en est de même pour les valeurs contenant des accents (Unicode 0xC0 à 0xFF) ou des caractères spéciaux (‘-#).
    - Les valeurs pour expression régulières doivent être entourées de guillemets
    - Dans les expressions régulières, il faut 2 caractères d'échappement dans l'outil (\\) et 4 dans le fichier de configuration (\\\\)
    - Dans les expressions régulières, on peut utiliser le circonflexe (^) pour le début de chaîne et le dollar ($) pour la fin de chaîne
    - Dans les expressions régulières, un point (.) correspond à n'importe quel caractère
    - Dans le cas des expressions régulières, on peut utiliser une recherche en case insensitive en ajoutant « ,i » derrière la valeur (exemple : [clé~“valeur“,i])

4.3 Sélection de la zone géographique

Diagramme de classes

Le diagramme est disponible au §4.1.

Le ViewController définit quelques méthodes utilisées pour la sélection de la zone géographique :

- **onSelectionClick()** : cette méthode est appelée lors d'un click sur le bouton « Sélectionner », le panneau est réduit et l'outil de sélection est préparé avec le type de sélection choisi.
- **checkSelection()** : cette méthode vérifie qu'une sélection est choisie et que celle-ci est correcte.
- **prepareSelector()** : cette méthode initialise l'outil de sélection de zone géographique en fonction de la configuration choisie sur le panneau
- **onSelectionDone()** : cette méthode est appelée lorsque la sélection est terminée pour afficher la sélection. Elle permet également d'obtenir les coordonnées de la sélection

Règles de Gestion

Pour la sélection de la zone géographique, on utilisera les règles de gestion suivantes :

- 1 sélection doit être faite pour exécuter l'import
- Pour la sélection d'une feature, un seul Polygon ou MultiPolygon doit être sélectionné
- Par défaut, le mode de sélection « Rectangle » est sélectionné

4.4 Sélection de la date minimale

Aucune méthode spécifique n'est définie dans le ViewController pour la gestion de la date, car les règles sont définies dans le DatePicker (dans la View).

Règles de Gestion

Pour la sélection de la date minimale, on utilisera les règles de gestion suivantes :

- La date sélectionnée doit être inférieure ou égale à la date du jour
- La date n'est pas obligatoire pour exécuter l'import

4.5 Sélection du rendu d'affichage

Diagramme de classes

Le diagramme est disponible au §4.1.

Le ViewModel de l'import OSM utilise un Store (RenderingsStore) pour récupérer les rendus définis dans un fichier JSON.

Un Model est défini pour les rendus (RenderingsModel). Voici les données de ce modèle :

Donnée	Type	Obligatoire	Description
name	String	OUI	Nom du rendu
fillcolor	String	OUI	Couleur de remplissage des données affichées, au format rgba Exemple : 'rgba(255, 192, 168, 0.3)'
strokecolor	String	OUI	Couleur des bordures des données, au format couleur HTML Exemple : '#00FFDD'

Fichier de configuration

Tous les rendus affichés dans la liste sont définit dans un fichier JSON. Ce fichier est enregistré dans le répertoire **resource/data/renderingsosm.json**. Ce fichier respecte le Model définit dans RenderingsModel.

Exemple de fichier JSON associé :

```
{
	"renderings":[{
		"name": "Rouge",
		"fillcolor": "rgba(255, 255, 255, 0.2)",
		"strokecolor": "#FF0000"
	},{
		"name": "Vert",
		"fillcolor": "rgba(255, 255, 255, 0.2)",
		"strokecolor": "#00FF00"
	},{
		"name": "Bleu",
		"fillcolor": "rgba(255, 255, 255, 0.2)",
		"strokecolor": "#0000FF"
	}]
}
```

Règles de Gestion

Pour la partie sélection du rendu d'affichage, on utilisera les règles de gestion suivantes :

- Validation du rendu à l'aide d'expressions régulières sur les valeurs à appliquer
- Par défaut aucun rendu n'est sélectionné
- En cas d'erreur, un rendu par défaut sera utilisé.

4.6 Import des données

Diagramme de classes

Le diagramme est disponible au §4.1.

