DUT Informatique - Promotion 2013/2014

PROJET N°7

INTERFACE WEB POUR DES SYSTEMES EXISTANTS DE RECONNAISSANCE D'ENTITES NOMMEES

RAPPORT FINAL DE REALISATION

1/3

Equipe de projet :

CATANESE Antoine – DULON Jeremy – LAGREOU Marc – SABUCO Baptiste

Commanditaires : **M. SALLABERRY Christian**

M. ROYER Albert

Tuteurs : **M. LABORIE Sébastien**

M. MARQUESUZAA Christophe

Remerciements

NOUS TENONS A REMERCIER TOUTES LES PERSONNES QUI NOUS ONT AIDES AU COURS DE NOTRE PROJET ET EN PARTICULIER :

- **M. Christophe Marquesuzaà**, notre tuteur qui s'est rendu disponible tout au long de notre projet afin de répondre à nos questions et nous donner de nombreux conseils avisés.

- **M. Sébastien Laborie**, notre tuteur, pour le temps passé à nous conseiller et à répondre à nos nombreuses questions sur les langages nouveaux pour nous, en particulier le XML et les requêtes XSLT.

- Nos commanditaires, **M. Albert Royer et M. Christian Sallaberry**, pour la richesse du projet proposé et leur disponibilité.

- **M. Thierry Nodenot** pour nous avoir rapidement mis en place un espace de travail sur le serveur Erozate.

- Le groupe de projet de GeoMediaTagger (**Mathieu Capdeville, Corentin David, Léo Dumouch**) pour la documentation ainsi que le code source fourni qui nous a permis de gagner du temps sur nos phases d'analyse et de conception.

- **Mme Corine Ospital**, notre professeur de communication, pour tous ses conseils concernant la rédaction de nos documents, la préparation de l'oral et la gestion de groupe ainsi que sa relecture durant la rédaction.

- **M. Pierre Gastellu**, notre professeur de gestion de projet, pour tous ses conseils sur l'organisation et l'avancement de notre projet ainsi que sa relecture durant la rédaction.

- Nos parents et amis pour avoir accepté de relire et de corriger notre rapport ainsi que de tester notre application.

Table des matières

1 Présentation du projet 4

1.1. Contexte du projet 4

1.2. Enjeux 6

1.3. Fonctionnalités 8

1.4. Contraintes fonctionnelles 9

1.5. Contraintes non fonctionnelles 11

1.6. Limites 12

2. Analyse Fonctionnelle 13

2.1. Diagramme de Cas d'Utilisation (DCU) 13

2.2. Scenarios essentiels détailles 14

2.3. Scenarios réels et maquettes 18

2.4. Déroulement de l'application 21

3. Gestion de projet 22

3.1. Démarche de développement 22

3.2. Recherche et Developpement 23

3.2.1. Étude de faisabilité ; les outils de R.E.N. 23

3.2.2. Le format pivot 25

3.2.3. Les outils de visualisation ; la carte, la frise chronologique et l'affichage des thèmes 27

3.2.4. La mise en page ; Bootstrap 29

3.3. Plannings 30

3.4. Organisation du projet 33

4. Conception et programmation 34

4.1. Diagrammes de séquences 34

4.2. Langages et technologies 37

4.2.1. Langages utilisés 37

4.2.2. Logiciels utilisés 39

4.3. APIs 40
4.3.1. Définition *40*
4.3.2. AlchemyAPI *40*
4.3.3. TextRazor *40*
4.3.4. Google Maps v3 API *41*
4.3.5. SIMILE Timeline *41*
4.3.6. CHAP Link Library *41*
5. Perspectives et Bilan 42
5.1. Bilan 42
5.2. Perspectives 43
6. Abstract 45
7. Glossaire 46
8. Webographie 47
Table des illustrations 48

1 Présentation du projet

1.1. CONTEXTE DU PROJET

Le projet dont nous avons la charge a été proposé par Monsieur Albert ROYER, enseignant-chercheur et maître de conférences en informatique au département Statistique et informatique décisionnelle de l'Institut Universitaire de Technologie (IUT) des Pays de l'Adour et Monsieur Christian SALLABERRY, maître de conférences Habilité à Diriger des Recherches (HDR) en informatique, en poste à l'UFR de Droit, Economie et Gestion de l'Université de Pau et des Pays de l'Adour (UPPA - *Figure 1)*

Figure 1 : Logo de l'UPPA

Ils sont tous deux membres de l'équipe de recherche du T2I au sein du Laboratoire Informatique de l'UPPA (LIUPPA - *Figure 2* - *http://liuppa.univ-pau.fr/live/EquipesdeRecherche/Equipe_T2I/*) et leurs travaux s'inscrivent dans le domaine de la recherche d'informations dans leurs dimensions spatiale, temporelle et thématique.

Figure 2 : Logo du LIUPPA et de l'équipe T2i

Ces recherches d'informations consistent en l'analyse de corpus textuels comme des bibliothèques numériques (fonds documentaires territoriaux, par exemple).

Dans ce cadre, ils ont notamment développé GeoPot, un outil puissant d'analyse lexicale, syntaxique et sémantique du langage écrit afin d'y détecter des Entités Nommées (EN).

Par Entités Nommées nous entendons « une unité textuelle faisant référence à un nom de personne, d'entreprise, de lieu ou encore à une date, une heure ou une unité monétaire repérable ». Par exemple, dans la phrase « Mardi 17 janvier 2014, j'ai rendez-vous avec Mathieu Dupont pour aller voir un match de rugby au nord de Pau », nous pouvons imaginer que cet outil est capable de reconnaître la date, le nom de la personne, le lieu et le thème.

Afin de pouvoir exploiter cet outil de manière simple, rapide et intuitive, un premier site web a été développé par un groupe d'étudiants en 2010-2011, à l'occasion de leur projet de fin de DUT. Ce projet, intitulé GeoText2Map (*Figure 3* - *http://erozate.iutbayonne.univ-pau.fr/geotext2map/*), consistait en la réalisation d'une application web permettant à un utilisateur de renseigner un texte (manuellement, en le téléchargeant ou *via* son URL d'accès) et de le soumettre à cet outil. L'utilisateur récupérait ensuite son texte enrichi de marqueurs indiquant les lieux détectés. La réponse était accompagnée d'une carte sur laquelle on retrouvait les lieux signalés sous forme de puces. L'utilisateur avait également la possibilité d'annoter du texte manuellement en le sélectionnant et en le faisant correspondre à l'endroit de son choix sur la carte.

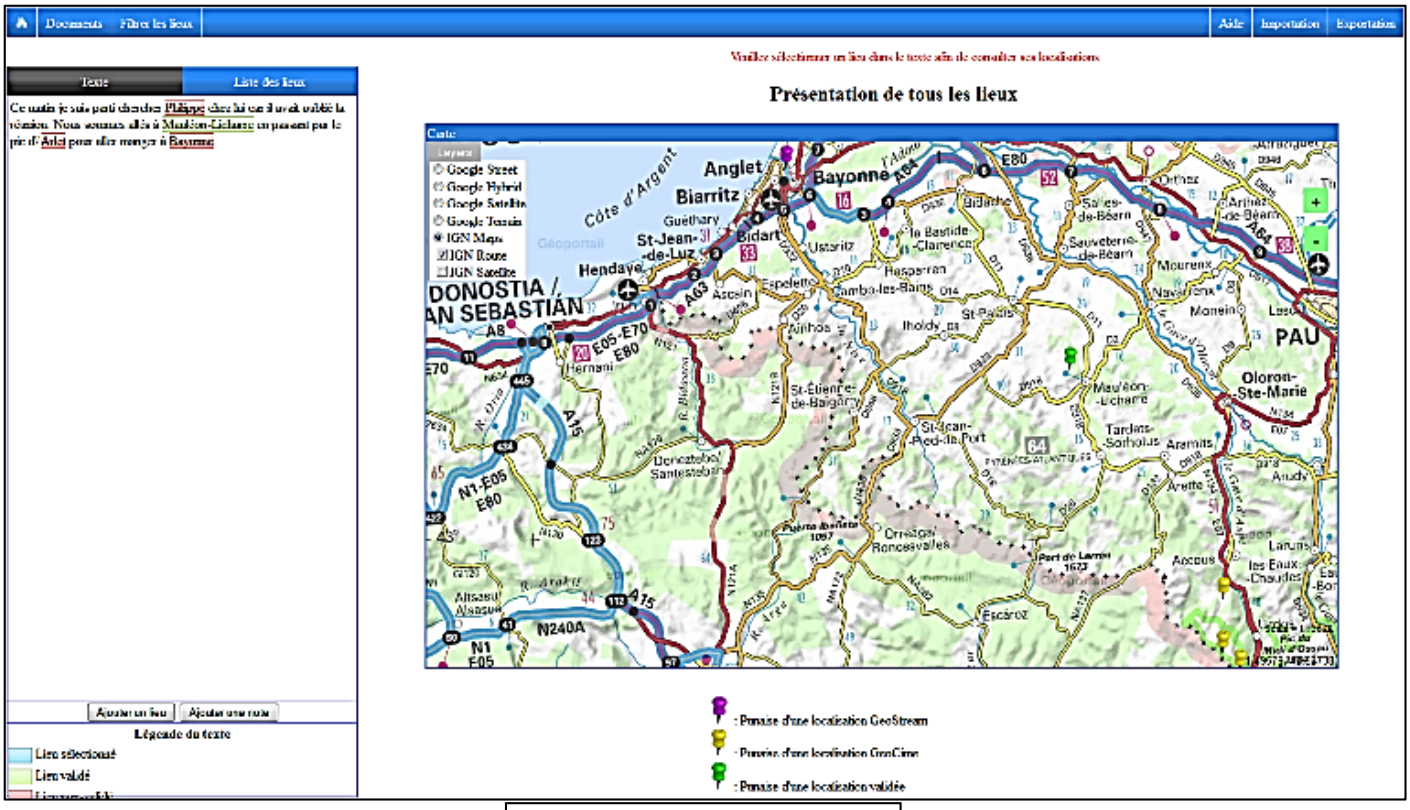

Figure 3 : Copie d'écran GeoText2Map

Par la suite, cette application, toujours existante en l'état, a été enrichie par un nouveau groupe d'étudiants en 2012-2013, également dans le cadre de leur projet de synthèse nommé GeoMediaTagger (*Figure 4* - *http://erozate.iutbayonne.univ-pau.fr/geotext2maptime2013/Stable/*).

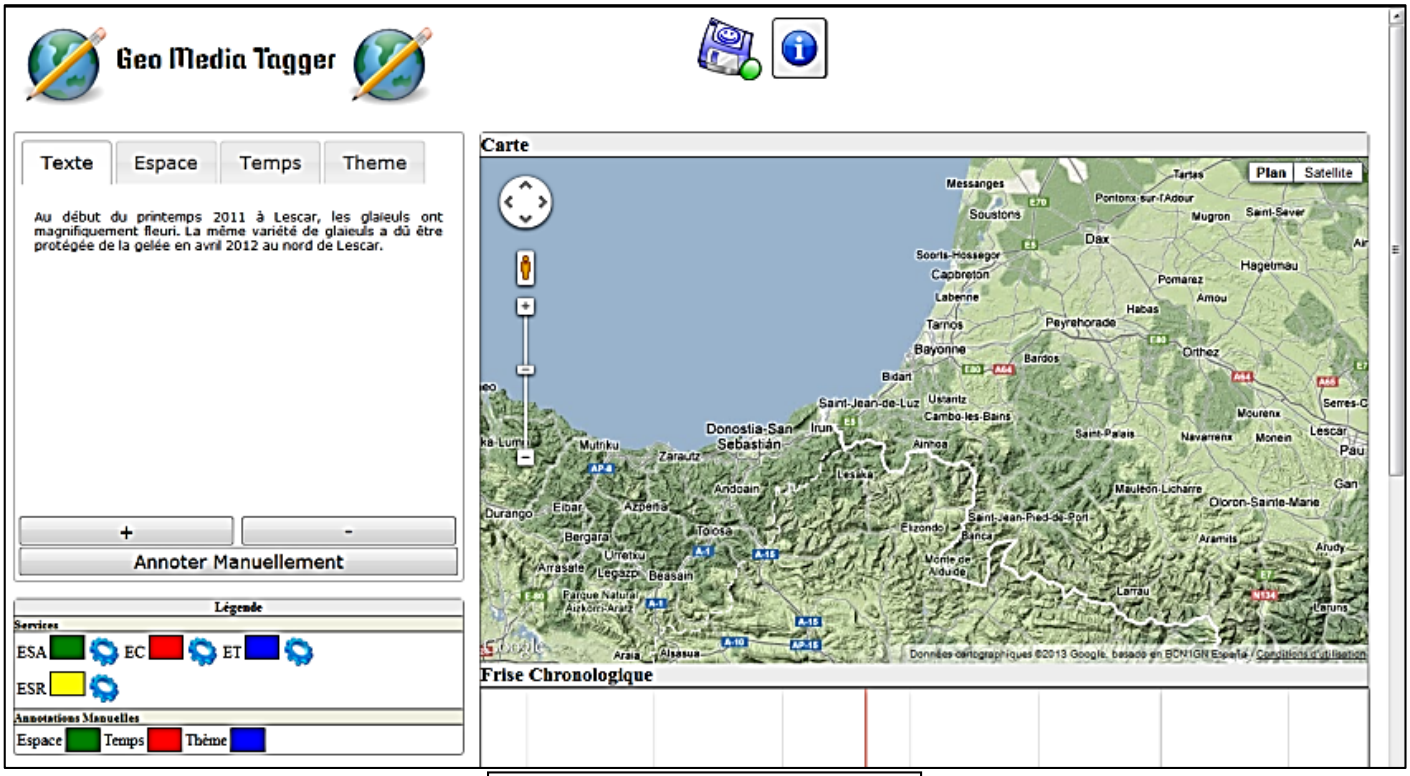

Figure 4 : Copie d'écran Geo media

GeoMediaTagger possède comme nouvelles fonctionnalités de détecter les thèmes et les dates qui sont également représentées sur une frise. Cette version permet en outre d'annoter manuellement non seulement du texte mais également des photos.

Figure 5 : Logo de application Eye Of The Tagger

Suite au développement de ces deux applications et dans une volonté d'aller plus loin dans la conception et l'implémentation d'un outil web de reconnaissance d'entités nommées (REN), nos commanditaires ont exprimé le besoin de disposer d'une application web regroupant plusieurs outils existants de REN, afin de pouvoir comparer leur outil dans le but de l'enrichir et de le perfectionner. C'est donc dans le cadre de notre projet de fin de cycle qu'il nous a été demandé de concevoir et développer une solution à leurs besoins, appelée « Eye Of The Tagger » (*Figure 5*).

Pour ce faire, nous sommes accompagnés de Monsieur Christophe MARQUESUZAA et Monsieur Sébastien LABORIE en qualité de tuteur. Ils tous deux enseignants chercheurs et maîtres de conférences en informatique à l'IUT de Bayonne et du Pays Basque, mais également membres de l'équipe T2i de LIUPPA.

1.2. ENJEUX

Suite à la sélection par nos commanditaires d'outils open source de REN (*Figure 6*) qu'ils considèrent comme étant les plus pertinents pour leurs recherches, ils nous ont demandé de réaliser une application sous la forme d'un site web semblable à GeoMediaTagger, intégrant un ou deux outils parmi ces 9.

AlchemyAPI	CICEROLITE	OpenCalais
http://www.alchemyapi.com/	http://www.languagecomputer.com/	http://www.opencalais.com/
ANNIE	HeidelTime	STANDFORD NER
http://gate.ac.uk/ie/annie.html	http://dbs.ifi.uni-heidelberg.de/	http://nlp.stanford.edu/

Broccoli	NERD	Yahoo!BossPlaceSpotter
http://broccoli.informatik.uni-freiburg.de/	http://nerd.eurecom.fr/	http://developer.yahoo.com/boss/geo/

Figure 6 : tableau d'outils de R.E.N. Open Source

Les enjeux sont dans un premier temps, d'étudier ces 9 outils afin de mettre en évidence le ou les outils (un ou deux) qui répondent le plus aux besoins des commanditaires. Cette sélection se fera donc en fonction de critères de qualité de service et de format de réponse, et sera validée par les commanditaires lors de réunions. Pour les outils retenus, il faudra centraliser leur accès depuis un même site internet et ainsi permettre à l'utilisateur de choisir l'outil avec lequel il veut analyser son texte, puis de récupérer la réponse dans le format spécifique à l'outil choisi.

Dans un second temps, seront mis en place des modules de « transcription » des formats de sortie des différents outils en un format commun, de manière à standardiser la réponse. En d'autres termes, il s'agit de mettre en place pour chaque outil, une méthode de conversion de la réponse dans un format choisi selon les directives de nos commanditaires que nous appellerons « format pivot ».

En effet, chaque outil utilisé, retourne différents formats de réponse (Json, XML, RDF…). Ces dits formats, sont plus précisément des nomenclatures (langages) permettant de structurer (d'organiser) des données tout en faisant apparaître des relations entre ces dernières, à la manière d'une Base de Données (voir « Langage et Technologie » Partie 4.3). Exemple pour le XML, même si deux réponses peuvent contenir exactement les même informations, elles peuvent être organisées différemment et posséder des noms de « rubriques » différentes.

Voir sur les deux exemples de fichier XML ci-dessous (*Figure 7*), que bien que contenant à peu près les mêmes informations (contenu en gras présent entre les balises), certaines apparaissent sur l'une et non sur l'autre, elles ne sont pas organisées (ordonnées) de la même manière, et surtout elles n'ont pas les mêmes noms de « rubrique » (noms des balises différents d'un exemple à l'autre).

```
<Entite>
  <Appellation>France</Appellation>
  <Type>Entité Spatiale</Type>
  <Sous-type>Pays</Sous-type>
  <NbHabitants>660000000</NbHabitants>
  <HabitantParKm2>103,8</HabitantParKm2>
  <Surface>641185</Surface>
</Entite>
```

```
<Entite_Spatiale>
  <Type>Pays</Type>
  <Sous_Type>République</Sous_Type>
  <Monnaie>Euro</Monnaie>
  <Nom>France</Nom>
  <Superficie>641185</Superficie>
  <Population>660000000</Population>
  <Densite>103,8</Densite>
</Entite_Spatiale>
```

Figure 7: Exemples de formats XML

C'est pourquoi, avant de mettre en place une structure d'exploitation de ces résultats, il faut dans un premier temps mettre en place des convertisseurs qui vont transformer chacune de ces réponses en conservant les informations transmises mais en les restructurant toutes au même format afin que l'on puisse les traiter automatiquement. Ce format final commun sera donc notre « format pivot ».

