Projet CICEROW

***C****artographie* ***I****nformatique et* ***C****hronologie d’****E****vénements* ***R****eliés* ***O****btenus via* ***W****ikipédia*



Simon Fabry

Manon Girard

Adélaïde Lorieux

Valentin Sasyan

Table des matières

[I. Présentation du sujet 1](#_Toc418693614)

[I.1 Cahier des charges 1](#_Toc418693615)

[II. Solution proposée 1](#_Toc418693616)

[II.1 Choix techniques et fonctionnels 1](#_Toc418693617)

[II.2 Architecture 1](#_Toc418693618)

[1. Diagramme de cas d’utilisation 1](#_Toc418693619)

[2. Base de données : 1](#_Toc418693620)

[3. Echange des données : 1](#_Toc418693621)

[4. Diagrammes de séquence 1](#_Toc418693622)

[III. Gestion du projet 1](#_Toc418693623)

[IV. Perspectives d’amélioration: 1](#_Toc418693624)

[Annexe : Légalité de la réutilisation des données Wikipédia 1](#_Toc418693625)

[Annexe : Documentation 1](#_Toc418693626)

# Présentation du sujet

Notre projet nous a été commandé par Emmanuel Fritsch.

Il consiste à créer une interface permettant de visualiser spatialement et temporellement des événements historiques appartenant à un thème donné à partir de données Wikipédia : L’utilisateur doit pouvoir choisir un thème historique parmi une liste et obtenir tous les événements géoréférencés de Wikipédia sur une carte et une frise chronologique liés à ce thème.

Les thèmes correspondent soit à des portails de Wikipédia (par exemple [le portail de la Rome Antique](http://fr.wikipedia.org/wiki/Portail:Rome_antique)), soit à une page Wikipédia (par exemple [la page de la Guerre de Trente ans](http://fr.wikipedia.org/wiki/Guerre_de_Trente_Ans)).

## Cahier des charges

Le commanditaire nous a demandé par ordre d’importance de

* **-Créer un modèle de données à stocker**

Ce modèle doit notamment comprendre :

Le nombre de langue dans lequel une page Wikipédia est disponible

La longueur de la page Wikipédia

Le nombre de visites de la page

Et toute information que nous trouverons pertinente

* **-Permettre la visualisation des données avec un viewer et une frise chronologique**

L’utilisation de l’API google TimeMap (<https://code.google.com/p/timemap> ) nous a été recommandée par le commanditaire.

Pour la frise chronologique, l’étendue d’un événement doit pouvoir être modifiée selon son importance.

* **-Créer un « aspirateur » de données**

Langage de programmation:

Nous avons le choix entre du PHP pur, du PHP avec du JavaScript côté client ou du Node.

L'aspirateur doit être récursif : Pour chaque lien, on vérifie que la page appartient au thème (nom du portail) et si oui, on traite la page et chaque lien dans cette page.

Les informations qui ne sont pas géolocalisées doivent tout de même être aspirées pour pouvoir être affichées sur la frise chronologique.

* **-Créer une interface de personnalisation de la carte**

L'utilisateur doit pouvoir choisir les événements qu'il veut voir apparaître sur la carte et sur la frise chronologique, et modifier leur apparence.

* **-Optionnel : descendre en granularité dans l'aspirateur**

Récupération d’informations sur des événements qui n'ont pas de page Wikipédia mais seulement un paragraphe dans une page en rapport avec le thème (Par exemple, dans la page sur la bataille de Verdun, paragraphe sur les combats autour de Douaumont).

Pour chaque étape, des tests unitaires doivent être effectués.

# Solution proposée

## Choix techniques et fonctionnels

**Langages de programmation  utilisés:**

PHP avec du JavaScript côté client

Développement de la Base de données avec mysql et du PHP orienté objet

Notre application est compatible avec PHP 5.5 (et les versions supérieures) et mysql 5.6.17.

**Aspirateur de données**

Pour chaque portail qui correspond à un thème, le parseur analyse toutes les pages liées à ce portail.

Le parseur permet de récupèrer des informations relatives aux pages (comme sa longueur) et des informations contenues dans les infobox (cadre à droite de la page), comme les coordonnées géographiques ou la date de l’événement. Si l’infobox ne contient pas de date, on récupère la première date qui apparaît dans l’article.