Pour l'import des données, les méthodes suivantes sont définies pour le ViewController :

- **onImportClick()** : cette méthode est appelée lors d'un click sur le bouton « Import »
- **onCancelClick()** : cette méthode est appelée lors d'un click sur le bouton « Cancel »
- **checkParams()** : cette méthode contrôle que tous les paramètres pour l'import sont corrects (elle ne contrôle pas les paramètres déjà vérifiés ou bloqués par la vue
- **prepareRequest()** : cette méthode renvoie la requête préparée pour l'appel à l'API OSM en fonction de tous les paramètres définis par l'utilisateur
- **executeRequest()** : cette méthode exécute la requête vers l'API OSM
- **onRequestFinished()** : cette méthode est appelée une fois la requête vers l'API OSM terminée. Elle s'occupe d'afficher les résultats (en cas de succès ou d'erreur).

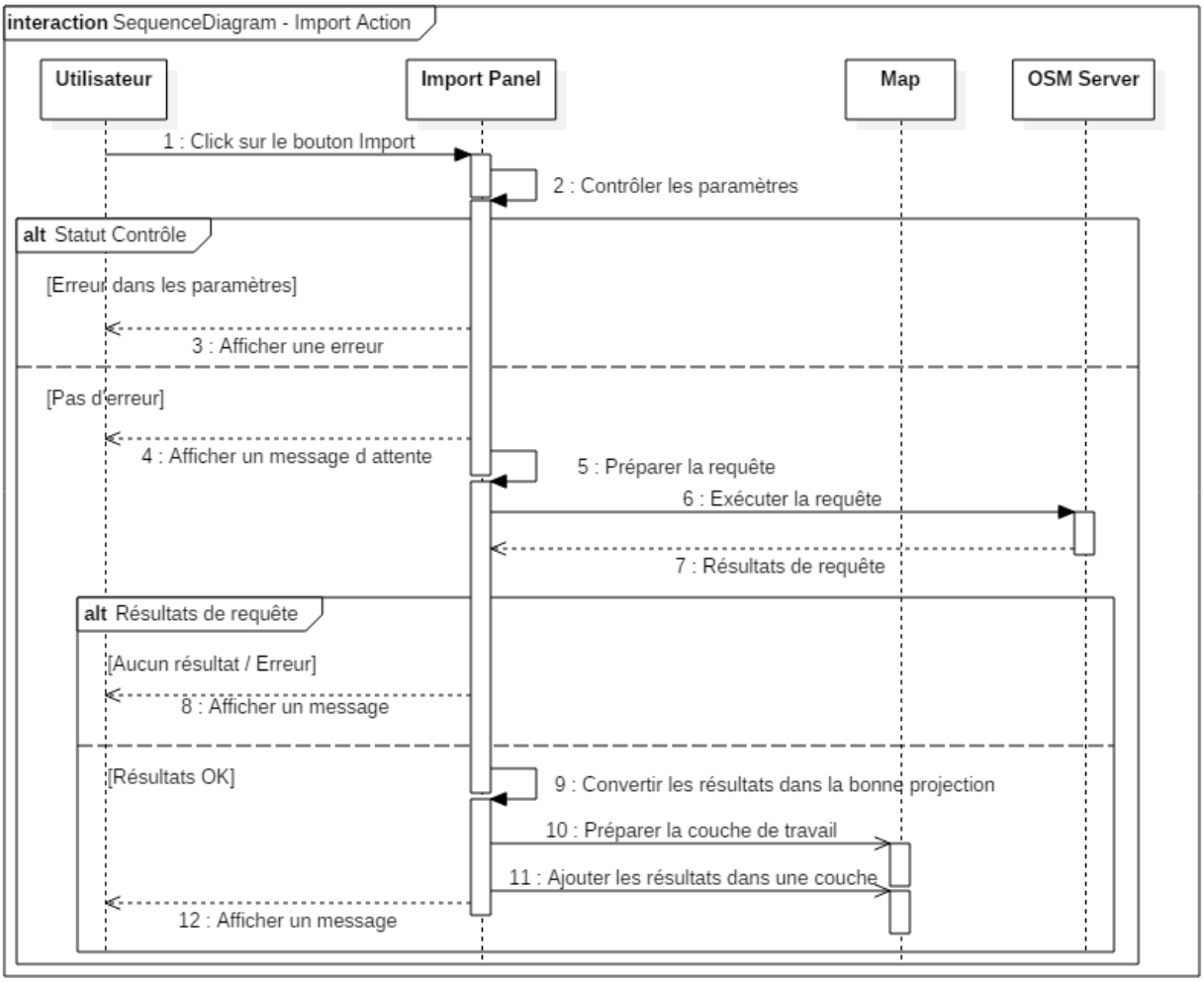
Le Store OsmImportStore définit dans l'Action OsmImport est utilisé pour l'accès à l'API OSM.