Suite à ce premier traitement nous pourrons développer un module qui récupère des informations au format pivot et les exploite en affichant par exemple, les informations spatiales sur une carte et les temporelles sur une frise chronologique.

Le principe de l'application à produire est donc résumé sur le schéma suivant :

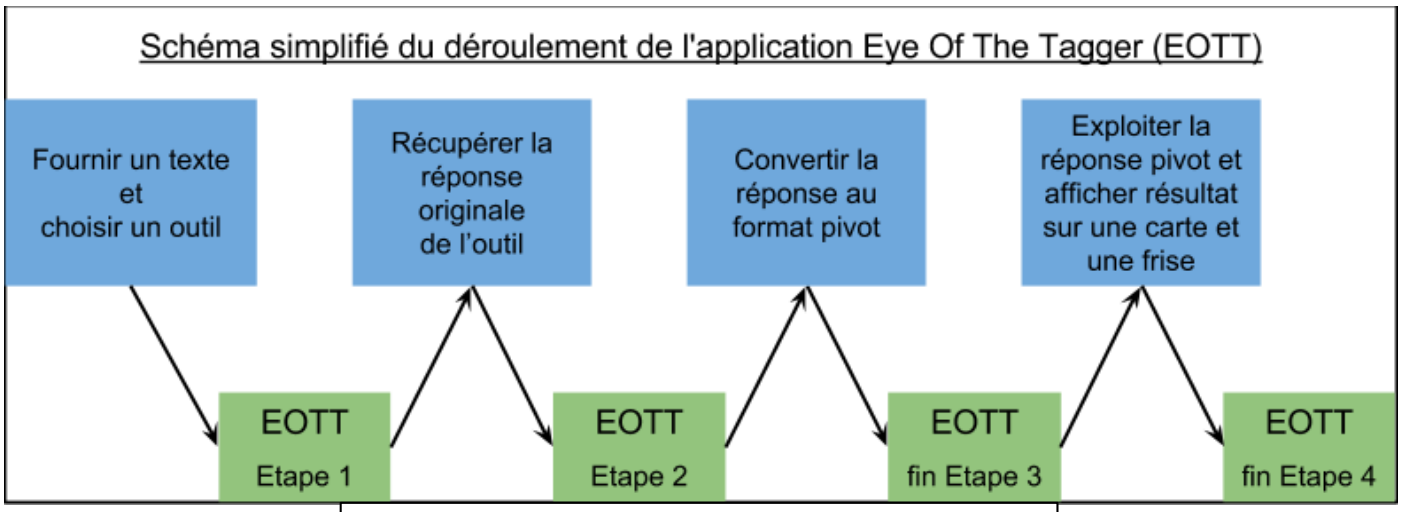

Figure 8 : Schéma simplifié du déroulement général de l'application V1

L'objectif est également de pouvoir envisager dans un futur projet, de combiner la puissance de chacun des outils disponibles en leur soumettant le même texte en même temps et ainsi, fusionner toutes leurs réponses en une seule grâce à ce « format pivot ».

La mise en place d'un tel outil sera l'occasion pour nos commanditaires, de posséder une base solide et fiable d'évaluation et de comparaison de leur propre outil, afin de l'enrichir et le perfectionner efficacement. De nombreux membres de l'équipe du T2I pourraient ainsi intégrer cet outil-maison à leurs propres travaux de recherche.

1.3. FONCTIONNALITES

L'application web que nous devons produire devra se présenter sous la forme d'un site internet accessible depuis n'importe quel ordinateur, fixe ou portable, connecté. Il est important de noter que nous ne gèrerons pas l'affichage sur petits écrans de type tablette ou smartphone.

L'utilisateur pourra soumettre un texte à l'application en le saisissant manuellement, en le téléchargeant ou en fournissant son URL d'accès. Il sera ensuite invité à choisir un outil d'analyse. Une fois analysé, l'application lui retournera son texte dans lequel elle aura surligné toutes les entités détectées (lieux, dates, noms de personnes…).

Elle proposera également à l'utilisateur de pouvoir télécharger la réponse aux formats bruts de l'outil choisi (XML, JSON, RDF…) ou dans le « format pivot », mais également de pouvoir faire apparaître au moyen de puces, sur une carte et une frise, les lieux et dates détectés.

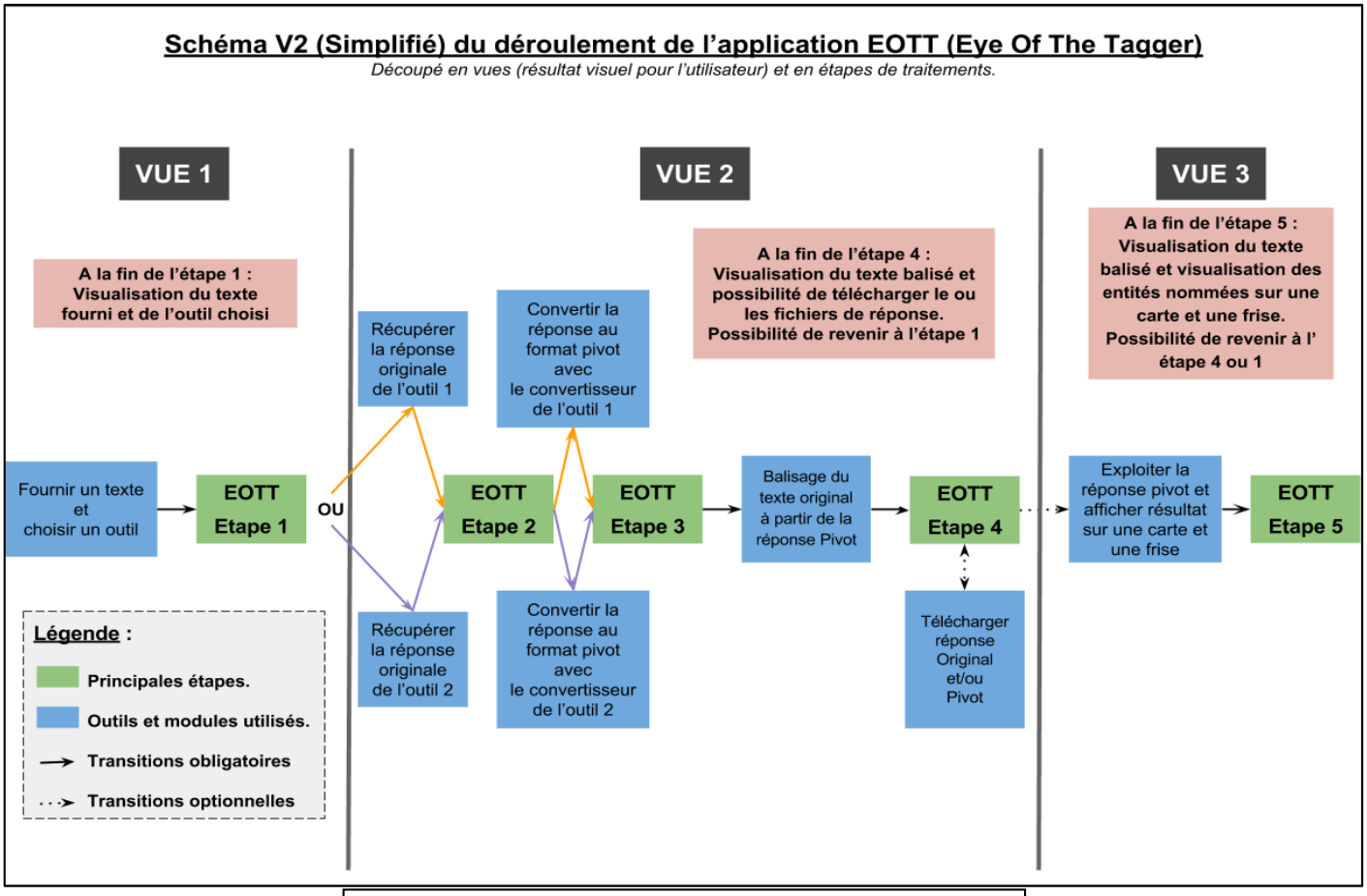

Figure 9 : Schéma simplifié du déroulement général de l'application V2

1.4. CONTRAINTES FONCTIONNELLES

En termes de critère ergonomique, il nous a été précisé que chaque écran (ou vue) de l'application devra tenir sur une page, c'est-à-dire sans avoir à utiliser de barre de défilement.

Il nous a aussi été imposé que l'utilisation de l'application se déroule en trois étapes distinctes (trois écrans différents) pour lesquelles nous avons fait des maquettes.

La première vue (*Figure 11*) contiendra un formulaire à onglets permettant à l'utilisateur de saisir un texte selon trois méthodes : manuellement ou en le téléchargeant, ce depuis sa machine ou une URL. Cette vue sera également l'occasion pour l'utilisateur de choisir un outil (à l'aide de boutons radio) et enfin de lancer l'analyse de son texte avec l'outil choisi. Cette vue correspond à l'étape 1 du schéma de déroulement de l'application (*Figure 9*).

Figure 10 : Première vue de l'application

Le lancement de l'analyse par l'utilisateur déclenche l'enchainement des étapes deux et trois du schéma de déroulement de l'application (*Figure 10*) et conduit à la deuxième vue de l'application (*Figure 11*) qui présente le texte précédemment choisi sur lequel chaque entité est surlignée d'une couleur propre à son type.

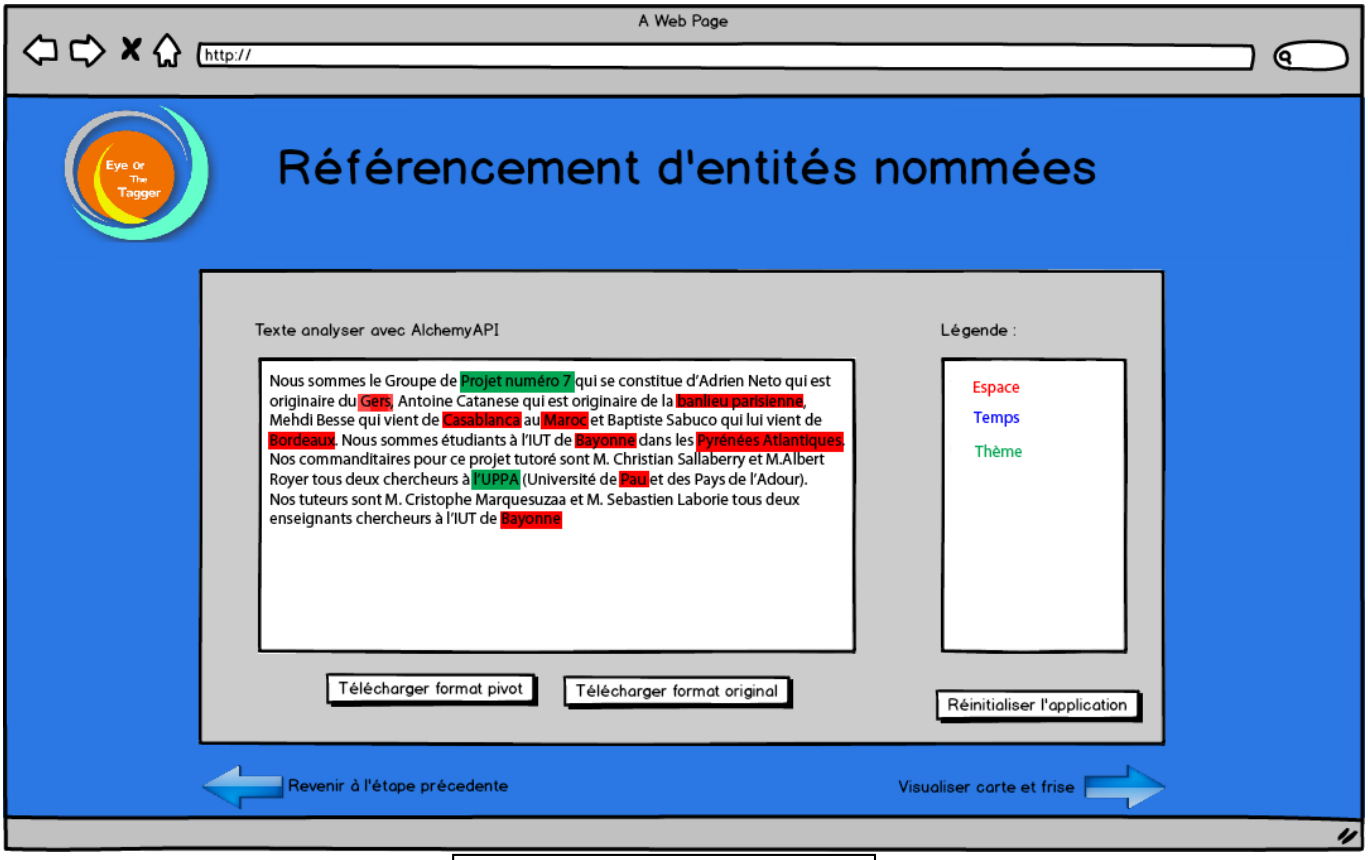

Figure 11 : Deuxième vue de l'application

Cette vue propose également deux liens permettant à l'utilisateur de récupérer son texte analysé au format original ou pivot. Enfin l'utilisateur pourra, à cette étape, demander la visualisation des entités spatiales et temporelles, détectées dans son texte, sur une carte et une frise.

Ce dernier choix conduira au déclenchement de l'étape 4 du schéma de déroulement de l'application (*Figure 9*) qui fournira ainsi la troisième et dernière vue. (*Figure 12*)

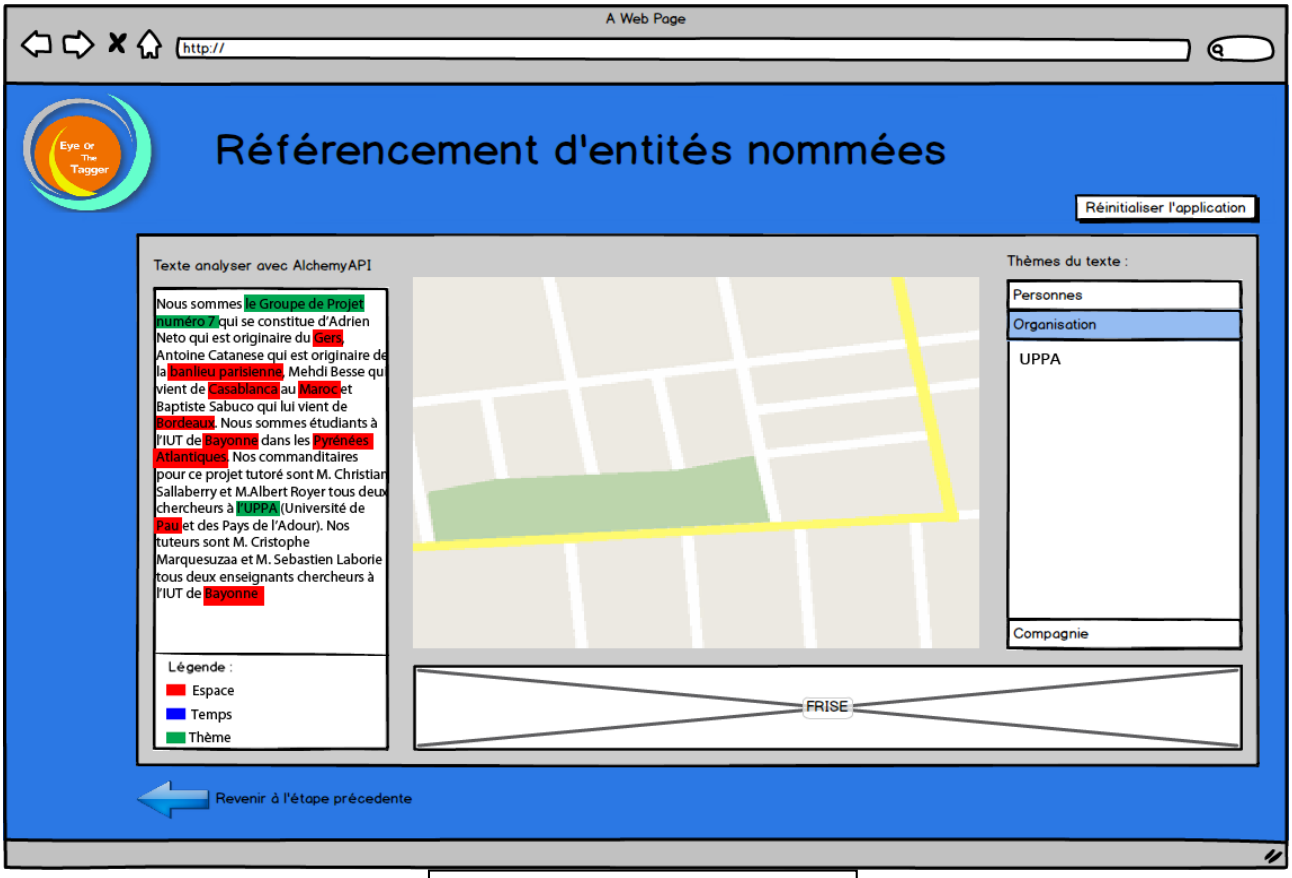

Figure 12 : Troisième vue de l'application

Elle se découpera en quatre zones contenant respectivement le texte surligné de la vue précédente, une carte contenant les entités spatiales détectées, une frise contenant les entités temporelles, enfin une zone de texte contenant le reste des entités détectées triées par thème.

1.5. CONTRAINTES NON FONCTIONNELLES

Afin d'accroître la robustesse de notre application, nous la développerons selon les règles du Modèle Vue Contrôleur (MVC) à l'aide d'un Framework que nous n'avons pour l'heure pas encore déterminé.