Les informations stockées d’une page sont : son identifiant, sa longitude, sa latitude, le type d’infobox (conflit militaire, traité…), le nombre de langues, la longueur de la page, l’année, le mois et le jour du début de l’événement, l’année, le mois et le jour de la fin de l’événement, la date de la dernière mise à jour de la page, la distance au portail. Nous avons décidé ne pas inclure le nombre de visites d’une page dans les informations stockées car le seul nombre de visites que nous avons pu trouver est le nombre de visites par jour qui est très variable selon la date à laquelle les informations sont enregistrées et n’est donc pas pertinent.

Nous avons laissé de côté le point optionnel car associer une date avec un événement au milieu d’une page sans logiciel capable d’analyser un texte de façon intelligente nous paraît très difficile à mettre en place.

**Personnalisation d’une carte**

Possibilité de changer le symbole, la taille, et la couleur d’un événement.

Génération de thèmes (choix d’un symbole, d’une taille et d’une couleur) automatiques selon le type d’infobox.

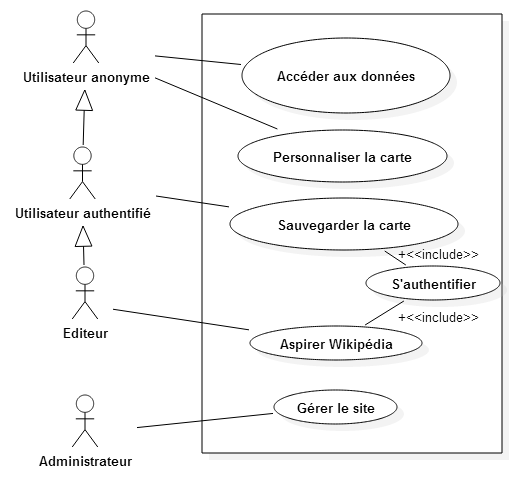
Affichage en petit ou en grand sur la carte d’un événement en fonction de son importance. Nous avons décidé que l’importance d’une page serait calculée selon cette formule :

Pour la calculer, nous utilisons des données récupérées par le parseur telle que la longueur de la page, la distance au portail, et le nombre de langues. Nous avons choisi la formule en faisant en sorte que plus la page est longue, plus l’article est important. De même plus le nombre de langues dans lequel l’article est disponible est important plus la page est importante. L’importance est comprise entre 0 et 1.

L’importance est calculée à l’aide d’une fonction et d’une prodédure SQL. Il aurait été mieux de la calculer dans une vue mais cela donne alors trop de calculs (pour chaque page, il faudrait rechercher le nombre de langues et la longueur maximum dans toutes les pages du portail) et le calcul est trop long (400 résultats en 3 min). L’utilisation de la procédure permet de traiter 7000 résultats en 1,5s.

## Architecture

### Diagramme de cas d’utilisation

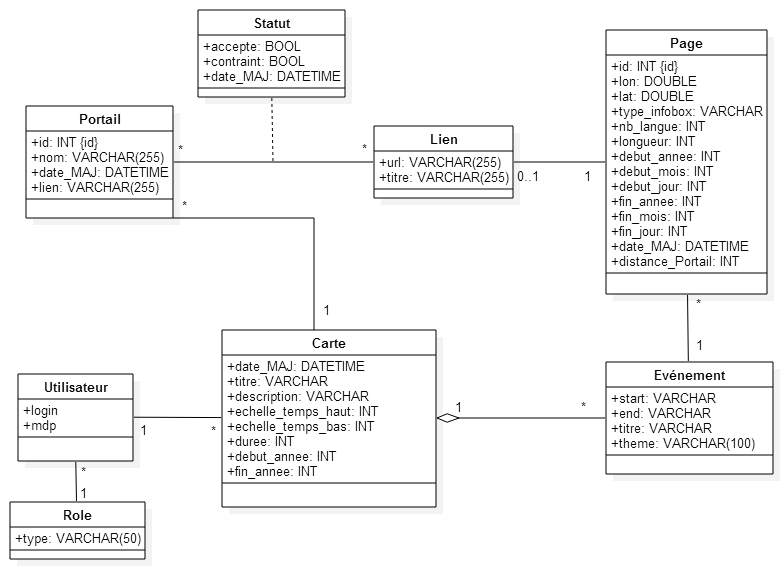


Toutes les données du site (c’est-à-dire les cartes générées accompagnées d’une frise chronologique) sont accessibles par n’importe quel **utilisateur anonyme**. Ces **utilisateurs anonymes** peuvent aussi décider de Personnaliser une carte. Cependant, seuls les **utilisateurs authentifiés** peuvent Sauvegarder une carte personnalisée.

