MOVING Modification BackOffice

*Spécifications*

Ce document synthétise les spécifications logicielles relatives aux changements à mettre en œuvre dans le BackOffice.

Benjamin Verney – Thomas Badin

06/05/2014

MOVING Modification BackOffice

Spécifications

Les besoins exprimés pour le (les) fronts offices demandent certaines modifications dans le BackOffice. Les spécifications de ces changements sont consignées dans ce document. Les changements dans le back offices se feront directement en modifiant le projet [Zend Framework 2](http://framework.zend.com) (projet hébergé au 6 mai 2014 sur [Github](https://github.com/thbadin/cervin.git)).

Table des matières

[Authentification via les API 2](#_Toc387225010)

[Correspondance Tag 🡺 Elément 3](#_Toc387225011)

[Correspondance Tag (non GPS) 🡺 Eléments 4](#_Toc387225012)

[Correspondance GPS 🡺 Eléments 5](#_Toc387225013)

# Authentification via les API

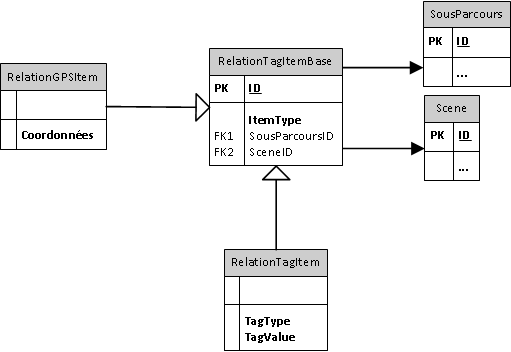
Priorité : 1 Très haute

Le BackOffice doit être modifié afin de stocker des informations d’authentification pour les API.

(Conception et dev à réaliser).

# Correspondance Tag 🡺 Elément

Pour les besoins de l’application mobile, il sera nécessaire de rajouter un mécanisme de correspondance entre l’ID d’un Tag (NFC, QRCode, iBeacon) ou d’une coordonnée GPS et d’un élément. Pour cela nous avons besoin d’étendre la base de données de la manière suivante :



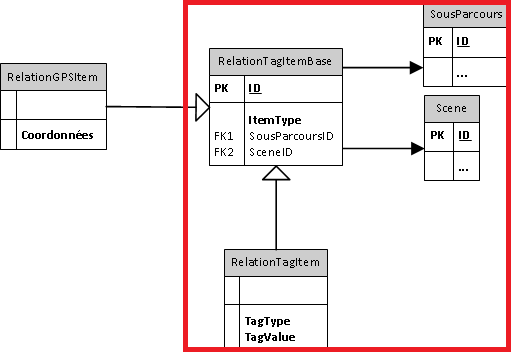
Le fonctionnement sera simple : par exemple l’application demandera au Back Office via un Web Services (cf document de conception des WebServices) a quoi correspond cet Identifiant NFC. Le WebService lui répondra qu’il correspond à la Scène (ou au Sous parcours) avec tel Identifiant de base de donnée.

La complexité entre la relation Tag 🡺 Eléments et position GPS 🡺 Eléments étant vraiment différentes nous avons choisi de séparer les deux développements.

## Correspondance Tag (non GPS) 🡺 Eléments

Priorité : 5 Moyenne

Le back office doit être modifié afin de stocker la correspondance ID tag (NFC, QRCode, iBeacon) et un point d’intérêt. Un point d’intérêt est soit une scène, soit un sous parcours. Cette partie représentante la portion suivante du diagramme :



Descriptions des attributs :

|  |  |  |  |
| --- | --- | --- | --- |
| Table | Attributs | Type | Description |
| RelationTagItemBase | ID | Automatique | Entier auto incrémenté |
| RelationTagItemBase | ItemType | Enumération | 2 valeurs possible : SousParcours / Scene |
| RelationTagItemBase | SousParcoursID | Entier (FK) | Si ItemType == SousParcours : Id du sous parcours ; Sinon : null |
| RelationTagItemBase | SceneID | Entier (FK) | Si ItemType == Scene : Id de la scène; Sinon : null |
| RelationTagItem | TagType | Enumération | 3 valeurs possibles : NFC/QRCode/iBeacon |
| RelationTagItem | TagValue | String | Valeur du Tag (Id de la puce NFC par exemple) |

La base de donnée et Doctrine (l’ORM utilisé dans ZF2) devra être étendu, en respectant la logique du schéma ci-dessus.