En outre, étant donnée l'utilisation par notre application de services web relativement lourds (services de REN, services d'annotation de cartes ou de frises…), il nous a été suggéré d'utiliser de la technologie Ajax pour la développer.

Figure 13 : Logo de l'AJAX

En effet, Ajax est une solution permettant de ne pas rafraîchir la totalité de la page à chaque action demandée par l'utilisateur. Une partie du contrôleur de l'application est déportée côté client, de sorte qu'une action de l'utilisateur ne déclenche que le rafraîchissement de la zone nécessaire. Permettant de fluidifier la navigation de l'utilisateur, lui évitant des temps de chargements lourds, longs et inutiles. Cette technique est possible grâce à la combinaison du JavaScript, de l'XML et de communications asynchrones entre le client et le serveur (d'où AJAX pour Asynchronous Javascript And Xml).

Il est possible d'utiliser un Framework de type JavaScript facilitant l'implémentation en Ajax puisque cette méthode peut sembler relativement complexe au premier abord.

Parmi les Framework JavaScript, nous retenons :

AngularJS	jQuery
http://angularjs.org/	http://jquery.com/
Foundation	Bootstrap
http://foundation.zurb.com/	http://getbootstrap.com/

Figure 14: tableau de Framework MVC utilisant AJAX

1.6. LIMITES

Les limites de notre projet sont établies par le fait que pour développer notre application, nous nous appuyons sur l'utilisation de services web en tant que simples clients. Nous sommes donc entièrement dépendants de ces services ainsi que de leurs performances et qualités de résultats respectives.

Seuls des textes pourront être analysés par notre application, et seule l'intégration de deux outils nous est demandée.

Notre but est de mettre en place une structure qui soit la plus modulaire possible afin de pouvoir y intégrer très simplement de nouveaux outils ou interfaces de visualisation.

2. Analyse Fonctionnelle

2.1. DIAGRAMME DE CAS D'UTILISATION (DCU)

Lors de notre première rencontre avec nos commanditaires, nous avons pu préciser ce que nous avions compris du sujet et ainsi dégager un cas d'utilisation (CU) principal, à savoir faire analyser son texte par l'outil de Reconnaissance d'Entité Nommée (REN) de son choix et en faire ressortir les Entités Nommées (EN) détectées.

En considérant un CU comme un ensemble d'étapes à passer afin de tirer profit d'une des fonctionnalités d'un système, nous avons décidé de détailler notre CU principal par une « inclusion » (CU obligatoire) qui est la saisie de ce texte. En effet, cet étape étant réalisable selon trois manières distinctes, nous avons jugé préférable d'en faire un CU à part. Enfin, nous avons fait apparaître sous forme « d'extensions » (CU optionnels) les possibilités de pouvoir, après l'analyse de son texte, télécharger les réponses générées ou visualiser les EN détectées sur une carte et une frise chronologique. Cette première analyse a donc donné lieu à la première version de notre Diagramme des Cas d'Utilisation (DCU V1 - *Figure 15 - ci-dessous*).

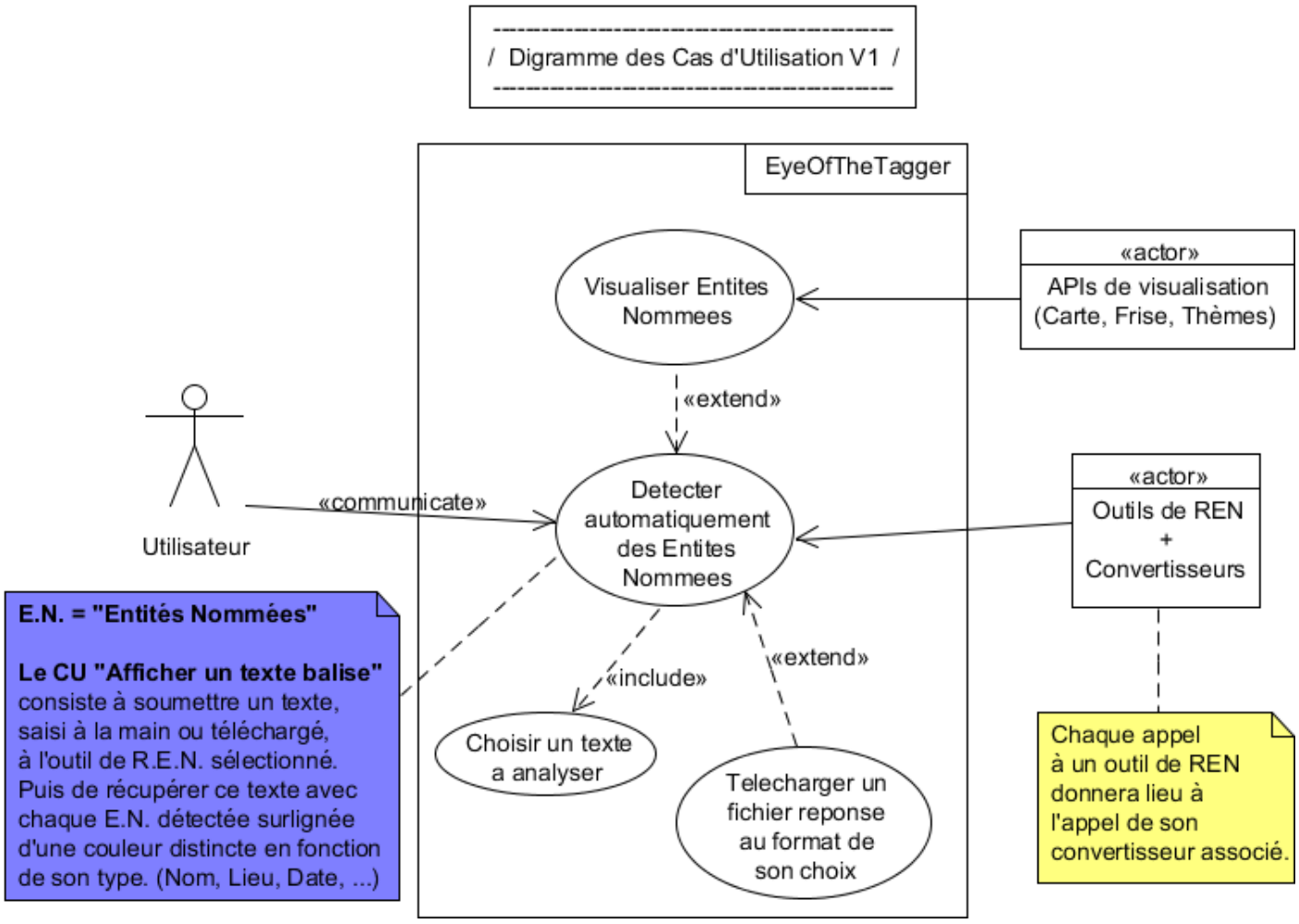

Figure 15 : DCU V1

Suite à cette première proposition, nous avons voulu préciser encore le CU « Choisir un texte à analyser » et ainsi produit une version plus détaillée de notre DCU (DCU V2 - *Figure 16*). Nous avons pour cela commencé par renommer le dit CU en « fournir un texte à analyser », puis nous lui avons ajouté un CU optionnel appelé « Importer un texte » qui se spécialise en deux CU distincts. Ceci a été décidé afin de faire clairement apparaître la possibilité pour l'utilisateur d'importer son texte depuis sa machine ou une URL. Nous avons également décidé de spécialiser le CU « Télécharger un fichier réponse au format de son choix », renommé « Télécharger un fichier réponse », en deux CU pourtant quasiment similaires, afin de bien montrer que lors du traitement d'un texte par l'application, les deux formats de réponse étaient conservés et ainsi disponibles au téléchargement.

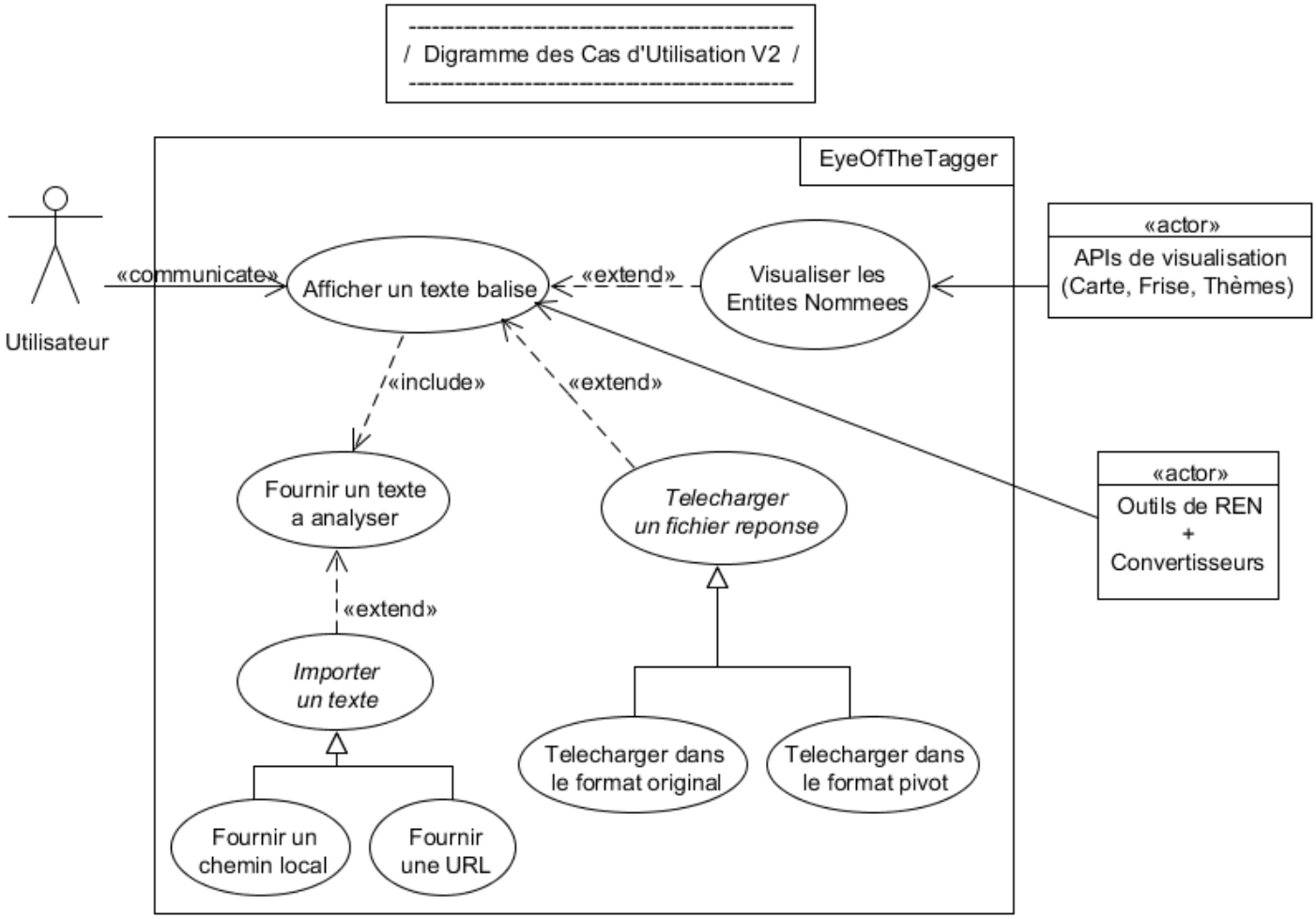

Figure 16 : DCU V2

A partir de cette version finale, nous avons produit les scénarios dérivants chacun des CU identifiés en cherchant à couvrir le plus de cas de figures possibles sans pour autant être exhaustifs.

2.2. SCENARIOS ESSENTIELS DETAILLES

Nous avons suivi les recommandations de nos enseignements d'UML et commencé par produire les scénarios essentiels détaillés de chaque CU (voir annexe – « CDCF »). Ces scénarios ont pour vocation de décrire étape par étape tous les échanges possibles entre l'utilisateur et le système. C'est à cette occasion que l'on a pu concevoir les comportements précis qu'adopte le système pour chaque action de l'utilisateur, notamment en cas de mauvaise utilisation. Voici pour exemple les scénarios essentiels détaillés correspondant au CU « Afficher un texte balise ».

Sommaire d'identification

Titre : Afficher un texte balise **Type :** Essentiel détaillé

Résumé : Après avoir saisi un texte et choisi un outil, l'utilisateur demande au système une analyse de son texte. Le système retourne alors le texte fourni en surlignant chaque entité détectée. Une couleur spécifique est attribuée à chaque type d'entité. L'utilisateur peut ensuite télécharger sa réponse dans le format de son choix ou afficher les entités spatiales et temporelles détectées sur une carte et une frise.

Acteurs : Utilisateur (principal), Outil de R.E.N. (secondaire), Convertisseur (secondaire).

Date de création : 11 janvier 2014 **Date de mise à jour :** -

Version : 1.0 **Responsable :** Antoine Catanese

Description des scénarios

Préconditions : L'utilisateur a saisi un texte sur l'application et a choisi son outil d'analyse.

Scénario nominal

1. Ce cas d'utilisation commence après le CU « Fournir un texte à analyser ».	
2. L'utilisateur lance l'analyse.	3. Le système envoie le texte à l'outil choisi.
4. L'outil récupère le texte et l'analyse.	
5. L'outil renvoie au système le texte analysé dans le format original XML.	6. Le système récupère la réponse originale et l'enregistre sur le serveur dans un fichier ".xml".
	7. Le système convertit la réponse originale au format pivot.
	8. Le système enrichit la réponse convertie afin de la faire correspondre au format pivot en vue de la validation
	9. Le système soumet au validateur la réponse obtenue.
	10. Le système valide le format de la réponse.
	11. Le système enregistre la réponse au format pivot sur le serveur dans un fichier ".xml".

	12. Le système présente le texte de l'utilisateur en surlignant les E.N. détectées dans la réponse au format pivot. Chaque type d'E.N. (Personnes, Lieux, Dates, etc.) est surligné d'une couleur spécifique.
	13. Le système propose le téléchargement de la réponse (au format original ou pivot), d'afficher le texte balisé accompagné d'une carte et d'une frise, de revenir à l'étape précédente ou de réinitialiser l'application.
14. L'utilisateur choisit de visualiser les entités nommées	**Voir scénarios essentiels détaillés du CU " Visualiser les Entités Nommées "**

Post-conditions : -

Enchaînements alternatifs

A1 : Téléchargement de la réponse originale de l'outil choisi

L'enchaînement A1 démarre au point 14 du scénario nominal.

14. L'utilisateur demande le téléchargement de la réponse au format original.

    Voir scénarios essentiels détaillés du CU " Telecharger fichier reponse "

Le scénario nominal reprend au point 12.

A2 : Téléchargement de la réponse de l'outil choisi, convertie au format pivot

L'enchaînement A2 démarre au point 14 du scénario nominal.

14. L'utilisateur demande le téléchargement de la réponse au format pivot.

    Voir scénarios essentiels détaillés du CU " Telecharger fichier reponse "

Le scénario nominal reprend au point 12.

A3 : Retour à l'étape précédente

L'enchaînement A3 démarre au point 14 du scénario nominal.

14. L'utilisateur choisit de revenir à l'étape précédente.
15. Le système affiche à l'utilisateur la vue précédente de saisie de texte contenant le dernier texte qu'il a renseigné.

Le scénario nominal reprend au point 1.

A4 : Après un retour à l'étape précédente, nouveau lancement de l'analyse

L'enchaînement A4 démarre au point 14 du scénario nominal.

14. L'utilisateur choisit de revenir à l'étape précédente.
15. Le système affiche à l'utilisateur la vue précédente de saisie de texte contenant le dernier texte qu'il a renseigné.
16. L'utilisateur lance l'analyse à nouveau avec le même texte.
17. Le système ne refait pas l'analyse et affiche directement les informations collectées précédemment.

Le scénario nominal reprend au point 12.

A5 : Réinitialisation de l'application et retour à l'accueil

L'enchaînement A5 démarre au point 14 du scénario nominal.

14. L'utilisateur choisit de réinitialiser l'application.
15. Le système supprime tout ce qui n'est pas enregistré sur le serveur.
16. Le système affiche la page d'accueil.

Le scénario nominal reprend au point 1 du scénario essentiel détaillé du CU "Fournir un texte a analyser".

Enchaînements d'erreurs/exceptionnels

E1 : Outil de R.E.N. Inaccessible

L'enchaînement E1 démarre au point 4 du scénario nominal.

4. Le système ne parvient pas à se connecter à l'outil de R.E.N.
5. Le système signal à l'utilisateur qu'il n'a pas réussi à se connecter à l'outil de R.E.N. demandé et l'invite à réessayer ou à choisir un autre outil.

Le système reprend au point 1 du scénario essentiel détaillé du CU « Fournir un texte a analyser ».

E2 : Texte non renseigné

L'enchaînement E2 démarre au point 1 du scénario nominal.

1. L'utilisateur choisit de lancer l'analyse sans avoir renseigné de texte à analyser.
2. Le système signal à l'utilisateur qu'il n'a pas saisi de texte et lui propose de réinitialiser l'application ou de revenir en arrière afin de récupérer un éventuel texte renseigné précédemment.
3. L'utilisateur revient à l'étape précédente.
4. Le système affiche le dernier texte saisi.

Le système reprend au point 1 du scénario essentiel détaillé du CU « Fournir un texte a analyser ».

E3 : Conversion au format pivot non validée

L'enchaînement E3 démarre au point 10 du scénario nominal.

10. Le système ne valide pas le format de la réponse.
11. Le système signale à l'utilisateur qu'un problème s'est produit lors de la conversion au format pivot.
12. Le système propose le téléchargement de la réponse au format original, de revenir à l'étape précédente ou de réinitialiser l'application.
13. L'utilisateur revient à l'étape précédente.
14. Le système affiche le dernier texte saisi.

Le système reprend au point 1 du scénario essentiel détaillé du CU « Fournir un texte a analyser ».