Les « **éditeurs** » sont des utilisateurs enregistrés privilégiés qui peuvent aussi mettre à jour les pages associées à un portail et créer de nouveaux portails (« Aspirer Wikipédia »).

Un **administrateur** gère le côté technique du site. Son rôle se différencie bien des autres acteurs dont il n’hérite donc pas.

### Base de données :



La BDD comporte deux parties à considérer séparément dans un premier temps : une partie « aspiration des données » (objets **Portail**, **Statut**, **Lien** et **Page**) et une partie « personnalisation des cartes » (objets **Utilisateur**, **Carte** et **Evénement**).

#### Partie Aspiration des données :

Un utilisateur « éditeur » va dans un premier temps créer un **Portail** puis ajouter au portail des **Lien**s. Chaque **Lien** est stocké puis considéré ou non comme pertinent. Si le **Lien** est considéré comme pertinent (accepte = true) un objet **Page** est créé comportant les informations supplémentaires à stocker.

L’éditeur peut forcer l’acceptation ou le rejet d’une **Page**.

#### Partie Personnalisation des cartes :

Lorsqu’un utilisateur veut personnaliser une **Carte**, l’application crée dans un premier temps l’objet **Carte**, associé à un **Portail**. Puis toutes les **Page**s associées au **Portail** de la **Carte** sont recherchées et un **Evénement** associé à chaque **Page** est créé. L’utilisateur peut choisir de ne créer que les événements qui se sont déroulés dans un intervalle de temps qu’il définit et qui est stocké dans la base de données grâce aux attributs « debut\_annee » et « fin\_annee ».

L’utilisateur peut ensuite modifier ou supprimer les **Evénement**s pour personnaliser la carte. Les données de sémiologie de chaque événement (taille, couleur, symbole) sont regroupées dans l’attribut « theme ». Les attributs « start » et « end » permettent de modifier la durée d’affichage sur la carte d’un événement.

### Echange des données :

De nombreuses données sont échangées entre JavaScript et PHP au moyen de la notation JSON :

#### Echanges JSON__EditEchanges pour la partie Edition des portails :

Le programme peut échanger au choix :

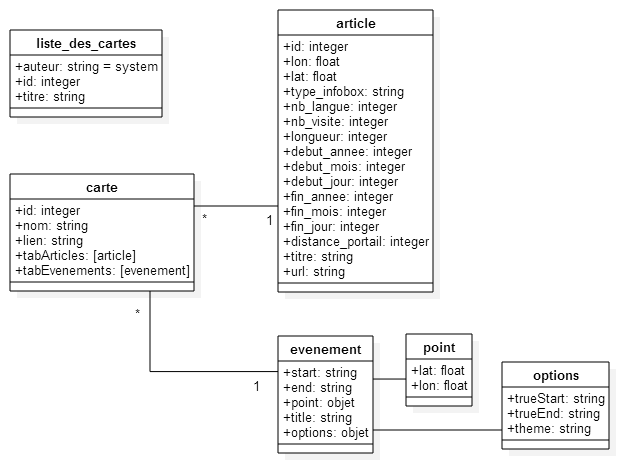
* Une liste de portails, c’est-à-dire un tableau « liste\_des\_portails », contenant comme informations le **nom** du portail, la date de la dernière mise à jour (**date\_MAJ**), et le lien vers le portail.
* Une liste des articles à ajouter, c’est-à-dire un tableau « liste\_des\_articles\_a\_ajouter », contenant toutes les informations nécessaires pour compléter l’objet « Page » dans la BDD.

#### Echange pour la partie Viewer :

Le programme peut échanger au choix :

* La liste des cartes disponibles, c’est-à-dire le tableau « **liste\_des\_cartes** », contenant l’identifiant, le nom et l’auteur de chaque carte. Pour chaque portail, est générée automatiquement et dynamiquement une « carte auto », référencée comme ayant pour auteur « systeme ».
* Une **carte**, c’est-à-dire un objet reprenant les informations de l’objet **Carte** de la BDD et un **tableau** :
  + d’événements (**tabEvenements**) lors qu’il s’agit d’une carte personnalisées.
  + d’articles (**tabArticles**) lorsqu’il s’agit d’une « carte auto ».

Le programme transforme côté client les **Articles** en **Evénements** affichables.