Une interface d’administration devra être rajoutée à Moving BO, permettant d’ajouter / supprimer / modifier une correspondance.

Pour l’ajout et la modification, l’interface devra comporter les éléments suivants :

* Le choix du type de Tag (via une liste déroulante par exemple). Dans cette première version la liste proposera NFC / QRCode / iBeacon
* La valeur du Tag (TextBox classique, valeur non nulle)
* Le choix entre Scène et SousParcours. Une fois le choix Scène / SousParcours fait, l’interface proposera une interface de selection de Scène / de SousParcours. (Avec un champs de recherche)

Les validations suivantes devront être effectuées :

* La valeur du tag doit être alphanumérique (sans espace), et non null
* Le couple Tag Type / Tag Valeur doit être unique
* Si ItemType correspond à Scene, SceneID doit être non nul
* Si ItemType correspond à SousParcours, SousParcoursId doit être non nul
* L’intégrité référentielle doit être respectée

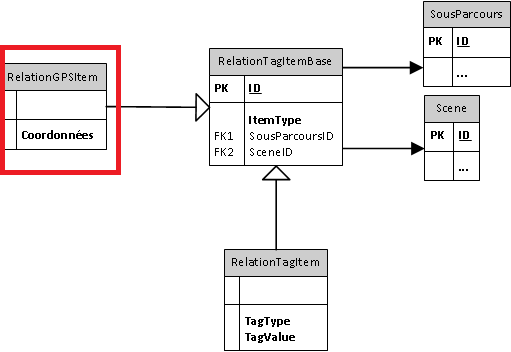
## Correspondance GPS 🡺 Eléments

Priorité : 7 Basse

Cette partie est assez complexe et utilise des technologies peu courantes. N’étant pas un expert en Système d’Information Géographique, et cette fonctionnalité n’étant pas primordiale pour la première version, les spécifications ne seront pas très poussé, et la conception assez libre. (Il faudra a priori tester différentes technologies).

Pour stocker les coordonnées GPS, il faudra se baser sur l’extension GIS de MySQL (<http://dev.mysql.com/doc/refman/5.0/fr/gis-introduction.html>). Pour doctrine, une extension existe (<https://github.com/tvogt/doctrine2-spatial/>), à tester !

La base de données devra être étendue afin de stocker les données géographiques :



La table RelationGPSItem (héritant de RelationTagItemBase) stockera une coordonnée au format [Polygon](http://dev.mysql.com/doc/refman/5.0/fr/gis-class-polygon.html). Le Polygon correspondra à une partie de carte liée à un élément. Exemple : Une scène décrit l’histoire de l’Inovallée. On stockera dans la table de correspondance un polygon correspondant au contour de l’Inovallée dans la base de données, et on le liera à la scène décrivant l’histoire de l’Inovallée. Lorsque l’utilisateur entrera dans le polygon représentant l’Inovallée, l’application lui proposera de consulter la scène associée.

Note : On pourra remplacer l’utilisation des [Polygon](http://dev.mysql.com/doc/refman/5.0/fr/gis-class-polygon.html) par des [MultiPolygon](http://dev.mysql.com/doc/refman/5.0/fr/gis-class-multipolygon.html) afin de permettre d’associer directement plusieurs régions à un élément. (Plus complexe à gérer).

La principale difficulté se trouve au niveau de l’interface d’administration. Après avoir choisi un Tag de type GPS (a rajouter dans les choix), l’utilisateur devra définir le polygon. Il faudra donc développer une interface de sélection géographique (regarder du coté d’OpenStreetMap).