Une fois les scénarios essentiels détaillés finalisés, nous avons produit les scénarios réels afin de pouvoir clairement identifier les éléments d'interfaces nécessaires à l'application.

2.3. SCENARIOS REELS ET MAQUETTES

Contrairement aux scénarios essentiels qui décrivent les interactions possibles avec l'application sans se préoccuper du « comment ? », les scénarios réels vont de paire avec des maquettes et permettent de présenter explicitement les moyens utilisés pour manipuler l'application.

En voici un exemple :

Sommaire d'identification

Titre : Fournir un texte a analyser **Type :** Réel

Résumé : Un utilisateur saisie un texte manuellement et choisit un outil.

Acteurs : Utilisateur (principal).

Date de création : 21 février 2014 **Date de mise à jour :** -

Version : 1.0 **Responsable :** Antoine Catanese

Description des scénarios

Préconditions : L'utilisateur est connecté à Internet et possède un navigateur Web.

Scénario nominal

1. L'utilisateur se trouve sur la page d'accueil de l'application.	2. Le système **affiche** un formulaire contenant : • une **zone de saisie**[1] « Texte à analyser » servant à accueillir le texte de l'utilisateur. • Des **boutons radios**[2] « Choix outil » permettant à l'utilisateur de choisir un outil. • un **bouton**[3] « Analyser texte » permettant de lancer l'analyse. • un **onglet**[4] « Saisir URL » permettant à l'utilisateur d'importer un texte depuis une URL. • un **onglet**[5] « Importer texte» permettant à l'utilisateur d'importer un texte depuis sa machine. • Un **onglet**[6] « Saisir manuellement » permettant à l'utilisateur d'accéder à la **zone de saisie**[1] manuelle. • un **logo**[8] « Eye of The Tagger » permettant à l'utilisateur de retourner à la page d'accueil et de réinitialiser l'application. **VOIR VUE 1** Note : Le logo[8] « Eye Of The Tagger » restera présent tout au long de l'utilisation.
3. L'utilisateur rédige un texte manuellement.	4. Le système affiche le texte dans la **zone de saisie**[1].
5. L'utilisateur **clique** sur le **bouton radio**[2] correspondant à l'outil de REN de son choix.	6. Le système affiche l'outil choisi.

Post-conditions : L'analyse du texte sélectionné est prête à être lancée.

Enchaînements alternatifs

A1 : L'utilisateur importe un texte depuis une URL

L'enchaînement A1 démarre au point 3 du scénario nominal.

15. L'utilisateur **clique** sur l'**onglet**[4] « Saisir URL ».

Voir scénarios réels du CU "Fournir une URL"

Le scénario nominal reprend au point 4.

A2 : L'utilisateur importe un texte depuis sa machine

L'enchaînement A2 démarre au point 3 du scénario nominal.

3. L'utilisateur **clique** sur l'**onglet**[5] « Importer texte ».

   Voir scénarios réels du CU "Fournir un chemin local"

Le scénario nominal reprend au point 4.

A3 : L'utilisateur importe un texte depuis une URL après en avoir saisi un manuellement

L'enchaînement A3 démarre au point 5 du scénario nominal.

5. L'utilisateur **clique** sur l'**onglet**[4] « Saisir URL ».
6. Le système ne conserve pas la saisie en cours.

   Voir scénarios réels du CU "Fournir une URL"

Le scénario nominal reprend au point 4.

A4 : L'utilisateur importe un texte depuis sa machine après en avoir saisi un manuellement

L'enchaînement A4 démarre au point 5 du scénario nominal.

5. L'utilisateur **clique** sur l'**onglet**[5] « Importer texte ».
6. Le système ne conserve pas la saisie en cours.

Voir scénarios réels du CU "Fournir un chemin local"

Le scénario nominal reprend au point 4.

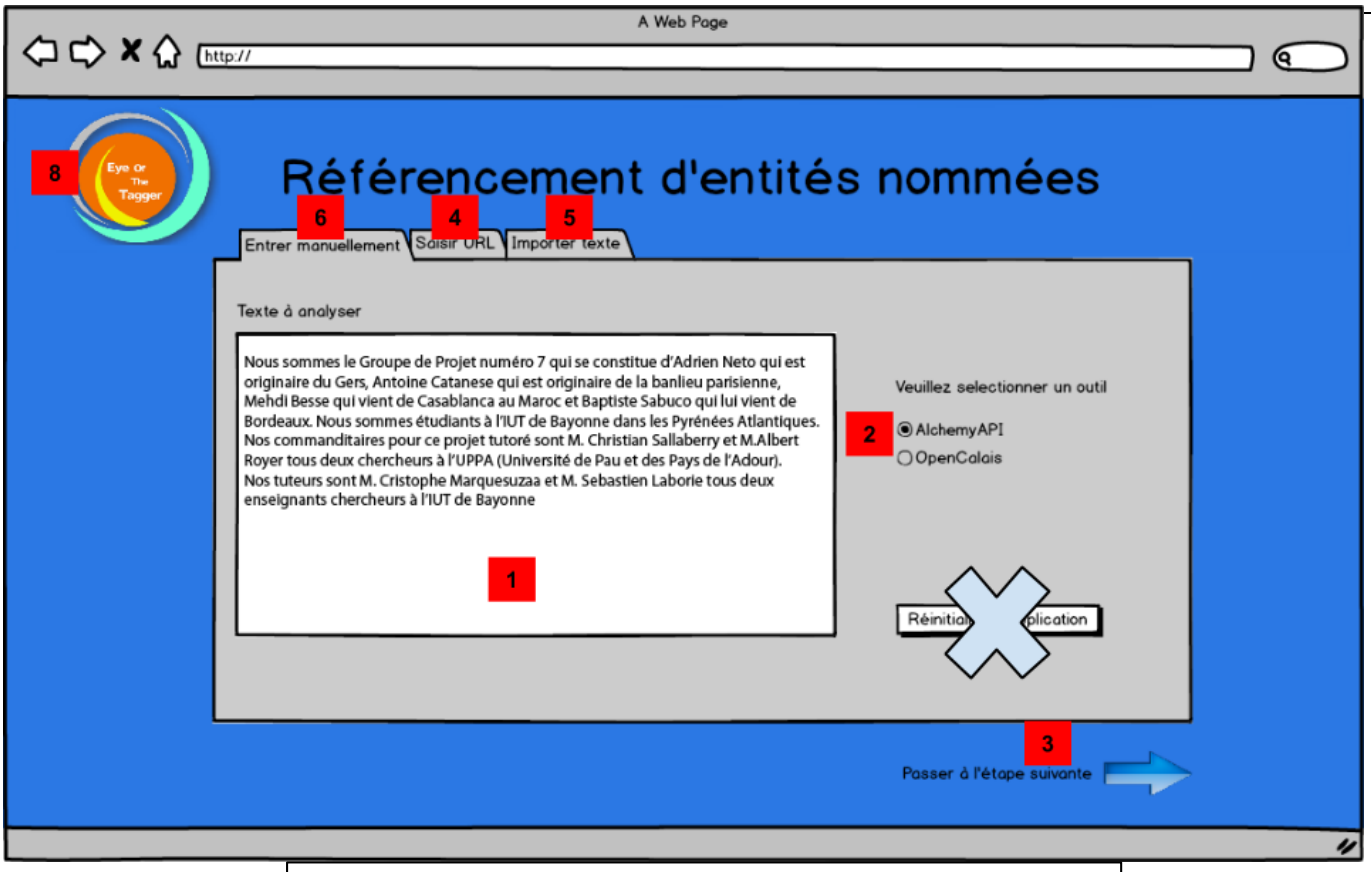

Figure 17: Maquette VUE 1 - Scénarios réels - CU « Fournir un texte a analyser »

2.4. DEROULEMENT DE L'APPLICATION

Après avoir produit deux versions simplifiées du schéma de déroulement général de l'application, nous avons produit la dernière version incluant tous les traitements que voici :

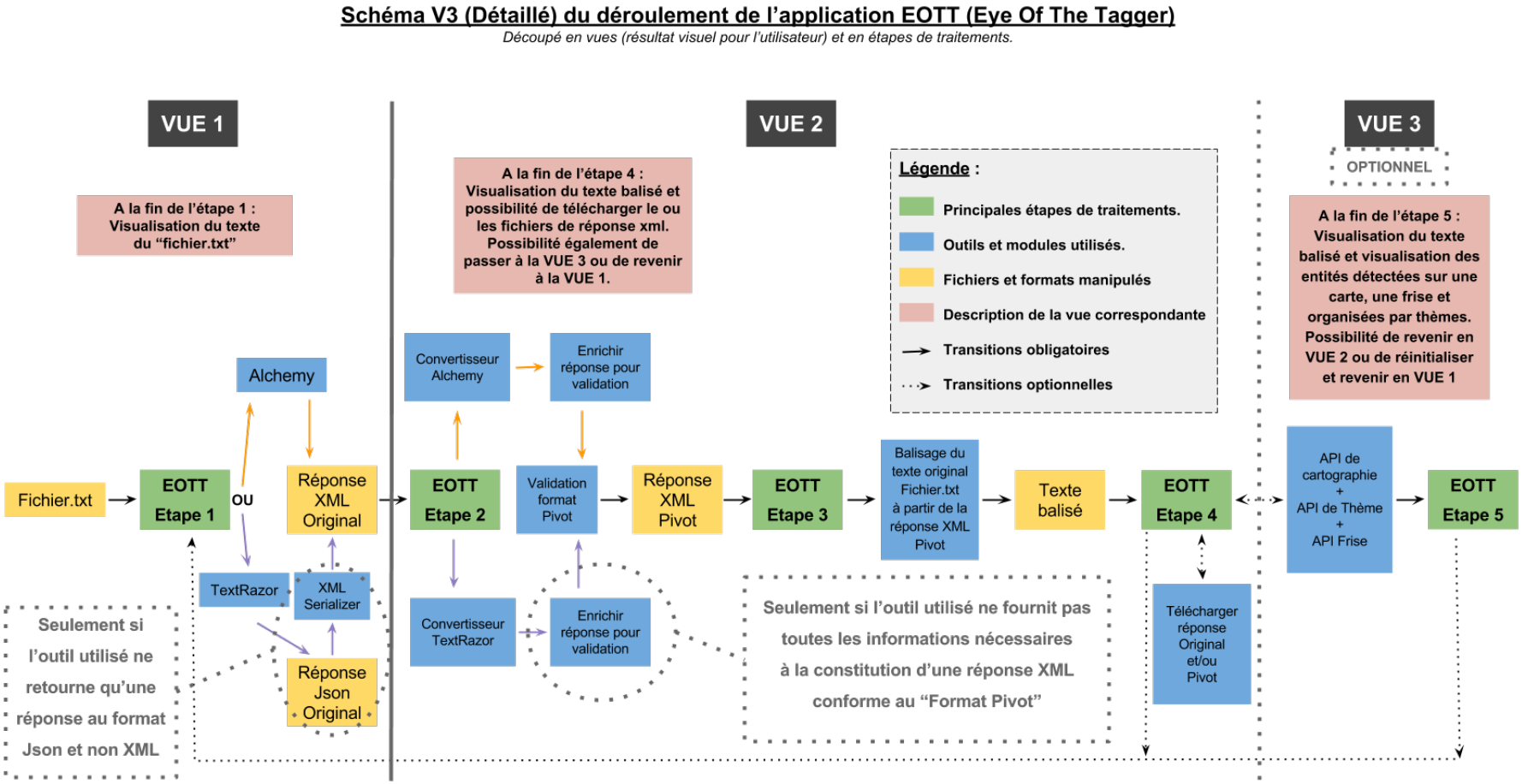

Figure 18 : Schéma détaillé du déroulement général de l'application V3

3. Gestion de projet

3.1. DEMARCHE DE DEVELOPPEMENT

La première équipe mise en place pour réaliser ce projet présentait des caractéristiques intéressantes puisque qu'elle rassemblait des profils très différents. En effet, certains membres moins performants en analyse et programmation pure, compensaient par de réelles qualités de designers et ergonomes.

Nous avons donc décidé de tirer profit de cette situation en scindant l'équipe.

La première sous-équipe ayant plus de capacités techniques, a été affectée aux tâches de recherche et développement, la deuxième, possédant davantage une fibre artistique, s'est chargée de la partie graphique, c'est-à-dire du design et de l'IHM (Interface Homme Machine).

Nous avons dû prendre très tôt en compte une contrainte forte, à savoir l'utilisation des outils existants de REN (Reconnaissance d'Entités Nommées). Cette contrainte nous obligeant à nous adapter aux outils disponibles, nous avons dû mettre tout de suite en place des tests de faisabilité afin de déterminer si l'outil présélectionné répondait bien à nos besoins et correspondait à nos attentes.

Dans cette optique, nous avons décidé d'utiliser une méthode de développement appelée RAD2 (*Rapid Application Developpement*) et appartenant à la famille des méthodes de Génie Logiciel (GL) dites méthodes AGILE.

Cette méthode particulièrement bien adaptée au développement de petits projets s'appuyant sur des technologies nouvelles ou non maitrisées, repose essentiellement sur la communication rapide et régulière entre les sous-équipes, de toutes les évolutions du projet. En effet, au moment de la répartition des tâches, chaque équipe se spécialise sur une problématique précise et réduite du projet. Puis, suite à une première analyse concise du problème, elle développe directement un premier prototype. Une fois que la faisabilité du module est prouvée par la présentation au reste de l'équipe de ce prototype fonctionnel, le module est intégré à la structure de l'application, et l'équipe s'attaque alors au problème suivant.

En résumé, cette méthode de développement au cycle de vie itératif et incrémental, repose sur une spécialisation des rôles, une formalisation de processus légers, une architecture de réalisation (imbrication des itérations), une forte utilisation des réunions et moyens de communications, par une organisation régulière de différents types de réunions (technique, organisationnel…) afin de maintenir une cohésion entre les membres du projet et faire en sorte que chacun soit, en quasi temps réel, mis au courant de l'avancée du projet.

3.2. RECHERCHE ET DEVELOPPEMENT

3.2.1. Étude de faisabilité ; les outils de R.E.N.

Comme nous l'avons vu dans les enjeux (Partie 1.2 - *Figure 6*), les commanditaires nous ont fourni, en début de projet, une liste d'outils de REN, présélectionnés par leurs soins en fonction de leurs performances. Dans un premier temps, nous avons donc dû les étudier afin d'en sélectionner deux.

Nos critères de sélection ont d'abord été la possible utilisation de l'outil depuis une application tiers ou en d'autres termes, savoir s'il s'agit d'un service web ou d'une simple application. En effet, CiceroLite ou encore Broccoli ne peuvent être utilisés que depuis les sites originaux et ne sont donc pas exploitables depuis une autre application.

Le deuxième critère, et non des moindres, est de pouvoir utiliser ces services web depuis une application web développée en Php, un langage de programmation spécifique au web (voir partie 4.3 « Langages et technologies »). Nous avons donc dû retirer de notre liste le service ANNIE de Gate, ainsi que le service Stenford Ner qui ne proposent qu'un environnement de développement Java et sont donc inexploitables aux vues de nos objectifs.

Un autre outil, HeidelTime, ne répond pas complètement à nos besoins puisqu'il ne détecte que les entités temporelles. Etant donné que notre application est un outil de détection d'entités spatio-temporelles et thématiques, cet outil n'est pas assez complet et a donc été lui aussi supprimé de la liste.

Enfin, nous avons retiré l'outil N.E.R.D. de notre présélection dans la mesure où cet outil n'en est pas un à proprement parler, puisque qu'il intègre en réalité de nombreux outils existants, notamment des outils présents dans notre sélection. Cet outil s'apparente donc plus au type d'application que nous devons mettre en place, qu'à un outil de R.E.N. pure. Cependant, nous avons pu nous en inspirer, en ajoutant notamment à notre présélection, certains des outils qu'il utilise et que nous ne connaissions pas comme par exemple TextRazor (*https://www.textrazor.com/*).

Une fois cette deuxième présélection effectuée, nous avons étudié chacun des outils restants afin de connaître la richesse de leurs résultats, de leurs documentations et de leurs communautés (activité sur les forums de développeurs utilisant ces outils). En raison de ces trois critères, nous avons dans un premier temps, sélectionné Alchemy API et Open Calais.

Après avoir étudié plus en détails ces deux outils, nous avons effectué nos premiers appels à l'aide de formulaires HTML, à partir desquels nous avons effectué de simples requêtes HTTP POST et ainsi récupéré nos premières réponses originales.

Ces deux outils étant paramétrables, ce qui nous permettait de choisir le format de réponse souhaité (XML, Json, RDF…), nous demandions des réponses au format XML. En effet, il avait été décidé avec les commanditaires que ce format serait préféré, étant donné qu'il est pour le moment le plus répandu sur la toile.

Alchemy API retournant une réponse au format XML, nous avons pu très vite commencer à travailler sur son convertisseur au format pivot.

En revanche, Open Calais retourne une réponse au format XML/RDF, c'est-à-dire un document RDF, rangé au format XML. Il faut savoir que le RDF (voir « Langage et Technologie » Partie 4.3) bien que pouvant être rangé au format

XML, peut également être rangé au format Json ou textuel. Ce format représente en réalité une organisation complètement différente de l'organisation d'un fichier XML « simple » puisqu'elle représente un graphe, et non une arborescence. Cela en fait donc un format bien plus riche qu'il est quasiment impossible de convertir en XML, puisqu'il y a nécessairement une perte d'informations. Automatiser le traitement de sélection de ce que l'on garde ou non, est compliqué et génère bien souvent des erreurs ainsi que des pertes d'informations importantes et non souhaitées.

Après avoir essayé de passer par la réponse Json proposée par OpenCalais afin de solutionner notre problème, nous avons remarqué que la réponse Json était également une réponse RDF et qu'à défaut de trouver une solution, l'on tournait en rond. Il a donc été décidé après deux mois de recherches d'abandonner cet outil, ce à regret car il comptait parmi les plus complets et les plus puissants des différents outils de R.E.N. évalués.