Un Model définit les données renvoyées par l'API OSM (OsmImportModel). Quelques méthodes sont définies pour utiliser les données :

- **calculateGeom()** : cette méthode permet de retourner la géométrie (y compris les coordonnées) pour afficher la donnée sur la carte.
- **containsSearchedTags() :** cette méthode permet de savoir si la donnée est bien une donnée recherchée ou une « sous-donnée » (l'API overpass renvoyant également les points de chaque ligne ou tous les éléments d'une relation de la même manière que les éléments recherchés).

Séquence d'actions

Le schéma suivant présente un diagramme de séquence décrivant les actions réalisées pour l'import des données OSM, par l'utilisateur et par l'outil (Import Panel). On peut également voir l'échange entre l'outil et l'API OpenStreetMap ainsi que la gestion de l'affichage dans la carte (Map) :



Règles de gestion

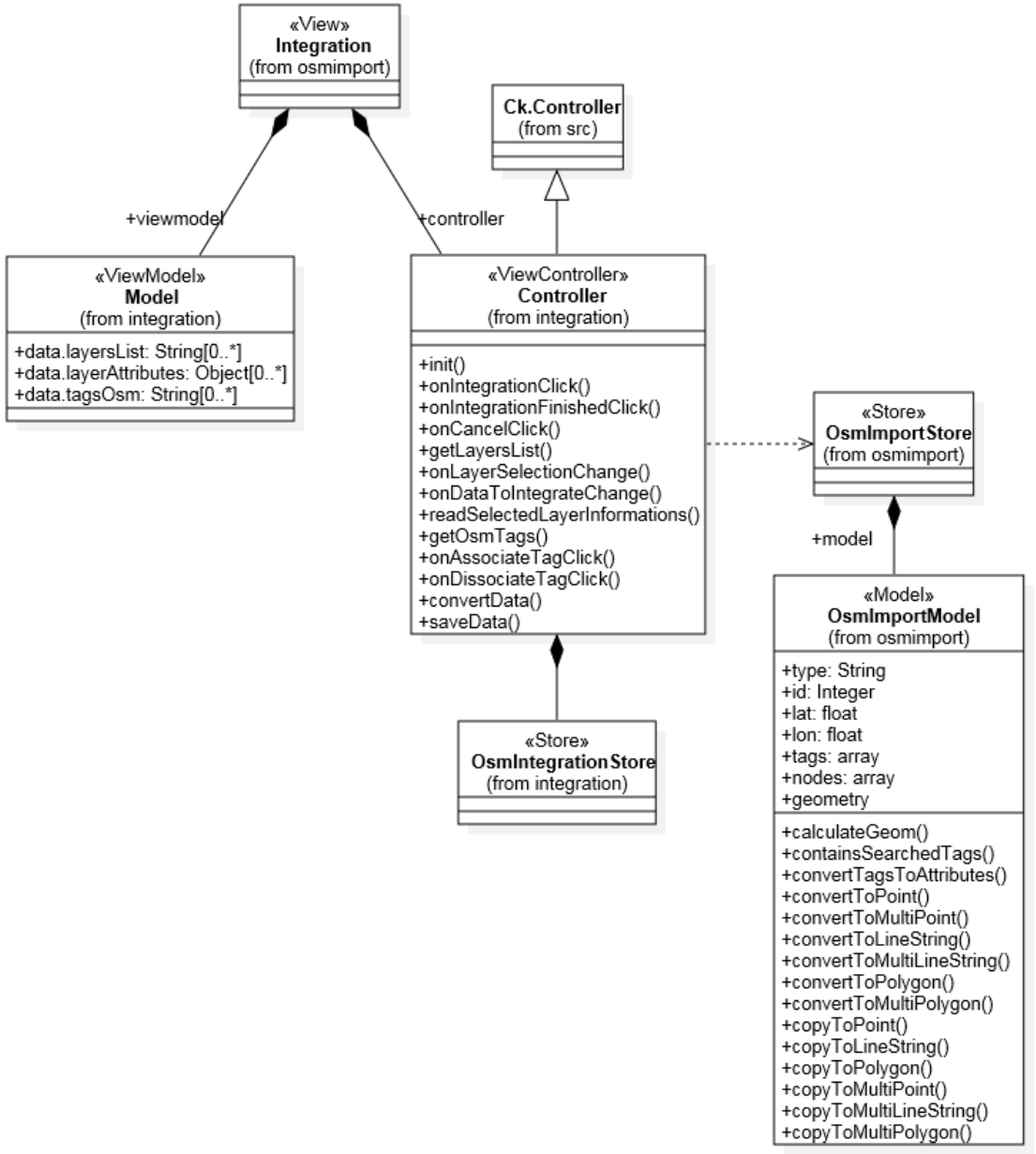
Pour la partie sélection du rendu d'affichage, on utilisera les règles de gestion suivantes :