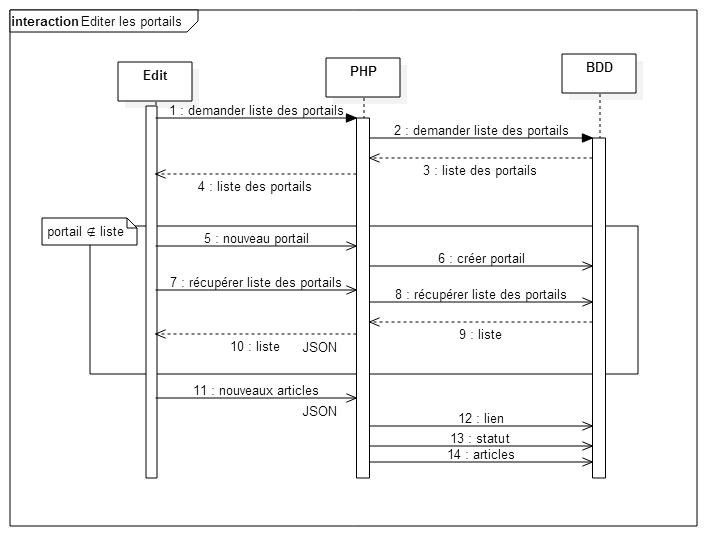
### Diagrammes de séquence

#### Edition des portails :

La création d’une carte s’effectue par un éditeur sur le site web de l’application. Cela est effectué coté client via JavaScript. Dans un premier temps le client demande au serveur la liste des portails. Puis le client tape le nom d’un portail.

Si le portail est inexistant (c'est-à-dire pas dans la liste), on demande au client s’il veut créer le portail. Dans ce cas-là, le client vérifie que le portail existe sur le site de Wikipédia puis ajoute le portail.

Si le portail existe, le client affiche les informations disponibles sur le portail et les interactions disponibles (notamment la mise à jour).



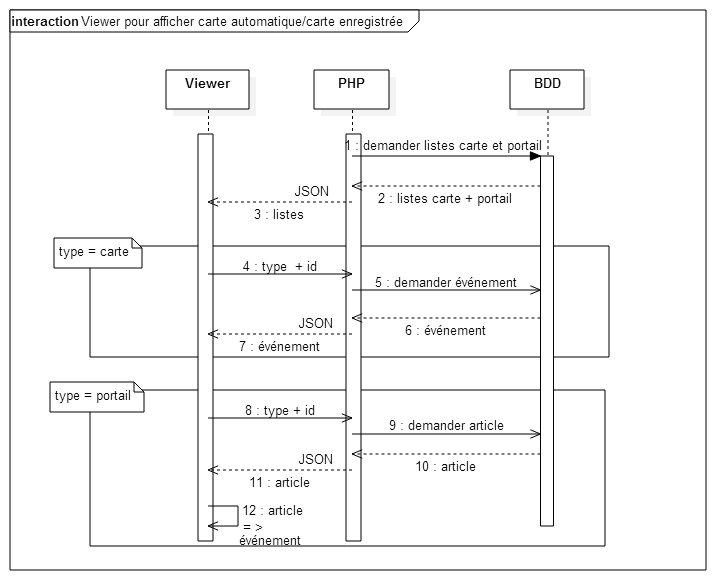
#### Viewer :

Le Viewer est effectué sur le site de l’application. Il est accessible aux utilisateurs et aux visiteurs. Le travail est effectué en JavaScript coté client.

Le client demande dans un premier temps la liste des cartes et des portails (carte « auto »).

L’utilisateur va en choisir un.

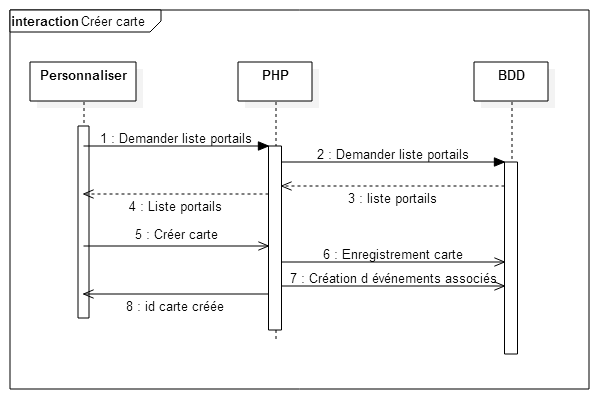
Si c’est un portail qui est choisi (c’est-à-dire une « carte auto ») le client demande au serveur tous les **Article**s associés au portail et les convertis en **Evénement**s avant de les afficher.