Nous nous sommes donc concentrés exclusivement sur Alchemy API afin de mettre en place un premier prototype de l'application qui déroulerait toutes les étapes décrites précédemment (voir « Fonctionnalité » Partie 1.3) ; ce, en vue de prouver la faisabilité de l'application en n'y intégrant qu'un seul outil. Par suite, il s'agissait de montrer que l'intégration d'un outil supplémentaire resterait simple et rapide.

Pour ce faire, nous avons utilisé un module fourni par l'équipe de développement d'Alchemy API (Alchemy Team), contenant une classe nommée AlchemyAPI. Celle-ci permet d'invoquer le service en créant un nouvel objet AlchemyAPI, puis en utilisant sa méthode prévue à cet effet. Le problème que nous avons rencontré est que cette méthode retournait un fichier Json converti en array (tableau en php). Il nous a fallu modifier les paramètres d'appel à l'intérieur-même de la classe puisqu'aucune méthode ne permettait de modifier ce paramètre.

A ce stade, le convertisseur était prêt et nous avions donc pu produire des réponses converties au format pivot à partir desquelles nous avions développé le module d'appel de la carte. Ce module, ainsi que le convertisseur ont donc été intégrés à l'application. Cependant, AlchemyAPI ne fournissant pas des réponses suffisamment complètes pour répondre aux exigences du format pivot, nous avons dû développer différents modules « d'enrichissement » de ces réponses ; notamment, en faisant appel à de nouveaux services comme geonames, afin de récupérer les coordonnées GPS des entités spatiales ou des fonctions php pour récupérer les positions des EN dans le texte.

Une fois AlchemyAPI totalement intégré (appel + conversion), nous nous sommes concentrés sur la constitution du module d'affichage de la frise chronologique afin de finaliser complètement la structure de l'application avant d'envisager l'intégration d'un nouvel outil. Malheureusement, AlchemyAPI ne détectant pas les entités temporelles, nous avons été obligés de revoir notre organisation et de reprendre notre recherche d'outil en ayant cette fois, comme contrainte forte, l'obligation de trouver un outil qui, en plus de remplir les critères de sélection exposés plus haut, soit capable de détecter des entités temporelles. Nous avons donc repris notre présélection d'outils et nous sommes cette fois orientés vers le choix de TextRazor, qui se trouve être un des outils de REN utilisé par N.E.R.D. En effet, cet outil nous est apparu comme l'un des plus complets en matière de détection de dates et périodes temporelles et a donc fait l'objet d'une étude approfondie en vue de son intégration à notre application.

Suite à cette étude une nouvelle difficulté est apparue puisque cet outil n'a qu'un seul format de retour, le Json (voir « Langage et Technologie » Partie 4.3). Il a donc fallu que nous trouvions un moyen de convertir cette réponse en XML

afin de pouvoir l'intégrer à notre application et ainsi tester l'intégration de la frise chronologique qui avait été développée entre-temps. Pour ce faire, nous avons cherché de nombreuses méthodes et solutions à notre problème mais la plupart des solutions trouvées sur Internet (forum comme « Developpez » ou « commentcamarche ») s'apparentaient plus à du bricolage et ne nous fournissaient pas le rendu espéré. Après plusieurs essais de différentes méthodes, nous avons été contraints de choisir un module d'extension Php appelé PEAR (voir « Langage et Technologie » Partie 4.3) qui contient notamment le package « XML_Serializer », lui-même constitué de différentes classes et permettant relativement simplement, de convertir une structure de données Php de type Array (tableau de données) en un objet XML de type « SimpleXML ». Il faut savoir qu'il existe également une fonction Php (json_decode) permettant très simplement de convertir du Json en un tableau de données.

Cependant, l'utilisation de ce package a soulevé de nouvelles difficultés puisque ce dernier n'étant disponible qu'en version beta (encore en phase de test), son utilisation entrainait l'apparition d'une multitude de messages d'avertissement dus notamment à l'utilisation de méthodes de classes qui n'était pas déclarées comme telles (absence de l'indicateur « *static* » devant les méthodes concernées). Après avoir bien étudié la manière dont les trois ou quatre classes que nous utilisions était implémentées, nous décidâmes de les récupérer indépendamment de leur package et de se les approprier en les corrigeant de manière à ce qu'elles répondent le plus possible à nos besoins. En outre, cette solution était nécessaire afin de pouvoir déployer notre application sur « Erozate », le serveur de recherche de nos commanditaires sans avoir à lui installer de modules supplémentaires ou à modifier sa configuration.

Nous nous trouvions donc prêts à développer un convertisseur pour ce nouvel outil ainsi qu'à faire évoluer notre format pivot afin qu'il intègre les nouvelles informations récupérées par TextRazor, qu'AlchemyAPI ne traitait pas comme les entités temporelles ou les espèces animales (saumon).

Enfin nous avons pu finaliser notre application en démontrant par l'utilisation de TextRazor, que notre module de frise chronologique était fonctionnel. En outre, l'intégration d'un nouvel outil par le simple ajout à l'application d'un module d'appel et d'un module de conversion, démontre la modularité de notre application. Ce critère était d'ailleurs primordial comme nous l'avons vu dans les contraintes fonctionnelles (voir « Contraintes fonctionnelles » 1.4).

3.2.2. Le format pivot

Afin de constituer un format pivot (format standard) répondant aux exigences des commanditaires, tout en intégrant un maximum d'informations venant de sources différentes, nous nous sommes tout d'abord appuyés sur un premier schéma, appelé « modèle pivot », fourni par les commanditaires et élaboré à partir de nos premiers appels d'AlchemyAPI. Nous l'avons enrichi, au fur et à mesure de l'avancée du projet, en fonction de nos besoins ou des nouvelles exigences de nos commanditaires, puis traduit en anglais afin d'internationaliser le code.

Le modèle pivot fourni par nos commanditaires n'était qu'une ébauche suffisamment étoffée pour nous indiquer la voie à suivre. Celui-ci permettait la prise en compte de deux types d'entités (spatiale et autres) pour les visualiser sur une carte ou sous forme d'un arbre.

Ce premier modèle nous a permis de prendre en main le processus de conversion et ainsi de produire un premier prototype de convertisseur conforme aux restrictions imposées par le modèle pivot.

Cependant, nous nous sommes heurtés à quelques difficultés quant à la mise en œuvre de ce premier modèle notamment lors de la formation de la balise texte qui contenait trois attributs : la position du mot dans le texte, sa taille et ses coordonnées géographiques. De plus, un attribut étant unique, la formation de cette balise ne nous permettait pas de récupérer toutes les positions d'une entité apparaissant plusieurs fois dans un texte. Pour y remédier nous avons choisi de changer ces attributs en balises distinctes.

Au fur et à mesure de l'avancée du projet, nous en sommes venus à étoffer ce schéma en y ajoutant la prise en compte d'entités temporelles permettant la possibilité de collecter des informations temporelles en vue de les afficher sur une frise chronologique.

Ce schéma se compose donc de trois types d'entités distinctes disposant chacune d'un contenu spécifique dont l'arborescence complète est disponible en annexe.

Afin d'illustrer ce schéma, nous allons examiner l'arborescence d'une entité spatiale.

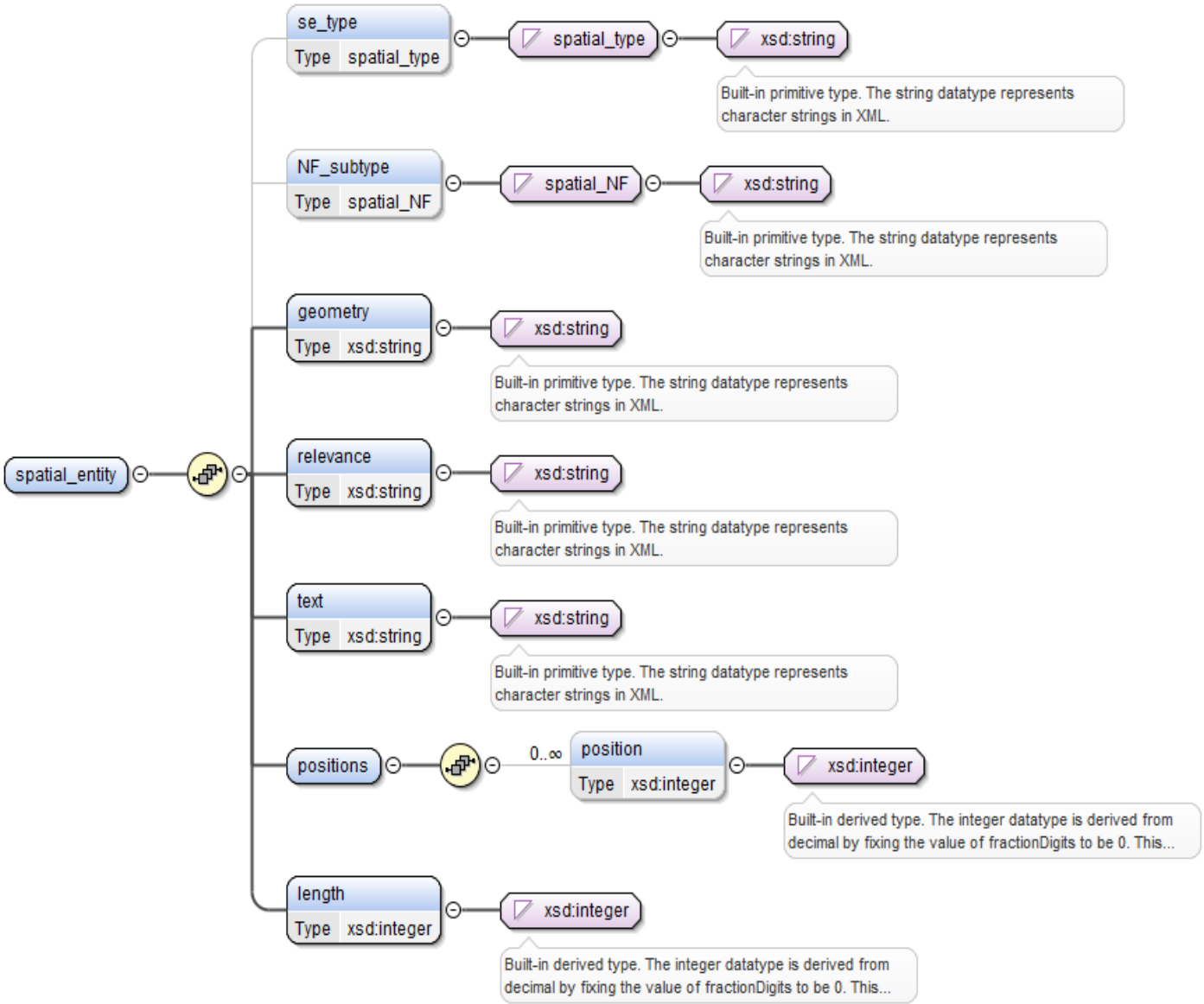

Figure 19 : Schéma produit à parir d'une entité spatial dans le format pivot

On observe ainsi qu'une entité spatiale est caractérisée par :

- Un type. Ex: City, Country, Natural_Feature.
- Un sous-type lorsque le type est Natural_Feature.
- Des coordonnées géographiques (GPS).
- Un indicateur de pertinence.
- Un texte qui lui est associé.
- Sa ou ses positions dans le texte.
- Sa taille en nombre de caractères.

L'intérêt de ce format pivot est d'unifier les réponses fournies par les différentes Api utilisées, afin de retravailler ces informations à l'aide d'outils de visualisation que nous verrons par la suite.

3.2.3. Les outils de visualisation ; la carte, la frise chronologique et l'affichage des thèmes

3.2.3.1. La carte :

Pour permettre d'organiser les éléments obtenus par l'analyse d'AlchemyAPI, il a été convenu que les entités spatiales seraient disposées dans la mesure du possible sur une carte.

Pour se faire, nous avons décidé d'utiliser l'API de Google Maps qui possédait une documentation complète et suffisante pour effectuer les géolocalisations.

Dans premier temps, lors de l'analyse d'un texte, AlchemyAPI retournait seulement le nom d'une entité spatiale après l'avoir repérée. Ainsi, le module Google récupérait le nom de ce lieu, en déduisait les coordonnées par rapport à sa base de données grâce à une procédure déjà existante, et plaçait des marqueurs correspondant à ces coordonnées grâce à une seconde procédure.

Toutes ces informations s'affichaient de cette manière :

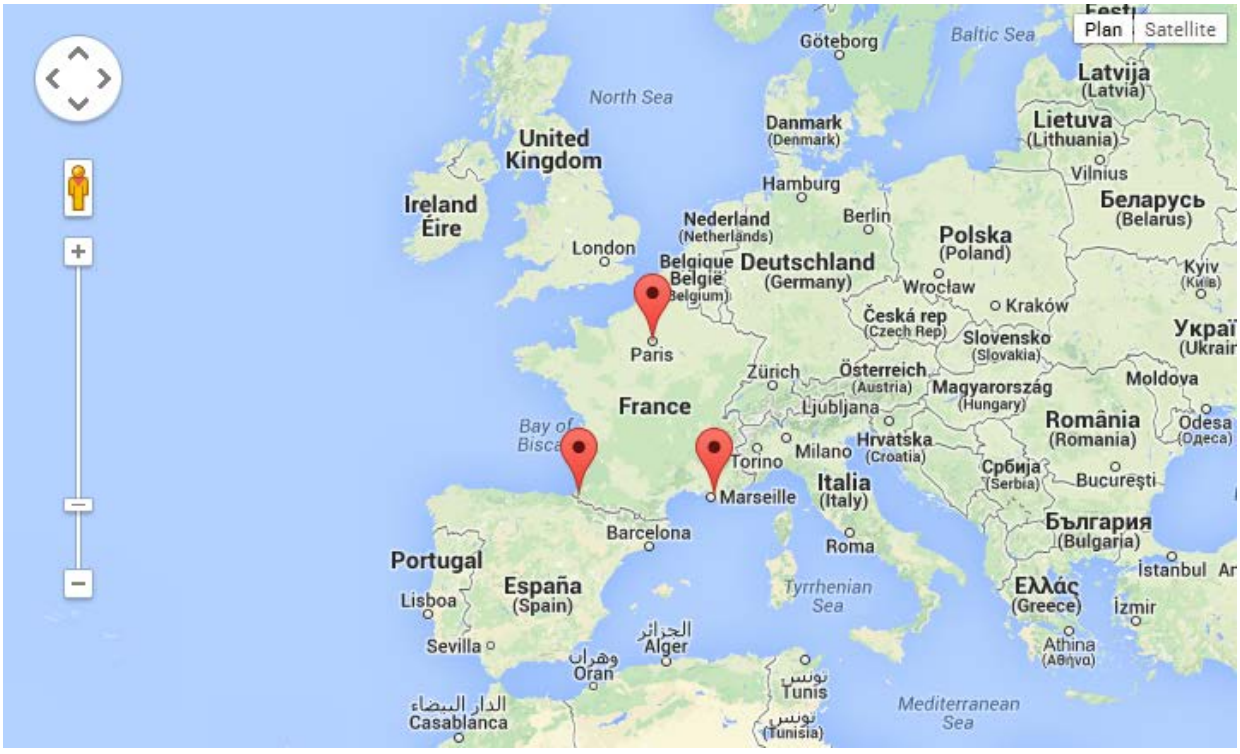

Figure 20: Exemple de fonctionnement de la carte Google Maps

Dans un second temps, nous avons réussi à récupérer les coordonnées d'une entité spatiale à partir d'AlchemyAPI, et ces données se trouvaient directement présentes dans le fichier pivot XML sous cette forme :

```
<entity>
    <spatial_entity>
        <se_type>Country</se_type>
        <relevance>0.100375</
        <text>France</text>
        <geometry>53 28</ge
        <positions>
            <pos>54</pos>
            <pos>134</pos>
            <pos>346</pos>
        </positions>
        <length>6</length>
    </spatial_entity>
</entity>
```

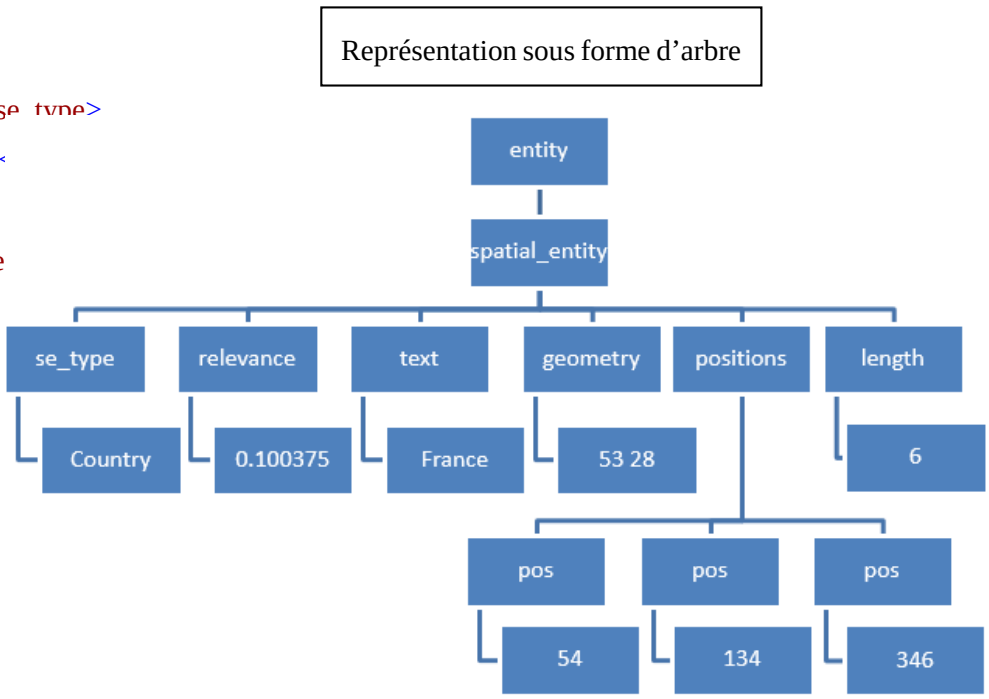

Figure 21 : Arborescence d'une entité spatiale dans le format pivot.

Grâce à ces informations, nous avons pu afficher les éléments sur la carte directement grâce aux coordonnées présentes dans la base de données d'AlchemyAPI.