- Lorsque les données sont importées, le nombre d'éléments insérés dans la couche de travail sera limité à 200 maximum, et ce quelque soit le nombre d'éléments récupérés depuis l'API OSM.

5 Partie 2 : Intégration des données

5.1 Diagramme de classes

Le schéma suivant présente le diagramme des classes utilisé pour la partie intégration des données :



On retrouve le design pattern MVVM associé à un ViewController. Le détail de ce schéma est présenté dans chacune des sous-parties.

Pour les parties communes, la méthode **init()** du ViewController est exécutée lors de l'instanciation de la Vue et de son Controller.

5.2 Sélection de la couche d'intégration

Diagramme de classes

Le diagramme est disponible au §5.1.

Dans le ViewModel, on définit les datas suivantes :

- **data.layersList** : liste de toutes les couches de l'application (ou du contexte) susceptibles d'accueillir les données importées depuis OSM.

Pour la sélection de la couche d'intégration, les méthodes suivantes sont définies pour le ViewController :

- **getLayersList()** : cette méthode est appelée à l'initialisation du panneau et permet d'alimenter la liste des couches susceptibles d'accueillir les données importées depuis OSM.
- **onLayerSelectionChange() :** cette méthode est appelée à chaque fois que la couche sélectionnée est changée. Elle permet de récupérer et d'afficher des informations sur la couche sélectionnée Règles de gestion

Pour la partie sélection de la couche d'intégration, on utilisera les règles de gestion suivantes :

- Par défaut, la première couche de la liste sera sélectionnée

5.3 Sélection du niveau d'informations à intégrer

Diagramme de classes

Le diagramme est disponible au §5.1.

Pour la sélection du niveau d'informations à intégrer, les méthodes suivantes sont définies pour le ViewController :

- **readSelectedLayerInformations() :** cette méthode permet de récupérer auprès du serveur divers informations sur la couche sélectionnée pour l'intégration (attributs, géométrie, projection)
- **getOsmTags() :** cette méthode permet de récupérer la liste des tags de tous les éléments importés depuis OpenStreetMap.
- **onAssociateTagClick() :** cette méthode est appelée lors d'un click sur le bouton avec la flèche vers la gauche. Il permet d'associer le tag OSM sélectionné avec l'attribut sélectionné.
- **onDissociateTagClick() :** cette méthode est appelée lors d'un click sur le bouton avec la flèche vers la droite. Il permet de dissocier un tag OSM de l'attribut sélectionné.

Dans le ViewModel, on définit les data suivantes :

- **data.layerAttributes** : liste de tous les attributs de la couche sélectionnée et tags OSM associés.
- **data.tagsOsm** : liste des tags OSM disponibles pour l'association.