#### Création des cartes :

La fonction est traitée côté client en JavaScript.

Le client se connecte au serveur et récupère la liste des portails. L’utilisateur choisi un **Portail** et crée ainsi une carte. Le programme PHP crée alors un **Evénement** associé à la **Carte** pour chaque **Page** associée au **Portail**. Il retourne ensuite l’identifiant (id) de la carte.



#### Chargement d’une carte :

La fonction est traitée côté client en JavaScript.

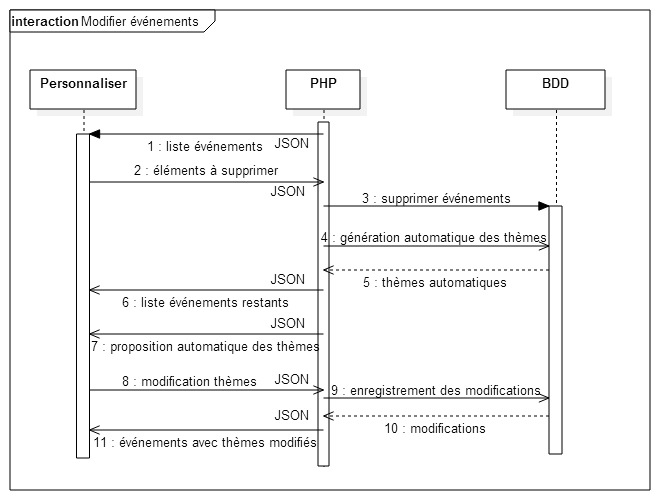
Le client se connecte au serveur et récupère la liste des cartes. L’utilisateur choisi une **Carte**. Le client demande au serveur les informations relatives à la carte et les affiche.



#### Modification des événements liés à une carte :

La fonction est traitée côté client en JavaScript.

Le client se connecte au serveur et récupère la liste des **Evénement**s associés à la **Carte**. L’utilisateur choisit les **Evénement**s à supprimer. Le client renvoie la liste au serveur. Ce dernier effectue une génération automatique des thèmes sur tous les **Evénement**s, puis renvoie la liste des **Evénement**s avec thème au client. Le client les affiche. L’utilisateur peut modifier le thème de chaque **Evénement**. Le client renvoie les informations au serveur.



