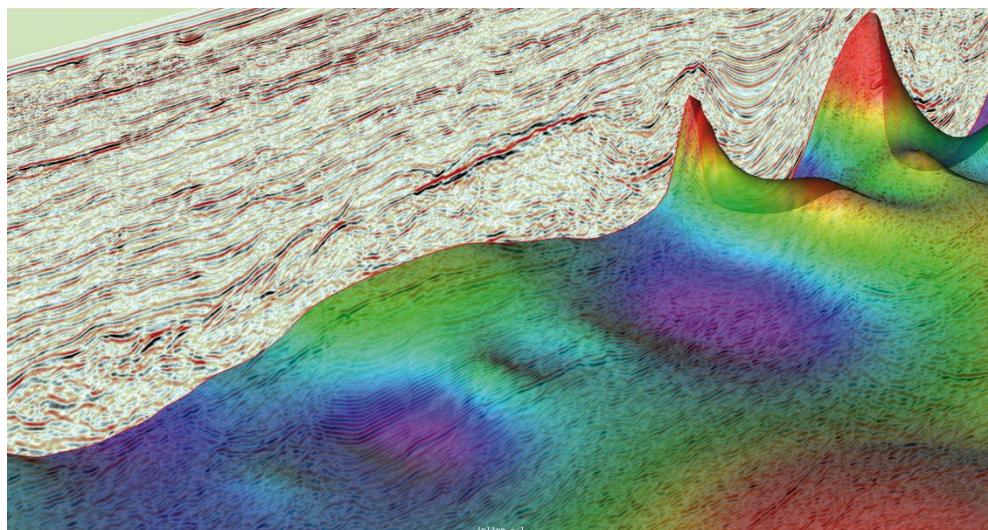


Internship report

3/08/2015 - 28/09/2015

Catherine Lellouche



SOMMAIRE

Chapitre 1 - Présentation.....	5
1.1 - Avant-propos	6
1.2 - Mon profil et mon parcours.....	7
1.3 - Abstract.....	8
1.4 - Remerciements	9
1.5 - Le cadre général du projet	10
1.5.1 Un cadre de développement professionnel : le service des applications Interactives de CGG	10
1.5.2 Les interlocuteurs et acteurs du projet.....	12
1.5.3 Les outils de développement	12
1.5.4 Gestion du projet en mode SCRUM	14
1.5.5 La démarche d'analyse avec UML.....	15
Chapitre 2 - Cahier des charges.....	16
2.1 - Présentation préliminaire du projet	17
2.1.1 Le métier de CGG.	17
2.1.2 Le client.....	17
2.2 - Le cahier des charges	18
2.2.1 Définir le contenu du backlog produit : les USER STORIES.....	18
2.2.2 Prioriser le backlog produit.	20
2.2.3 Définir le sprint backlog.....	22
Chapitre 3 - Analyse.....	24
3.1 - Les cas d'utilisation (CU) et le Diagramme de Cas d'Utilisation (DCU).....	26
3.1.1 Représentation graphique	26
3.1.2 Le DCU macroscopique	27
3.1.3 DCU Display data in cross plot and define display settings	29
3.1.4 Fiche de description textuelle d'un cas d'utilisation	30
3.1.5 Diagramme d'activité de « Import data »	32
3.2 - Les maquettes	33
3.2.1 La page d'accueil et de connexion	34
3.2.2 La boîte de dialogue « Sign up ».....	34

3.2.3	La boîte de dialogue « Import Data ».....	35
3.2.4	La page « Display datas (in cross plot) »	35
3.2.5	La page « Display datas (values) ».....	36
3.2.6	La page « Display Options »	36
3.2.7	La page « Edit Options ».....	37
3.2.8	La page « Rename»	37
3.3	- Le diagramme de navigation	38

Chapitre 4 Conception de la Base de Données 39

4.1	- La démarche utilisée.....	40
4.2	Le Modele Conceptuel des Donnees	41
4.2.1	Le dictionnaire des données.	41
4.2.2	Le modèle conceptuel des données.	43
4.2.3	Les règles de gestion.	44
4.2.4	La matrice des dépendances fonctionnelles.	45
4.2.5	Le Graphe des Dépendances Fonctionnelles.....	46
4.3	- Le diagramme de classes (DCL)	48
4.3.1	DCL entités : l'utilisateur et ses options d'affichage du cross plot.....	50
4.3.2	Le DCL entités : le fichier (file).....	51
4.3.3	Diagramme de classes Entités complet :	52
4.4	- Le modèle physique des données	53
4.5	- Les procédures stockées	54
4.6	- Le code SQL (LDD) de création de la base de données	56