Règles de gestion

Pour la partie sélection du niveau d'information à intégrer, on utilisera les règles de gestion suivantes :

- Pour associer un tag à un attribut, une ligne et une seule de chaque tableau doit être sélectionnée
- Un seul tag peut être associé à chaque attribut
- Un tag ne peut être associé qu'à un seul attribut.
- Pour dissocier les tags des attributs, lors d'un click sur le bouton correspondant, tous les tags de toutes les lignes sélectionnées dans le tableau d'attributs seront dissociés.
- Si aucun tag OSM n'est disponible ou si aucun attribut n'est sélectionné, le bouton « associer » doit être désactivé
- Si aucun tag OSM n'est associé avec un attribut, le bouton « dissocier » doit être désactivé

5.4 Intégration des données

Diagramme de classes

Le diagramme est disponible au §5.1.

Pour l'intégration, les méthodes suivantes sont définies pour le ViewController :

- **onIntegrationClick()** : cette méthode est appelée lors d'un click sur le bouton « Integration ». Elle lance l'intégration des données OSM dans la couche sélectionnée avec la configuration renseignée par l'utilisateur.
- **onIntegrationFinishedClick()** : cette méthode est appelée lors d'un click sur le bouton « Integration Finished ». Elle permet de signifier à l'outil que l'intégration est terminée et que le flux de travail peut reprendre par un import. Un message avertira l'utilisateur que les données non intégrées dans une autre couche seront perdues.
- **onCancelClick()** : cette méthode est appelée lors d'un click sur le bouton « Cancel » et ferme le panneau d'intégration.
- **convertData()** : cette méthode convertit les données importées avec la bonne géométrie et les bons attributs (en fonction de la configuration renseignée par l'utilisateur)
- **saveData()** : cette méthode sauvegarde les données ajoutées dans la couche d'intégration

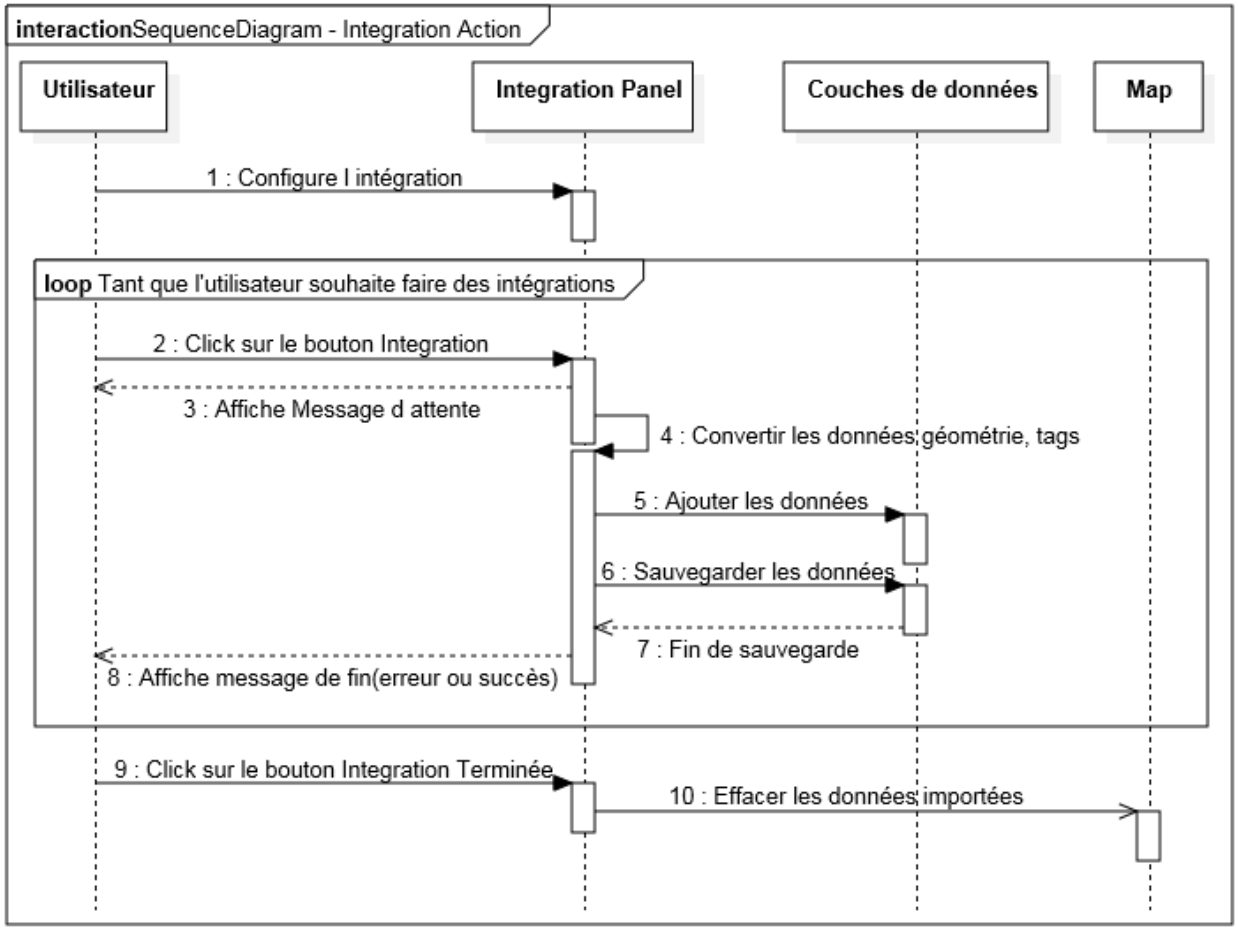
Pour l'intégration, les méthodes suivantes sont définies pour le Model OsmImportModel :