# Gestion du projet





Diagramme de Gantt du projet



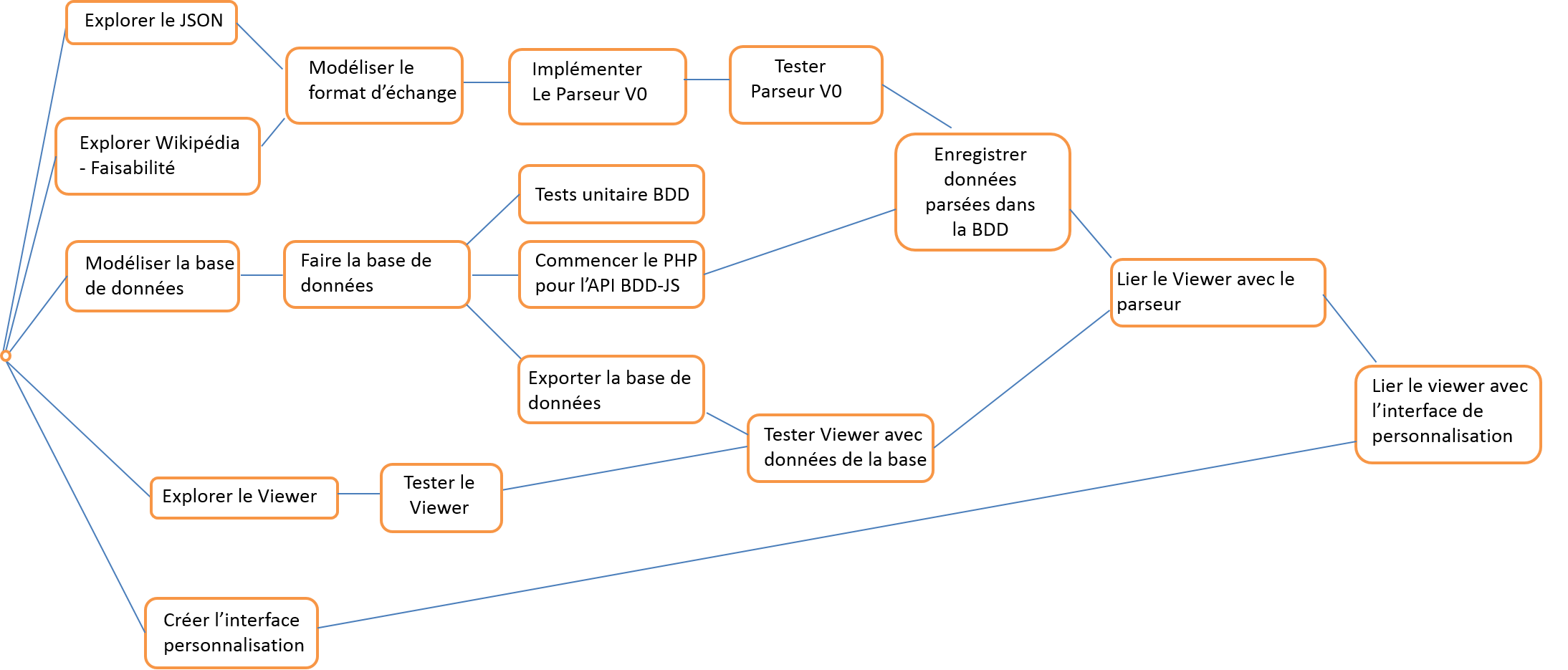


Diagramme Pert : Tâches effectuées

# Perspectives d’amélioration:

**Carte de page**

Notre application ne permet d’afficher des cartes qu’à partir de portails : nous n’avons pas eu le temps d’implémenter une fonctionnalité qui permette de créer une carte à partir des articles associés à une page (comme par exemple celle de la Guerre de Trente Ans).

**Rôle du modérateur**

La base de données est construite de façon à ce que le modérateur puisse à la main supprimer une page ou en ajouter une. Cependant nous n’avons pas eu le temps de développer ces deux fonctionnalités dans l’application.

**Personnalisation et importance**

Nous avions pensé à pouvoir lister les événements d’une carte en fonction de leur importance ou de n’afficher que les X événements les plus importants mais nous n’avons pas eu le temps d’implémenter ces fonctionnalités.

**Amélioration de la fonction pour récupérer une date s’il n’y a pas d’infobox**

Nous avons testé l’exactitude des dates récupérées à l’aide de cette fonction. Pour la Rome Antique, par exemple, l’utilisation de cette fonction permet d’afficher 116 événements en plus. Cependant sur ces 116 événements, seuls 41 (35%) correspondent à la date réelle de l’événement. Les 75 autres événements sont donc affichés dans le Viewer à une mauvaise date.

# Annexe : Légalité de la réutilisation des données Wikipédia

Tous les articles de Wikipédia sont partagés sous la licence « [Licence Creative Attribution - Partage dans les Mêmes Conditions 3.0 non transposé](http://creativecommons.org/licenses/by-sa/3.0/deed.fr) » (CC BY-SA 3.0). Cette licence autorise la réutilisation, la copie et la modification de tout ou partie du texte d’un article.

Cependant, pour une telle utilisation nous devons :

* Indiquer que le contenu utilisé est sous licence CC BY-SA 3.0 ;
* Permettre l’identification des auteurs en donnant une adresse web vers l’article de Wikipédia, ou en donnant une liste des auteurs (paternité) ;
* Indiquer si nous avons modifié le contenu original de Wikipédia ;
* Laisser tous nos travaux sous la même licence.

Dans un premier temps, il est important de noter que **nous ne modifions pas les données de Wikipédia**. Nous les réutilisons en l’état.

De plus, cette licence concerne uniquement le contenu des articles donc pas le code de Wikipédia, ni les logos, images, codes ou autres objets utilisés par la fondation Wikipédia.

Ainsi **nous devons publier les cartes avec frise temporelle générées sous la licence CC BY-SA 3.0**, mais pas les codes de l’application, les logos, etc.

Pour utiliser les données des articles Wikipédia, il nous faut donc :

* préciser que les informations sont utilisée sous la licence CC BY-SA 3.0 ;
* donner un lien vers chaque article utilisé ;
* indiquer que nous n’avons pas modifié les informations ;
* et indiquer que la carte avec frise temporelle est aussi sous licence CC BY-SA 3.0.

# Annexe : Documentation

**Documentation utilisateur** : Notre application étant très intuitive, avec l’accord de notre commanditaire nous n’avons pas fait de documentation utilisateur.

**Documentation installateur** : voir le fichier : Documentation installateur.pdf

**Documentation développeur** : voir le fichier : Documentation développeur.pdf