Chapitre 5 - Conception de l'application..... 61

5.1	- Les diagrammes de séquence détaillés.....	62
5.1.1	Le Diagramme de Séquence Détailé « IMPORT DATA »	63
5.1.2	Le Diagramme de Séquence Détailé « DISPLAY DATA ».....	64
5.2	- Les diagrammes de classes participantes	65
5.2.1	Le Diagramme de classes participantes « IMPORT DATA ».....	65

Chapitre 6 - Développement..... 66

6.1	- Données d'entrée.....	67
6.2	- Interface : Page « Display Data ».....	68
6.2.1	Page HTML principale	68
6.2.2	Feuille de style	69
6.2.3	Classes Javascript.....	70
6.3	- Controller	76

6.4 - Entities/Models et DAO	78
6.4.1 Classes de base commune	78
6.4.2 Exemple d'entité avec UserDisplayOptions	79
6.4.3 DAO d'import des données dans la base de données Topaz	80
Chapitre 7 - Déploiement.....	81
7.1 - Le diagramme de déploiement	82
7.2 - Le déploiement	83
Chapitre 8 - Conclusion.....	84
Chapitre 9 - Annexes.....	85
9.1 - Correspondances Projet/Reac.....	86
9.2 - Synthèse des outils utilisés.....	87
9.3 - Le métier de CGG	88
9.3.1 - EXPLORATION SISMIQUE : but et fondamentaux	89
9.3.2 - ACQUISITION DES DONNEES	90
9.3.3 - TRAITEMENT Des données	95
• Vitesse de propagation des ondes différente selon la roche :	95
• Propriétés des ondes :	96
• Réflexion et transmission :	97
• L'ordre de couverture ou FOLD COVERAGE :	97
• Le « Bruit ».....	99
• La trace	99
• Les datas enregistrées et traitées.....	100
9.3.4 - INTERPRETATION	100
9.4 - Le code de l'application	101
9.4.1 La feuille de style css complète.....	101
9.4.2 Le code complet de création et d'affichage d'un cross plot.....	104

CHAPITRE 1 - PRESENTATION

1.1 - AVANT-PROPOS

Ce rapport est la synthèse de ma première expérience professionnelle dans un service informatique d'ingénierie logicielle dans lequel j'ai mis à l'épreuve mon apprentissage chez M2i.

Il a pour but de retracer la réflexion que j'ai menée pour comprendre la problématique qui m'était posée par l'entreprise, et de vous montrer la démarche, les méthodes et les outils que j'ai utilisés pour y répondre, pour finalement aboutir à la conception de l'application et au codage.

Par ailleurs, j'ai eu l'opportunité de travailler sur ce projet en mode scrum.
Ce rapport tente donc de refléter cette démarche évolutive et itérative conjointement à l'approche de l'analyse et de la conception avec UML.

C'est avec plaisir que je reprends ces citations que notre formateur chez M2i, M. Pascal Buguet, nous a fait partager car leur pertinence me semble évidente aujourd'hui :

"*La plus grande attention doit être portée à la compréhension du problème, faute de quoi l'algorithme n'a aucune chance d'être correct*". Denis Lapoire

"*C'est toujours l'impatience de gagner qui fait perdre*", Louis XIV cité dans "L'immortel" de FOG.

J'écoute et j'oublie.
Je lis et je retiens.
Je fais et j'apprends.

(Proverbe chinois)

J'ajoute deux autres citations qui ont guidé ma pensée :

"*Il semble que la perfection soit atteinte, non quand il n'y a plus rien à ajouter, mais plus rien à retrancher*" Antoine de Saint-Exupéry

"*La créativité c'est relier les choses entre elle*" Steve Jobs

1.2 - MON PROFIL ET MON PARCOURS.

Après des études de commerce international, où le langage et la technique se mêlaient, j'ai étudié la vente et la négociation commerciale au cours du Cycle Supérieur des Forces de Vente de la CCI de Paris. C'était un parcours très enrichissant auprès de formateurs issus du monde professionnel qui m'ont fait découvrir l'intérêt de ce métier quand il est bien fait.

J'ai exercé pendant plusieurs années ma profession commerciale mais toujours en choisissant des postes où la technique avait une large part.

Petit à petit, dans mes choix de postes ou à travers les tâches qu'on me confiait, j'ai de plus en plus divergé vers des responsabilités connexes à mon métier et qui me rapprochaient de l'informatique au service de la fonction commerciale.

Ainsi, j'ai souvent été responsable des outils de communication multimédia, en relation avec nos fournisseurs, des professionnels de l'informatique.