- **convertTagsToAttributes()** : cette méthode permet de composer les attributs pour une donnée en fonction de la configuration renseignée par l'utilisateur
- **convertToPoint()** : cette méthode convertit la géométrie d'une donnée OSM en une géométrie Point intégrée par la suite. Renvoie undefined si la donnée ne doit pas être convertie. **convertToMultiPoint()** : cette méthode convertit la géométrie d'une donnée OSM en une géométrie MultiPoint intégrée par la suite. Renvoie undefined si la donnée ne doit pas être convertie.
- **convertToLineString()** : cette méthode convertit la géométrie d'une donnée OSM en une géométrie Line intégrée par la suite. Renvoie undefined si la donnée ne doit pas être convertie.
- **convertToMultiLineString()** : cette méthode convertit la géométrie d'une donnée OSM en une géométrie MultiLine intégrée par la suite. Renvoie undefined si la donnée ne doit pas être convertie.
- **convertToPolygon()** : cette méthode convertit la géométrie d'une donnée OSM en une géométrie Polygon intégrée par la suite. Renvoie undefined si la donnée ne doit pas être convertie.
- **convertToMultiPolygon()** : cette méthode convertit la géométrie d'une donnée OSM en une géométrie MultiPolygon intégrée par la suite. Renvoie undefined si la donnée ne doit pas être convertie.
- **copyToPoint** : cette méthode retourne la géométrie d'une donnée si celle-ci est compatible avec une géométrie Point. Renvoie undefined si la donnée ne doit pas être copiée. Renvoie un tableau si la donnée est découpée en plusieurs éléments.
- **copyToLineString** : cette méthode retourne la géométrie d'une donnée si celle-ci est compatible avec une géométrie LineString. Renvoie undefined si la donnée ne doit pas être copiée. Renvoie un tableau si la donnée est découpée en plusieurs éléments.
- **copyToPolygon** : cette méthode retourne la géométrie d'une donnée si celle-ci est compatible avec une géométrie Polygon. Renvoie undefined si la donnée ne doit pas être copiée. Renvoie un tableau si la donnée est découpée en plusieurs éléments.
- **copyToMultiPoint** : cette méthode retourne la géométrie d'une donnée si celle-ci est compatible avec une géométrie MultiPoint. Renvoie undefined si la donnée ne doit pas être copiée. Renvoie un tableau si la donnée est découpée en plusieurs éléments.
- **copyToMultiLineString** : cette méthode retourne la géométrie d'une donnée si celle-ci est compatible avec une géométrie MultiLineString. Renvoie undefined si la donnée ne doit pas être copiée. Renvoie un tableau si la donnée est découpée en plusieurs éléments.
- **copyToMultiPolygon** : cette méthode retourne la géométrie d'une donnée si celle-ci est compatible avec une géométrie MultiPolygon. Renvoie undefined si la donnée ne doit pas être copiée. Renvoie un tableau si la donnée est découpée en plusieurs éléments.