Après traitement par AlchemyAPI, le texte analysé était retourné avec certaines couleurs, permettant ainsi d'identifier et de différencier les entités que l'outil avait reconnues. Cette identification était déclinée pour les différentes entités spatiales tels que les pays (balisés « Country ») ou les villes (balisées « City »). Ainsi, pour permettre un affichage plus clair sur la carte, il a été convenu que les couleurs des marqueurs de la carte correspondraient avec le type d'entité spatiale qu'AlchemyAPI avait reconnu.

Par la suite, l'intégration de TextRazor n'a pas modifié l'intégration de ce module.

3.2.3.2. La frise :

Pour permettre la visualisation des entités temporelles, nous avons décidé d'utiliser un module de *Timeline*.

Un de nos tuteurs nous a conseillé d'utiliser la librairie Chap Links qui possédait une *Timeline* et qui paraissait assez simple à utiliser puisqu'elle avait déjà servi lors du projet précédent et que le résultat était dans l'idée que nous nous faisions d'une frise chronologique.

Durant notre phase de recherche sur cet outil, nous avons également pensé à utiliser Simile Timeline car Jérémy, qui était arrivé dans notre projet en milieu d'année, l'avait utilisé dans son projet précédent.

Après réflexion, il nous a semblé plus facile d'utiliser le widget de Simile puisque l'un des participants avait déjà utilisé cet outil. Mais il est toujours possible de remplacer cette Timeline par celle de Chap Links.

Dans le fichier pivot que nous utilisons, les entités temporelles se présentent ainsi :

```
<entity>
    <temporal_entity>
        <te_type>Date</se_type>
        <relevance>0.100375</relevance>
        <text>25/03/2014</text>
        <positions>
            <pos>5</pos>
            <pos>59</pos>
        </positions>
        <length>10</length>
    </temporal_entity>
</entity>
```

Grâce à l'outil Simile, nous pouvons afficher des événements ponctuels (représentés par les points) ou bien des événements qui s'effectuent sur la durée (représentés par des rectangles), bien que TextRazor ne reconnaisse pas les événements comme «*La Deuxième Guerre Mondiale*» en tant qu'un événement. Pourtant, si un outil de reconnaissance d'entités nommées est susceptible de retourner, il sera possible d'afficher ce type d'entité.

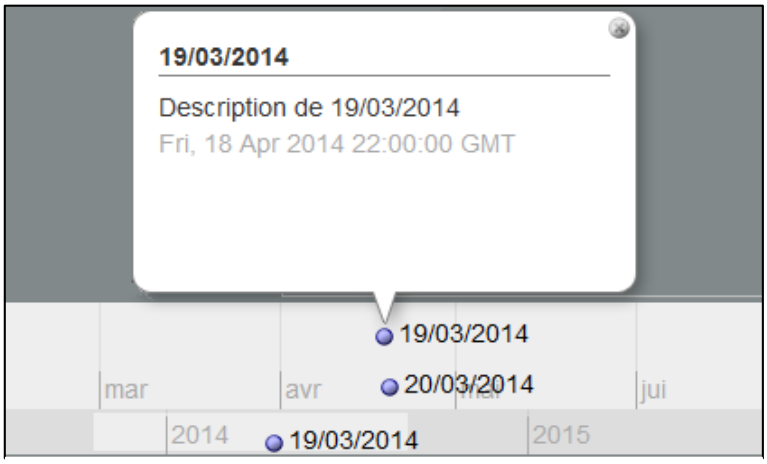

Figure 22 : Exemple de fonctionnement de SIMILE Timeline

3.2.4. La mise en page ; Bootstrap

Afin de réaliser la mise en page de cette application web, nous avons d'abord commencé par faire des maquettes afin de comprendre ce que les commanditaires voulaient en termes d'interface et de charte graphique. Une fois ces maquettes validées, nous avons commencé à mettre en place cette interface classiquement grâce aux feuilles de style CSS. Seulement au cours du développement de l'application, nous avons rencontré des difficultés de mise en place car les modules, tels que le module d'affichage de la carte ou encore le module d'affichage des thèmes avec ChapLinks, demandaient des paramétrages CSS complexes. Nous avons donc décidé d'utiliser Bootstrap qui est un framework CSS créé par la société Twitter. Ce framework intègre déjà plusieurs outils basiques qui permettent la mise en forme de composants simples comme par exemple des formulaires ou encore des boutons. De plus, Bootstrap permet de mettre en place le design d'un site web plus simplement car ce framework fonctionne avec un système de rangs et de colonnes, et permet de redimensionner les colonnes selon la taille de la fenêtre afin de l'adapter à la résolution de l'utilisateur. L'autre avantage d'utiliser ce framework, est qu'il est possible de le coupler avec jQuery, un framework JavaScript, afin de mettre en place facilement de nombreux plugins, comme par exemple des listes déroulantes, des onglets ou même un effet accordéon comme le permet le module de ChapLinks que nous avons utilisé pour l'affichage des thèmes.

3.3. PLANNINGS

Afin de s'adapter à notre organisation en équipes tout en gardant une cohésion globale, nous avons, dans un premier temps, organisé la répartition des tâches selon le planning suivant (V1 - le 30 Octobre 2013) :

Semaines civile	40	41	42	43	44	45	46	47	48	49	50
Semaines IUT	5	6	7	8	9	10	11	12	13	14	15
Tâches	Étude préliminaire des outils proposés		Prise en main des deux outils sélectionnés (OpenCalais, AlchemyAPI)		Prise en main d'OpenCalais et d'AlchemyAPI Mise au point d'une première maquette de l'application finale		Présentations des choix des différents formats et protocoles choisis Présentations de la charte graphique finale (générale)	Rédaction de la charte du projet	Validation des différents formats et protocoles choisis Validation de la charte graphique finale (générale)	Étude sur la standardisation des formats de retours et analyse des différents modules de l'application	

Semaines civile	51	52	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Semaines IUT	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Tâches	Conception des différents modules de l'application		Développement des différents modules de l'application avec intégration systématique au site déjà construit.						Finalisation de l'intégration de l'application ET rédaction du manuel d'utilisation ET finalisation du manuel technique				Tests et validation de l'application avant livraison ET rédaction du mémoire			Livraison du produit fini
	Construction de la structure du site qui accueillera l'application				Rédaction de la doc technique		Finalisation de la structure du site sur le plan graphique									

LEGENDE	Tout le groupe
	Antoine et Baptiste
	Adrien et Mehdi

Tout en respectant la répartition des tâches par spécialités, nous avons tenu à conserver des tâches en commun. Néanmoins, nos tuteurs nous ont fait remarquer que cette représentation n'était pas assez détaillée dans la mesure où elle ne faisait pas mention de tâches précises et ciblées dans le temps mais plutôt d'activités générales à mener par l'équipe pendant une période donnée. En outre, les jalons n'apparaissaient pas précisément avec des dates butoirs.

Nous avons donc détaillé davantage notre planification en prenant en considération les défauts soulevés et ainsi produit le planning suivant (V2 - 15 décembre 2013) :

Semaine civile N°	40	41	42	43	44	45	46	47	48	49	50	51	2	3	4	5	6	7	8	9	10	11	12	13	14
Semaine IUT N°	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
JALONS ET DATES IMPORTANTES =>										**1**			**2**	**3**	**4**		**5**		**6**		**7**		**8**	**9**	**10**
Dates des débuts de semaines	30/09	07/10	14/10	21/10	28/10	04/11	11/11	18/11	25/11	02/12	09/12	16/12	06/01	13/01	20/01	27/01	03/02	10/02	17/02	24/02	03/03	10/03	17/03	24/03	31- mars
↓Principales tâches↓	PHASE 1 - Analyse					PHASE 2 - Analyse					PHASE 3 - Conception			PHASE 4 - Développement				PHASE 5 - Intégration				PHASE 6 - Recettes et livraison			
Veille technologique	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X								
Étude préliminaire des outils proposés	X	X																							
Étude des différents langages à utiliser	X	X	X	X	X	X																			
Prise en main des outils sélectionnés			X	X	X	X																			
Mise au point de maquettes et scénarios					X	X	X	X	X	X	X	X													
Développement d'un prototype et de codes de test pour évaluer la faisabilité du projet (Antoine)							X	X	X	X	X														
Élaboration d'un format standard de sortie (Baptiste)									X	X	X	X													
Rédaction du CDCF (Charte)									X	X	X														
Développement du squelette du site (HTML/CSS)												X	X	X	X	X	X								
Rédaction du CDCT										X	X	X													
Développement du module de balisage du texte à partir du fichier XML retourné												X	X												
Intégration du module et tests unitaires												X	X	X											
Correction des bugs relevés pendant les tests														X											
Développement du module de représentation sur une carte des lieux détectés														X	X										
Intégration du module et tests unitaires														X	X	X									
Correction des bugs relevés pendant les tests																X									
Développement du module de représentation sur une frise des dates et périodes détectées																X	X								
Intégration du module et tests unitaires																X	X	X							
Correction des bugs relevés pendant les tests																		X							
Rédaction de la doc technique														X	X	X									
Rédaction du manuel d'utilisation																	X	X	X						
Finalisation de la charte graphique																		X	X	X	X				
Finalisation de l'intégration des différents modules																		X	X	X	X				
Recette finale																						X	X		
Correction des bugs relevés pendant les tests																							X	X	X
Rédaction du mémoire									X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		
Préparation de la soutenance																						X	X	X	X
Réunion de suivi d'avancée du projet	X	X		X		X		X		X		X													
Dates des débuts de semaines	30/09	07/10	14/10	21/10	28/10	04/11	11/11	18/11	25/11	02/12	09/12	16/12	06/01	13/01	20/01	27/01	03/02	10/02	17/02	24/02	03/03	10/03	17/03	24/03	31- mars

Ci-dessous la légende avec le détail des jalons :

Équipe responsable de l'aspect technique du projet : **Antoine Catanese et Baptiste Sabuco** Équipe responsable de l'aspect graphique et ergonomique : **Adrien Neto et Mehdi Besse** **L'équipe au complet**	**1)** Oral de présentation du projet – **2)** Remise de la charte (CDCF) – **3)** Validation du format de sortie et du CDCT – **4)5)6)** Remise et validation des tests unitaires des 1ers, 2èmes et 3èmes modules – **7)** Validation de la charte graphique – **8)** Remise du produit fini – **9)** Remise du mémoire accompagné des manuels d'utilisation et de maintenance – **10)** Présentation de la soutenance de fin de projet

Bien que cette planification soit plus détaillée que la précédente, elle ne s'est pas avérée très pratique car elle ne permet pas de différencier les tâches accomplies des tâches en cours ou non commencées. En outre, elle fait apparaître chaque semaine comme étant équivalente en temps, alors qu'elles ne se valent pas toutes. En effet, certaines semaines comme les semaines de vacances ou de programmation, nous laissent plus de temps pour avancer dans le projet qu'une semaine durant laquelle nous avons des partiels ou d'autres projets à rendre. L'une de nos difficultés, également, réside dans le fait que nous avons tenu à mettre en place une planification claire, précise et détaillée, en la faisant tenir sur une seule page afin de conserver en permanence une vision globale de l'avancement du projet. Enfin, nous avons décidé de ne pas utiliser d'outils comme Gantt que nous considérons comme trop contraignant ; or force est de constater que nous aurions dû. Nous l'utiliserons certainement à l'avenir lors de futurs projets car la planification est un secteur dans lequel nous avons vraiment péché lors de ce projet.

Il a été malaisé pour nous de rendre par écrit chacune des tâches à accomplir et à se fixer des dates butoirs. La planification n'a d'ailleurs que peu été mise à jour, cependant nous avons mis en place des petites planifications parallèles dont voici un exemple (Planification rédaction Mémoire – 11 mars 2014):

TRAVAIL A FAIRE	Responsable	Priorité	Avancement	Précisions
1 - Charte	TOUS	HAUTE	Terminée	
2 - Analyse Fonctionnelle				
_2.1 - DCU	ANTOINE		Terminée	
_2.2 - Maquettes	ANTOINE		Terminée	
_2.4 - Enchainement vues	ANTOINE		En cours	Intégré aux scénarios
_2.3 - Scénarios essentiels	ANTOINE		En cours	
3 - Gestion de projet				
_3.1 - Démarche de dvpt	ANTOINE	HAUTE	Terminée	
_3.2 - Reflexions de recherche	ANTOINE	HAUTE	Terminée	
_3.3 - Planning 1ere Equipe	ANTOINE	HAUTE	Terminée	
_3.4 - Planning 2nde Equipe	ANTOINE	HAUTE	A faire	
_3.5 - Organisation du projet	JEREMY	HAUTE	Terminée	
4 - Conception et programmation				
_4.1 - Diag Seq et/ou Diag Activ	BAPTISTE		A faire	
_4.2 - DET ou équivalent	BAPTISTE		A faire	
_4.3 - Langages et Technologies	ANTOINE		A faire	
_4.4 - APIs	JEREMY		Terminée	
5 - Perspectives et Bilan				
_5.1 - Perspectives	JEREMY			
_5.2 - Bilan	MARC	HAUTE	Terminée	
6 - Abstract	JEREMY		Terminée	
Manuel d'installation	MARC			
Manuel d'utilisation	MARC	HAUTE	A faire	
Manuel de maintenance	JEREMY ANTOINE			
Comptes-rendus réunions	MARC		A faire	Besoin des notes de MARC

DOCUMENTATION				
Enrichir tableau Doc	JEREMY		A faire	
Doc AlchemyAPI	JEREMY		A faire	Traduire + Revoir
Doc OpenCalais	JEREMY		A faire	Traduire + Revoir
Doc TextRazor*	JEREMY		A faire	

Cette planification a été établie pour tenter de palier le retard pris sur la rédaction du Mémoire, néanmoins son manque de précision nous a été à nouveau préjudiciable. En effet, l'absence de dates butoirs de réalisation pour chaque tâche, a eu pour conséquence le fait que nous nous sommes dispersés sur plusieurs tâches à la fois, alors que nous aurions dû nous concentrer sur chaque point en se répartissant mieux les rôles.

Nous avons porté ces problèmes d'organisation et de dispersion tout au long du projet, sans que nous parvenions réellement à les résoudre.

3.4. ORGANISATION DU PROJET

Comme nous l'avons vu dans la planification, au départ du projet, nous nous sommes répartis les tâches en fonction des compétences de chacun. Antoine et Baptiste ont pris en charge la veille technologique et le développement des différents modules, Adrien et Mehdi se sont occupés de l'aspect visualisation et de la structure de l'application (corps codé en HTML/CSS). Antoine a été désigné chef de projet en raison de son âge plus élevé que le reste de l'équipe et de ses compétences. Nous savions à cet instant que nous prenions un risque étant donné qu'Antoine ne possédait pas selon lui, les qualités d'organisation requises pour un chef de projet. Malgré sa volonté forte de mener à bien ce travail, force a été de constater que notre organisation ne nous a pas permis de progresser uniformément. Nous avons avancé rapidement sur l'aspect recherche et développement, et pris du retard sur la production de documentation technique et organisationnelle (planification, compte rendu de réunions…).

Nos commanditaires étant sur Pau, la quasi-totalité de nos réunions se sont déroulées par visioconférence, ce qui a constitué une difficulté supplémentaire en matière d'organisation et de préparation des réunions. Nous avons péché par défaut de préparation systématique d'ordre du jour détaillé pour nos réunions qui, à ce titre, n'ont parfois pas été efficacement menées.

En janvier, à la fin du troisième semestre, deux des membres de l'équipe (Adrien et Mehdi), qui avait à compenser la non-obtention de leur deuxième semestre, n'ont pas réussi à atteindre leurs objectifs, ont été contraints de redoubler et ainsi de quitter l'équipe du projet. A ce stade, nous ne pouvions plus continuer le projet puisqu'une équipe doit être composée de trois étudiants au minimum. Par chance, deux étudiants également « abandonnés » par leurs coéquipiers respectifs, ont pu se joindre à notre projet et ainsi nous permettre de continuer.

Ces deux nouveaux membres sont Jérémy Dulon et Marc Lagreou. Ils se sont très bien intégrés au projet pourtant pas évident à maîtriser.

Marc qui était sur un projet totalement différent (développement d'une application Android) a eu un peu plus de mal à trouver sa place mais a finalement pris en charge l'ancien rôle d'Adrien et Mehdi en reprenant les ébauches de structures existantes, et ainsi constituer un corps et une interface à l'application.

Jérémy quant à lui, a été très bénéfique à l'équipe puisqu'il manipulait des outils de cartographie et de frises chronologiques dans son précédent projet, et que précisément nous réfléchissions aux différentes APIs existantes afin d'en intégrer à notre application. Il a pu prendre en main cette partie du développement en intégrant à l'application une carte (voir 3.2.3.1 « La carte ») à partir de l'API Google Maps et une frise chronologique (voir 3.2.3.2 « La frise ») à partir de l'API SIMILE, outils qu'il utilisait déjà et qu'il a donc apportés au projet.

Nous avons également décidé, afin de tenter de corriger les problèmes d'organisation, de changer de chef de projet et ainsi de nommer Jérémy afin qu'Antoine puisse se consacrer à 100% aux différents aspects techniques qu'il maîtrisait le mieux ainsi qu'à la lourde tâche du transfert de connaissances en vue de la bonne intégration des nouveaux membres.

Malgré tous nos efforts, la complexité du sujet couplée aux problèmes rencontrés de restructuration, nous ne sommes jamais parvenus à régler ces problèmes d'organisation ; et bien que nos tuteurs et commanditaires soient assez satisfaits du travail que l'on a pu produire sur le plan technique, ils le sont nettement moins au sujet de notre gestion et organisation du projet.

Quoi qu'il en soit, ceci nous a rappelé, qu'un bon technicien n'est pas forcément un bon gestionnaire.

Toutes ces difficultés d'organisation nous ont finalement porté préjudices puisque nous ne serons très certainement pas prêts à temps et allons devoir rendre notre Mémoire avec du retard. Dans le même temps, ce retard nous a empêchés de constituer convenablement notre Cahier des charges techniques qui par conséquent, est relativement inexistant puisqu'il contient un Diagramme de Séquence détaillé et non de diagramme d'activité prévu initialement.