Pour l'intégration, le store OsmIntegrationStore est utilisé pour transférer les données au serveur.

Séquence d’actions

Le schéma suivant présente un diagramme de séquence décrivant les actions réalisées pour l’intégration des données OSM, par l’utilisateur et par l’outil (Integration Panel).



Conversion des données

Les données importées depuis OSM peuvent être très variables ; on ne peut donc pas appliquer un schéma de conversion pour chaque type de donnée. Il faut également prendre en compte le fait que toutes les données OSM ne sont pas parfaitement renseignées.

Dans le cas où l'utilisateur choisit d'intégrer toutes les géométries, on respectera le tableau de conversion suivant :

Type de données OSM / **Géométrie couche d'intégration**	**Node**	**Way non fermé** (Chemin)	**Way fermé** (Surface)	**Relation**
Point	Copie simple	Pas de copie	Création d'un Point sur le centre de la boundingbox	Création d'un Point sur le centre de la boundingbox
LineString (Plusieurs points reliés)	Pas de copie	Copie simple	Pas de copie	Pas de copie
Polygon (minimum 3 points formant une surface)	Création d'un Polygon étant un carré placé sur ce node	Pas de copie	Copie simple	Création d'un Polygon sur la boundingbox
MultiPoint (plusieurs Point non reliés)	Création d'un MultiPoint ne contenant qu'un seul Point	Pas de copie	Création d'un MultiPoint ne contenant qu'un seul Point, ce dernier étant le centre de la boundingbox	Récupération de tous les Points ayant un rôle dans la relation, de tous les way fermés créés comme des Point sur le centre de la boundingbox correspondante, regroupés dans un MultiPoint par relation Pas de copie des way non fermés de la relation
MultiLineString (plusieurs Line généralement reliées)	Pas de copie	Copie simple	Pas de copie	Récupération de tous les ways non fermés de la relation, regroupés dans un MultiLineString par relation On ne copie pas les autres éléments (node, way fermés)
MultiPolygon (plusieurs Polygon)	Création d'un MultiPolygon avec un seul Polygon (carré placé sur ce node)	Pas de copie	Création d'un MultiPolygon avec un seul Polygon	Récupération de tous les Polygon, de tous les points convertis en Polygon, regroupés dans un MultiPolygon par relation

Note : la boundingbox est le rectangle dans lequel la figure est directement inscrite (de la latitude minimale à la latitude maximale et de la longitude minimale à la longitude maximale)

Correspondance des types de géométrie

Dans le cas où l'utilisateur choisit de n'intégrer que les éléments importés dont la géométrie correspond à couche d'intégration sélectionnée, on utilisera la correspondance définie dans le tableau suivant :

Type de géométrie de la couche	**Type de données OSM intégrée**
Point	Node Tous les node issus de Relation
LineString	Way non fermés Tous les ways non fermés issus de Relation
Polygon	Way fermés Tous les ways fermé issus de Relation
MultiPoint	Node, chacun dans un MultiPoint différent Tous les node issus de Relation, chaque relation formant un MultiPoint différent
MultiLineString	Way non fermés, chacun dans un MultiLineString différent Tous les ways non fermés issus de Relation, chaque relation formant un MultiLineString différent
MultiPolygon	Way fermés, chacun dans un MultiPolygon différent Tous les ways fermés issus de Relation, chaque relation formant un MultiPolygon différent

Règles de gestion

Pour la partie intégration, on utilisera les règles de gestion suivantes :

- Une fois l'intégration terminée (fin signifiée par l'utilisateur), toutes les données importées depuis OSM ne devront plus être affichées dans la couche d'import, de même que la zone de sélection.