4. Conception et programmation

4.1. DIAGRAMMES DE SEQUENCES

Suite aux nombreux scénarios que nous avons produits, nous avons trouvé plus pertinent de ne faire qu'un seul Diagramme de Séquences (DS détaillé reprenant l'ensemble des scénarios nominaux afin de pouvoir visualiser la majorité des messages échangés entre les différents acteurs de notre système.

Voici donc notre DS découpé en deux parties afin qu'il apparaisse plus clairement. La première partie ne concerne que l'utilisateur et le système, et correspond à la phase de saisie d'un texte.

La deuxième partie quant à elle, démarre au lancement de l'analyse et concerne donc l'ensemble des acteurs.

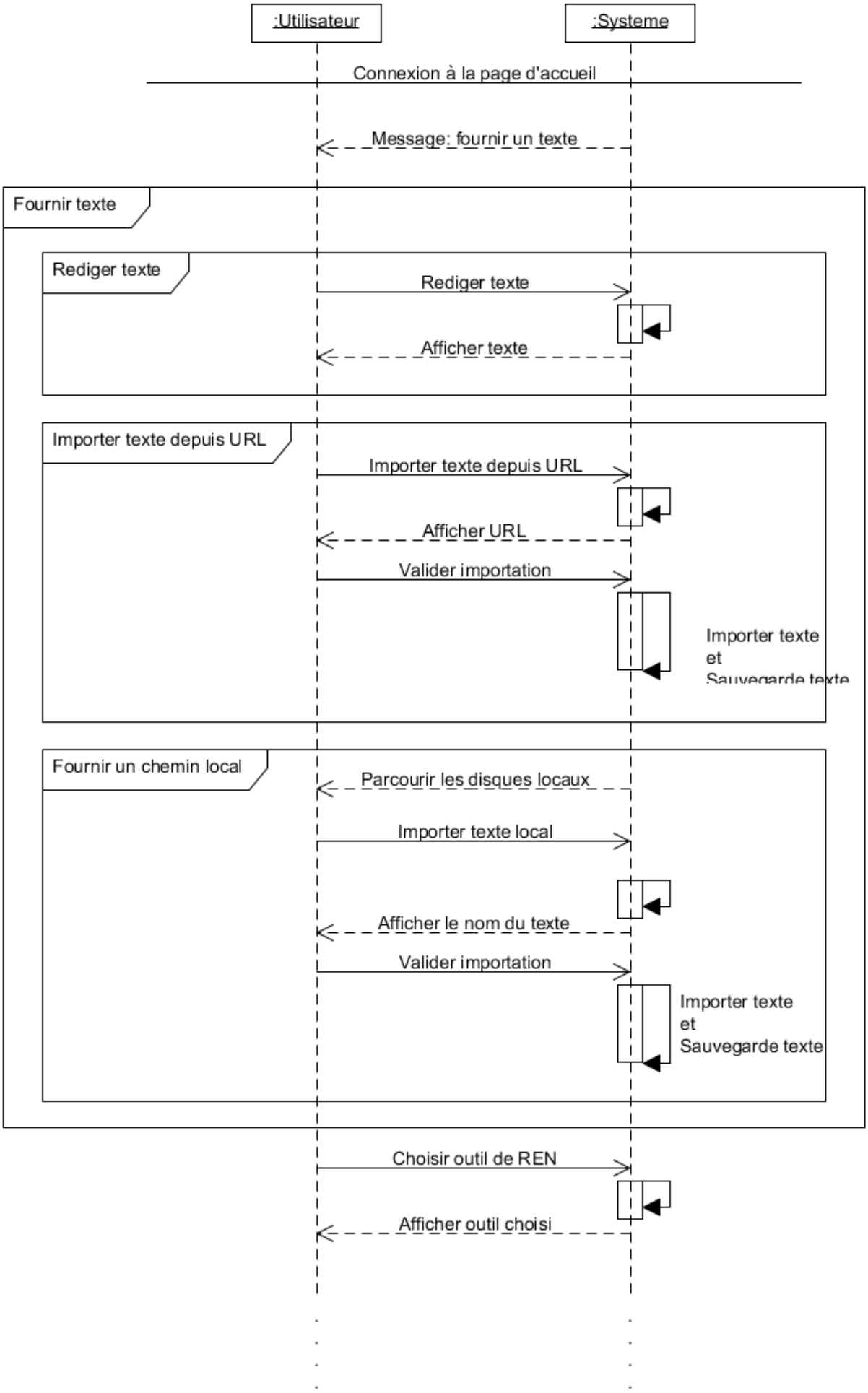

Figure 23 : Diagramme de Séquence détaillé - Partie 1

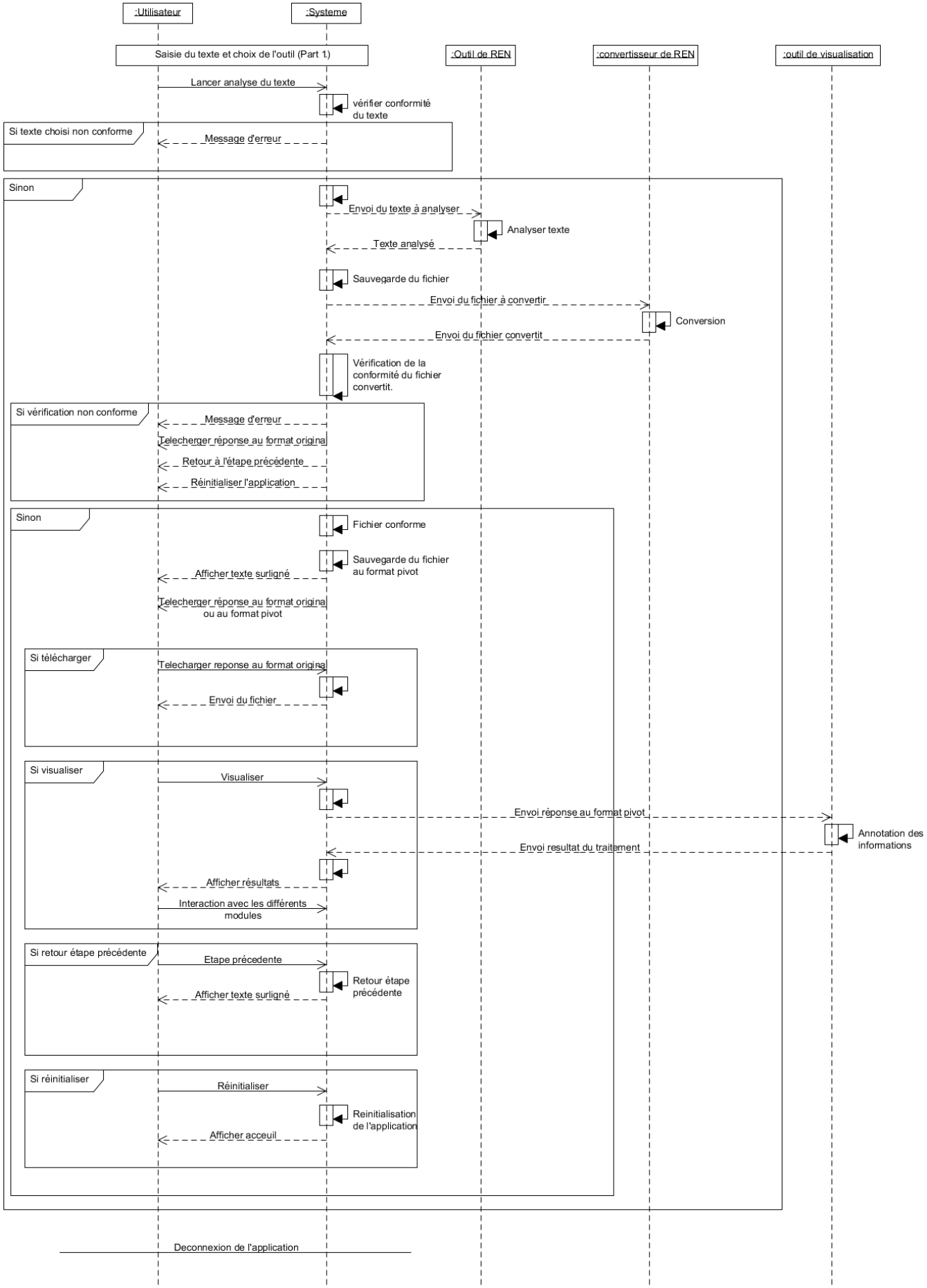

Figure 24 : Diagramme de Séquence détaillé - Partie 2

4.2. LANGAGES ET TECHNOLOGIES

4.2.1. Langages utilisés

CSS (*Cascading Style Sheets),* est un langage qui sert à décrire la présentation des documents HTML et XML Il permet d'appliquer un style aux pages web et de gérer séparément la forme du contenu.

Nous l'avons utilisé pour structurer la page, c'est-à-dire positionner les éléments où on le souhaite. Le CSS nous a permis de mettre en forme notre application.

HyperText Preprocessor, plus connu sous son sigle PHP (Acronyme récursif), est un langage de scripts libre principalement utilisé pour produire des pages Web dynamiques via un serveur HTTP, mais pouvant également fonctionner comme n'importe quel langage interprété de façon locale, en exécutant les programmes en ligne de commande.

PHP est un langage impératif disposant depuis la version 5 de fonctionnalités de modèle objet complètes. En raison de la richesse de sa bibliothèque, on désigne parfois PHP comme une plate-forme, et non plus comme un simple langage. (Source : http://fr.wikipedia.org/wiki/PHP).

Nous avons utilisé le PHP pour gérer les actions entre les différentes pages de l'application, et pour automatiser les tâches.

Quelques fois abrégé JS, JavaScript est un langage de programmation de scripts principalement utilisé dans les pages web interactives mais aussi côté serveur.

C'est un langage orienté objet à prototype, c'est-à-dire que les bases du langage et ses principales interfaces sont fournies par des objets qui ne sont pas des instances de classes, mais qui sont chacun équipé de constructeurs permettant de créer leurs propriétés, et notamment une propriété de prototypage qui permet d'en créer des objets héritiers personnalisés. (Source : http://fr.wikipedia.org/wiki/JavaScript).

Le Javascript nous a permis, entre autres, d'utiliser diverses API qui seront citées plus loin dans le rapport. De plus, ce langage nous a permis de dynamiser nos interfaces en utilisant la librairie JQuery citée au paragraphe suivant.

JQuery est une bibliothèque JavaScript libre qui porte sur l'interaction entre JavaScript (comprenant Ajax) et HTML, et a pour but de simplifier des commandes communes de JavaScript. La première version date de janvier 2006 (Source : http://fr.wikipedia.org/wiki/JQuery).

Nous avons utilisé JQuery afin de rendre nos interfaces dynamiques. Par exemple les onglets comprenant le texte saisi, les lieux, les dates ainsi que les thèmes sont interchangeables grâce à cette technologie.

L'Extensible *Markup Language* (XML, « langage de balisage extensible» en français) est un langage informatique de balisage générique qui dérive du SGML.

Cette syntaxe est dite « extensible » car elle permet de définir différents espaces de noms, c'est-à-dire des langages avec chacun leur vocabulaire et leur grammaire, comme XHTML, XSLT, RSS, SVG.

Elle est reconnaissable par son usage des chevrons (< >) encadrant les balises. L'objectif initial est de faciliter l'échange automatisé de contenus complexes (arbres, texte riche.) entre systèmes d'informations hétérogènes (interopérabilité) (Source : http://fr.wikipedia.org/wiki/Xml).

Nous avons utilisé le XML pour convertir les réponses obtenues grâce aux api, et pouvoir les manipuler en utilisant un langage de manipulation de données au format XML, le XPATH.

XPath est un langage (non XML) pour localiser une portion d'un document XML.

Initialement créé pour fournir une syntaxe et une sémantique aux fonctions communes à XPointer et XSL, XPath a rapidement été adopté par les développeurs comme langage d'interrogation simple d'emploi. (Source : http://fr.wikipedia.org/wiki/XPath).

Nous avons utilisé le langage XPath pour retrouver les valeurs souhaitées dans le fichier XML retourné par les services.

XSLT (*eXtensible Stylesheet Language Transformations*), défini au sein de la recommandation XSL du W3C, est un langage de transformation XML de type fonctionnel.

Il permet notamment de transformer un document XML dans un autre format, tels que PDF ou HTML, pour être affiché comme une page web. L'objectif principal est la transformation d'un document XML vers un autre schéma ou format (XHTML, XSL-FO, HTML, etc.).

Cependant, le langage XSLT permet aussi les transformations vers tout autre type de documents, au format texte ou dans un format binaire (bien que ceci ne soit pas originellement prévu par la recommandation XSLT). Étant données

deux entrées, un document XML à transformer et un document XSLT, un analyseur XSLT (*XSLT processor*) produit un fichier de sortie au format désiré. (Source : http://fr.wikipedia.org/wiki/Extensible_Stylesheet_Language_Transformations)

4.2.2. Logiciels utilisés

NotePad++ est un éditeur de texte générique codé en C++, qui intègre la coloration syntaxique de code source pour les langages et fichiers C, C++, Java, C#, XML, HTML, PHP, JavaScript, makefile, art ASCII, doxygen, .bat, ASP, Visual Basic/VB Script, SQL, Objective-C, CSS, Pascal, Perl, Python, R, MATLAB, Lua, TCL, Assembleur, Ruby, Lisp, Scheme, Properties, Diff, Small talk, PostScript et VHDL ainsi que pour tout autre langage informatique, car ce logiciel propose la possibilité de créer ses propres colorations syntaxiques pour un langage quelconque.

Ce logiciel, fondé sur la composante Scintilla, a pour but de fournir un éditeur léger (aussi bien au niveau de la taille du code compilé que des ressources occupées durant l'exécution) et efficace car il est également une alternative au bloc-notes de Windows (d'où le nom). Le projet est sous licence GPL.

(Source : http://fr.wikipedia.org/wiki/Notepad%2B%2B).

Microsoft Office Word (2013) est un logiciel de traitement de texte.

Nous l'avons utilisé lors de la rédaction de tous les documents écrits que nous avons eu à produire.

Microsoft Office Powerpoint (2013) est un logiciel de présentation.

Nous l'avons utilisé pour créer les diaporamas nécessaires pour les présentations orales.

Microsoft Office Excel (2013) est un tableur.

Ce logiciel nous a servi pour faire les différents plannings prévisionnels durant toute la durée du projet ou encore pour le planning final.

Mozilla Firefox est un navigateur internet personnalisable basé sur le moteur d'affichage libre de pages Web Gecko. Ce navigateur contient également un plug-in très intéressant lors de phase de programmation ou de test, *Firebug*, que nous avons également utilisé.

Chrome est un navigateur web développé par Google, fondé sur le projet libre Chromium fonctionnant sous Windows, Mac, Linux, Android et iOS. Google Chrome est identique à Chromium hormis un logo différent et quelques fonctionnalités en moins ou en développement (Source : http://fr.wikipedia.org/wiki/Google_chrome).

4.3. APIS

4.3.1. Définition

Une interface de programmation (*Application Programming Interface* ou API) est une interface fournie par un programme informatique. Elle permet l'interaction des programmes les uns avec les autres, de manière analogue à une interface homme-machine, qui rend possible l'interaction entre un homme et une machine. Du point de vue technique, une API est un ensemble de fonctions, procédures ou classes mises à disposition par une bibliothèque logicielle, un système d'exploitation ou un service. La connaissance des API est indispensable à l'interopérabilité entre les composants logiciels.

(Source : http://fr.wikipedia.org/wiki/Interface_de_programmation).

4.3.2. AlchemyAPI

AlchemyAPI permet d'extraire des données sémantiques sous forme de métadonnées comme des personnalités, des lieux, des sociétés ou des faits. Le résultat d'une analyse peut ensuite être transmis à un site web qui peut l'utiliser pour enrichir sa base de données ou pour traiter ses données, et les afficher à un utilisateur sous forme de cartes ou de frises.

(Source : http://www.alchemyapi.com/api).

4.3.3. TextRazor

Le service de TextRazor API fournit une analyse de certains passages de textes pour identifier les entités et les énoncés de faits cités par homonymie, et distinguer des chaînes de textes similaires. Elle utilise des machines avec des algorithmes d'apprentissage et un traitement du langage naturel, pour connecter un échantillon de textes avec une base de connaissances et 'identifier les éléments connus et leurs relations.

(Source : http://www.programmableweb.com/api/textrazor).

4.3.4. Google Maps v3 API

Google Maps est une API de Google permettant de géolocaliser des lieux sur une carte à l'aide de leur latitude et de leur longitude. Cette API permet de localiser tout type de données sur une carte (routière, satellite, mixte) à partir de son adresse postale. Cette API s'avère très utile pour proposer aux internautes une vision globale et géographique de données (membre d'une communauté, restaurants d'un quartier...). Les résultats sur la carte apparaissent sous la forme d'un marqueur (de la forme d'une goutte d'eau inversée) que l'on peut choisir de colorer en rouge par défaut.

(Source : http://www.dicodunet.com/definitions/google/api-google-map.htm).

4.3.5. SIMILE Timeline

SIMILE Projet a été créé par le *World Wide Web Consortium* (W3C) dans le but de chercher à améliorer l'interopérabilité entre les actifs numériques de schémas / vocabulaires / ontologies, de métadonnées et de services. Un défi majeur qui a été résolu, était de rendre interopérables les collections qui sont distribuées à travers les individus, les communautés et les magasins institutionnels - en s'appuyant sur les éléments d'actif de schémas / vocabulaires / ontologies, et les métadonnées détenues dans ces magasins.

C'est dans ce contexte qu'ont été créés les SIMILE Widgets, dont Timeline, un outil permettant de créer des frises chronologiques, faisait partie.

(Source : http://simile.mit.edu/wiki/SIMILE:About).

4.3.6. CHAP Link Library

CHAP Link Library est une bibliothèque de visualisation basé sur le Web pour l'affichage des graphiques, de réseaux et d'échéanciers.

Les outils sont développés comme les graphiques de visualisation de Google pour JavaScript. CHAP Link Library est développé par Almende dans le cadre du CHAP (Common Hybrid Agent Platform).

Nous avons utilisé CHAP pour gérer l'affichage des thèmes sur la dernière vue de notre application.

5. Perspectives et Bilan

5.1. BILAN

Ce projet, l'aboutissement de nos deux années passées en DUT Informatique, nous a permis d'exploiter une grande partie des connaissances acquises en cours.

Il a représenté une expérience de vie enrichissante tant par son aspect professionnel que par son aspect relationnel.

Il nous a également permis d'approfondir nos connaissances en matière de programmation web. Nous avons dans ce cadre, développé de nombreuses compétences techniques car nous avons utilisé plusieurs technologies que nous n'avions pas étudiées durant notre formation, comme par exemple le JavaScript ou encore le XML ainsi que de nombreuses API (GoogleMaps et AlchemyAPI notamment). Ce contexte nous a conduits à faire un travail de recherche personnel important.

Sur le plan relationnel, ce projet nous a tout d'abord montré l'importance d'une communication régulière avec les commanditaires et les tuteurs : en effet, cela a permis au groupe de garder clairement l'objectif en tête mais également de vérifier à chaque nouvel avancement si nos choix étaient pertinents et si des rectificatifs étaient nécessaires.

Nous avons également eu l'occasion d'être confrontés aux différents problèmes de restructuration qui apparaissent régulièrement dans le milieu professionnel. Cette situation particulière et relativement peu courante dans le cadre de projets de synthèse, a entrainé comme vue plus haut la reconstitution d'une équipe à partir de trois morceaux d'équipes. Il nous a fallu communiquer, nous transmettre les informations et nous réorganiser rapidement en milieu d'année, afin de ne pas mettre en péril le projet. Nous avons fait preuve à cette étape d'une motivation qui a permis l'intégration de deux membres nouveaux et l'avancée du travail. Nous sommes conscients de la rigueur et du sérieux dont a fait preuve notre chef de projet dans ce contexte malgré ses difficultés de gestion et d'organisation. Il était essentiel qu'il ait et conserve une vision globale du projet.

Comme nous l'avons vu, notre projet découle de la volonté de nos commanditaires de posséder un outil regroupant plusieurs outils de REN et ainsi disposé d'une base de test pour leurs propres travaux. De par sa nature, notre projet avait donc pour vocation de prouver la faisabilité d'un tel outil. Il nous a donc été demandé de développer notre application comme un prototype « jetable » qui servirait de base de travail a un éventuel futur projet. Ainsi, bien qu'il ait été prévu dans un premier temps de développer notre application selon l'architecture MVC (modèle vue contrôleur), cette optique a été abandonnée et laissée au rang des perspectives d'évolutions. Cependant, étant donné que l'application à produire n'était pas d'une taille très importante, nous avons conservé l'esprit du MVC lors du développement de notre outil et ainsi produit un code qui sans utiliser aucun Framework, respecte dans les grandes lignes ce concept de MVC. Le seul Framework que nous maitrisions, puisque nous l'avions étudié durant notre formation, est le Framework pédagogique développé par M. Patrick Etcheverry. Malheureusement, il est peu adapté à une utilisation de l'Ajax et nous a donc tout de suite été déconseillé par nos tuteurs qui en avaient fait l'expérience l'année précédente. Afin d'utiliser un Framework, il aurait donc fallu apprendre à en maîtriser un autre, mais cela n'a pas été possible faute de temps.

5.2. PERSPECTIVES

Il est envisageable que cette application soit développée en programmation orientée objet et que des classes soient faites, car pour le moment, nous mélangeons programmation procédurale (fonction Php « classique ») et programmation objet avec l'utilisation des outils de REN (classes Php fournies par les équipe de développement respectives des outils utilisés) ou la manipulation du XML (objet Php DOMDocument et DOMXpath) ainsi que l'utilisation des APIs JavaScript.

Ainsi voici une proposition d'évolution vers une application orientée objet à partir de ces diagrammes de classe.

Afin de représenter l'application sous forme de classe, nous avons élaboré une première version.

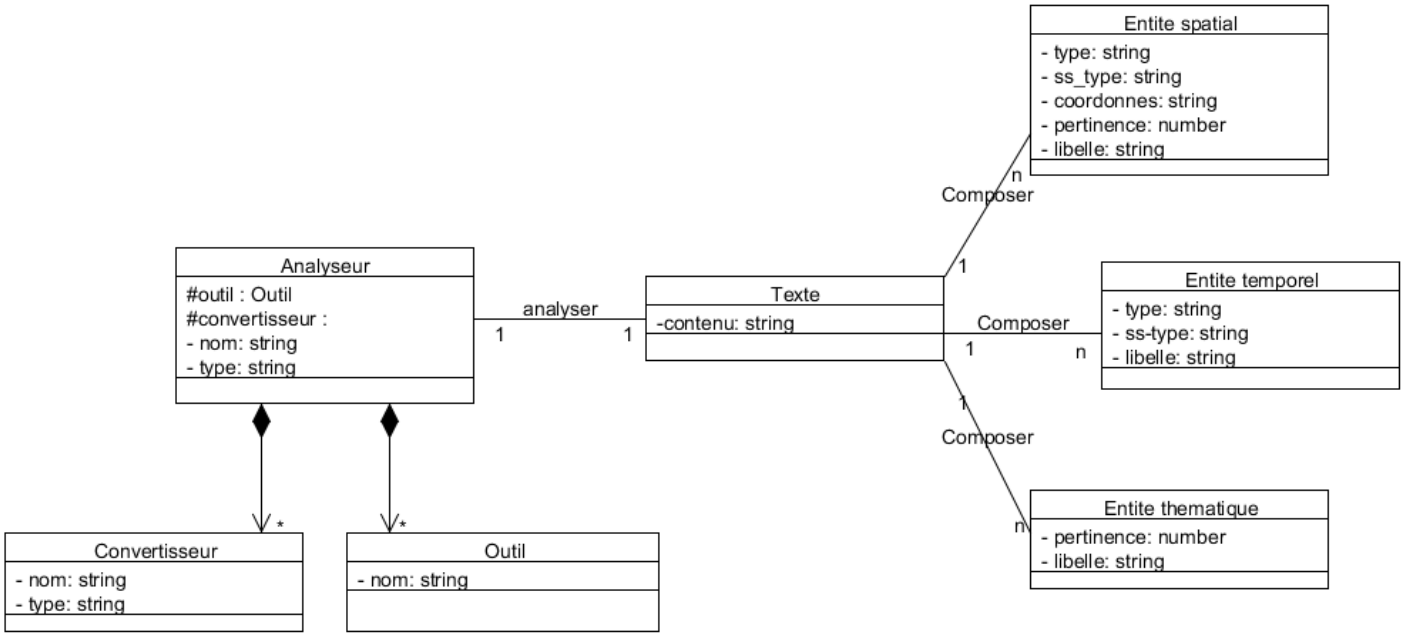

Figure 25 : Diagramme de classe V1

Ainsi, un Analyseur se compose d'un convertisseur, caractérisé par un nom et du type de format accepté pour la conversion, et d'un outil de détection d'entités caractérisé par son nom.

Un Analyseur analyse un texte.

Un texte est analysé par un analyseur et est composé d'une ou plusieurs Entités (Entités spatial, Entités temporel, Entités thématique).

Puis au cours du semestre 4, nous avons eu un cours sur les design patterns ce qui nous a semblé cohérant avec les besoins de notre application.

Nous avons donc élaboré une deuxième version du diagramme de classe.

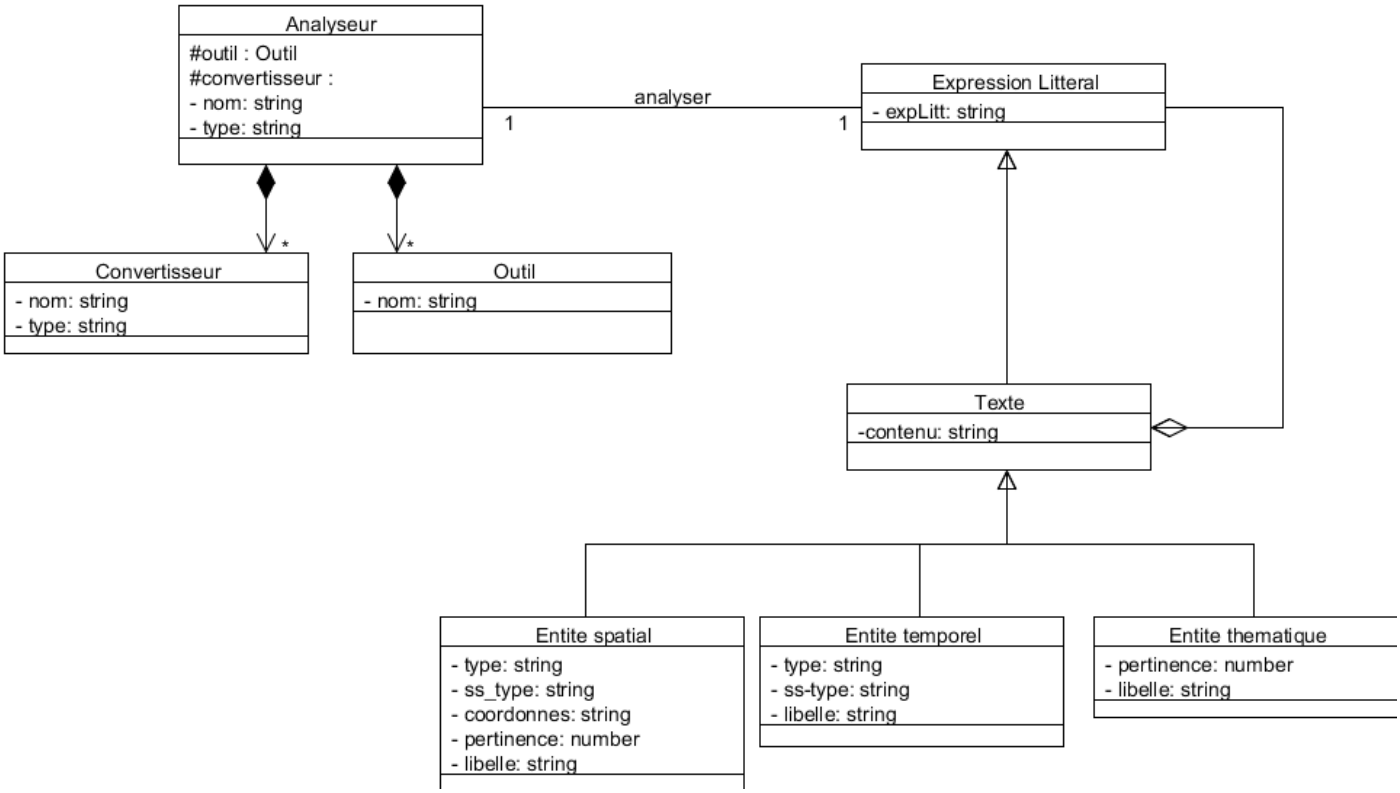

Figure 26 : Diagramme de classe V2

Dans cette version 2 nous avons décidé de le modéliser par le modèle composite des design patterns.

Nous avons gardé la même structure que la version 1 à la différence près que nous avons introduit une classe « Expression Littéral » dont la classe texte hérite. De plus les entités deviennent des spécifications de la classe Texte.

Nous avons pas eu l'occasion de proposer ces diagramme à un enseignant ce qui signifie que nous ne sommes pas certain de leur justesse.

Pour finir, nous pouvons dire que cette expérience fut certes par moments difficile, mais elle nous a appris à gérer les problèmes qui se présentaient, aussi bien techniques qu'humains.

6. Abstract

For several years, the research laboratory of the UPPA (University of Pau and Adour's Lands) focused their work on spatial, temporal and thematic information retrieval. They decided to create an automated information retrieval from documents collections (like texts from books) or databases (like website's ones)

In this context, they asked us to develop a new application which includes many free existing textual lexical, syntactic and semantic analysis tools.

The purpose of the application is to recognize different types of data, such as spatial (like "Bayonne", "New York", "Brazil") and temporal entities (like "14/03/2014"). We also need to group them into themes (like "Sport", "City", "Actor"). Then, this data will be converted into visual information like timelines and maps.

For this, we use various tools, including the Google Maps v3 API (Application Programming Interface) and the Chap Link Library Timeline API, but we must also invoke web services to be able to recognize these types of entities. As our application is a mash up of many tools, it can be easily modified by adding or deleting of some tools.

After a text analyze, the application gets the response of the chosen tool and highlights all the entities which are found in the user's text. For each type of entity, one specific color is used.

Then, the application puts markers on a map and a timeline on all of places and dates which are identified on the response.

Authors: Antoine Catanese, Jeremy Dulon, Marc Lagreou, Baptiste Sabuco

Keywords: Computer Science, IT, geo-location, spatio-temporal data

7. Glossaire

Ajax : Ajax (Asynchronous Javascript and XML) est la combinaison de technologies telles que JavaScript, CSS, XML, dans le but de réaliser des applications internet riches (Rich Internet Application ou RIA) sans recharger toute la page. Cela offre une maniabilité et un confort d'utilisation supérieurs.

API : Application Programming Interface ou Interface de Programmation (bibliothèque de fonctionnalités mises à disposition afin de faciliter la programmation).

CSS : CSS (Cascading Style Sheets) est un langage qui sert à décrire la présentation des documents HTML et XML. Il permet d'appliquer un style aux pages xHTML et de gérer séparément la forme du contenu.

Framework : Espace de travail modulaire qui regroupe un ensemble de bibliothèques et de conventions (ensemble de composants logiciels) permettant le développement rapide d'applications.

HTTP : L'*HyperText Transfer Protocol,* ou HTTP, est un protocole de communication entre un client et un serveur.

JavaScript : JavaScript est un langage de programmation principalement utilisé dans les pages Web interactives côté client.

Media queries : Une media query (ou requête média) consiste en un type de média, et une expression CSS qui met à profit les particularités des supports multimédias (largeur, hauteur, couleurs).

Métadonnées : Les métadonnées servent à décrire ou à définir une autre donnée (photo texte son ou vidéo). Elles sont définies dans le cadre du modèle ressource description Framework (RDF).

PHP : PHP (Hypertext PreProcessor) est un langage de script libre de droit principalement utilisé pour produire des pages Web dynamiques via un serveur HTTP, mais pouvant également fonctionner comme n'importe quel langage interprété côté serveur.

SOAP : Le SOAP, pour *Simple Object Access Protocol*, est un protocole qui permet les messages entre objets. Il peut autoriser un objet à invoquer des méthodes d'objets qui sont présentes physiquement sur un autre serveur.

RDF : RDF (Resource Description Framework) est un modèle de graphe destiné à décrire de façon formelle les ressources Web et leurs métadonnées. RDF est une surcouche basée sur XML.

REST : Le REST, ou *REpresentational State Transfer*, est une architecture web à la manière de SOAP, à la différence qu'il s'appuie sur des ressources et non des méthodes.

XML : XML (Extensible Markup Language) est un métalangage (formalisme conçu pour décrire un langage) informatique de balisage générique. Il sert essentiellement à stocker et/ou transférer des données de type texte structurées en champs arborescents

8. Webographie

API Google Maps :

https://developers.google.com/maps/documentation/javascript/v2/

https://developers.google.com/maps/documentation/javascript/reference

API Chap Links :

http://almende.github.com/chap-links-library/

JavaScript :

http://javascript.developpez.com

JQuery :

http://jquery.com/

PHP :

www.php.net

SIMILE :

http://simile.mit.edu/

http://www.simile-widgets.org/timeline/

Wikipedia:

http://fr.wikipedia.org/wiki/Service_Web

http://fr.wikipedia.org/wiki/PHP

http://fr.wikipedia.org/wiki/JavaScript

http://fr.wikipedia.org/wiki/JQuery

http://fr.wikipedia.org/wiki/Xml

http://fr.wikipedia.org/wiki/XPath

http://fr.wikipedia.org/wiki/Resource_Description_Framework

http://fr.wikipedia.org/wiki/SPARQL

http://fr.wikipedia.org/wiki/Interface_de_programmation

http://www.dicodunet.com/definitions/google/api-google-map.htm

http://fr.wikipedia.org/wiki/NetBeans

http://fr.wikipedia.org/wiki/Notepad%2B%2B

http://www.w3.org/RDF/Validator

XML/XSLT/XPATH :

http://slaborie.perso.univ-pau.fr/index.php/fr/enseignement/xml

Table des illustrations

Figure 1 : Logo de l'UPPA 4
Figure 2 : Logo du LIUPPA et de l'équipe T2i 4
Figure 3 : Copie d'écran GeoText2Map 5
Figure 4 : Copie d'écran Geo media Tagger 5
Figure 5 : Logo de application 6
Figure 6 : tableau d'outils de R.E.N. Open Source 7
Figure 7: Exemples de formats XML 7
Figure 8 : Schéma simplifié du déroulement général de l'application V1 8
Figure 9 : Schéma simplifié du déroulement général de l'application V2 9
Figure 10 : Première vue de l'application 10
Figure 11 : Deuxième vue de l'application 10
Figure 12 : Troisième vue de l'application 11
Figure 13 : Logo de l'AJAX 12
Figure 14: tableau de Framework MVC utilisant AJAX 12
Figure 15 : DCU V1 13
Figure 16 : DCU V2 14
Figure 17: Maquette VUE 1 - Scénarios réels - CU « Fournir un texte a analyser » 20
Figure 18 : Schéma détaillé du déroulement général de l'application V3 21
Figure 19 : Schéma produit à parir d'une entité spatial dans le format pivot 26
Figure 20: Exemple de fonctionnement de la carte Google Maps 27
Figure 21 : Arborescence d'une entité spatiale dans le format pivot 28
Figure 22 : Exemple de fonctionnement de SIMILE Timeline 29
Figure 23 : Diagramme de Séquence détaillé - Partie 1 35
Figure 24 : Diagramme de Séquence détaillé - Partie 2 36
Figure 25 : Diagramme de classe V1 43
Figure 26 : Diagramme de classe V2 44