J'ai notamment été en charge, en tant que maître d'ouvrage, de la création d'un site internet pour la réservation de 10000 logements étudiants, de la création d'un serveur vocal interactif et d'une application dédiée à la gestion d'un parc de logements.

Nous avons si bien collaboré avec mes "fournisseurs", une petite SSII basée à Paris, qu'ils m'ont proposé de les rejoindre pour commercialiser leurs prestations.

Cette immersion dans leur société a complété ma vision des métiers de l'informatique et m'a définitivement convaincue que je peux m'épanouir dans cette orientation.

Puis j'ai beaucoup échangé avec un membre de ma famille, ingénieur de développement logiciel, qui m'a encouragée à franchir le pas et à me former au métier de concepteur développeur informatique, persuadé que j'avais les aptitudes pour réussir.

1.3 - ABSTRACT

CGG est une entreprise de services géophysiques fondée en 1931 et basée en France. Elle est la première compagnie mondiale de services et de produits géophysiques pour l'ensemble de l'industrie pétrolière et gazière.

Les activités de CGG vont de l'acquisition (sur terre, en eaux peu profondes, en eaux profondes), au traitement et à l'interprétation de données sismiques.

CGG emploie environ 9 800 personnes dans le monde entier.

Le sujet du stage constitue une expérimentation de la visualisation de données sismiques sur un client léger, via un navigateur Web, en exploitant une bibliothèque graphique développée en Javascript 5 fournie par la société INT.
En effet, toutes les applications développées à ce jour sont des applications desktop.

L'application Web **TOPAZ** doit permettre aux géophysiciens de visualiser sur un navigateur Web dans des cross plots les données sismiques issues des acquisitions des signaux, afin de leur permettre de les analyser et de repérer des anomalies.
Pour se faire, ils doivent pouvoir configurer l'affichage des cross plot en fonction de leur besoin d'analyse.

Les données à afficher proviennent de l'acquisition de signaux (enregistrements d'ondes acoustiques) faite par CGG sur le terrain et sont disponibles soit via des datas stores, soit dans des copies partielles disponibles sur les ordinateurs des géophysiciens à des fins d'expérimentation.

CGG (Compagnie Générale de Géophysique) is a French-based geophysical services company founded in 1931. CGG merged with Veritas DGC Inc. in January 2007 and made the acquisition of Fugro's Geoscience Division. CGG employs over 9 800 people around the world.

CGG is the world's leading geophysical services and products for the global oil and gas industry. In fact, about 95% of the activity is directed towards these areas.

CGG activities range from acquisition on land/shallow water, deep water, seabed, borehole and near-surface, to processing and interpretation of seismic data.

CGG has been a pioneer in the advancement of geophysics since the original company was created in 1931. Since then, the company has progressed technologically, operationally and financially.

The Subsurface Imaging department, where I do my Internship, is a research & development department where employees work on creating and improving Interactive application for geophysicist to visualize seismic data.

The aim of my project is to develop a "Web Viewer of seismic data" by using a JavaScript Library for building data visualization applications provided by INT Company.

1.4 - REMERCIEMENTS

Je remercie l'entreprise CGG qui m'a accueillie durant ce stage et plus particulièrement Mrs Gaël Youinou, Stéphane Bernard et Eric Smith qui m'ont accordé leur confiance en me permettant de travailler au sein de l'équipe des applications interactives.

Je remercie tout particulièrement M. Eric Smith qui a supervisé mon stage quotidiennement, pour le temps qu'il a pris pour me répondre et le savoir qu'il m'a transmis.

Vous m'avez réservé un excellent accueil et m'avez attribué un sujet très enrichissant. J'ai pu apprendre beaucoup grâce à vous, aussi bien techniquement que grâce à l'application des méthodes et des bonnes pratiques professionnelles.

Ce stage m'a permis de me conforter dans mon projet de reconversion professionnelle et je vous sais gré de l'avoir renouvelé et de me permettre ainsi de poursuivre le projet tout en continuant à me perfectionner.

Je remercie également mon formateur chez M2i, M. Pascal Buguet, pour sa grande disponibilité, la qualité et la richesse de son enseignement et son implication dans notre réussite.

1.5 - LE CADRE GENERAL DU PROJET

1.5.1 Un cadre de développement professionnel : le service des applications Interactives de CGG

J'ai réalisé mon stage au sein du département des applications interactives de l'entreprise CGG.

Au sein de la division R&D « Subsurface Imaging » (Imagerie du sous-sol), le département des applications interactives a pour mission de fournir aux géophysiciens un ensemble d'outils logiciel leur permettant de visualiser les données avant et / ou après traitement « batch ».

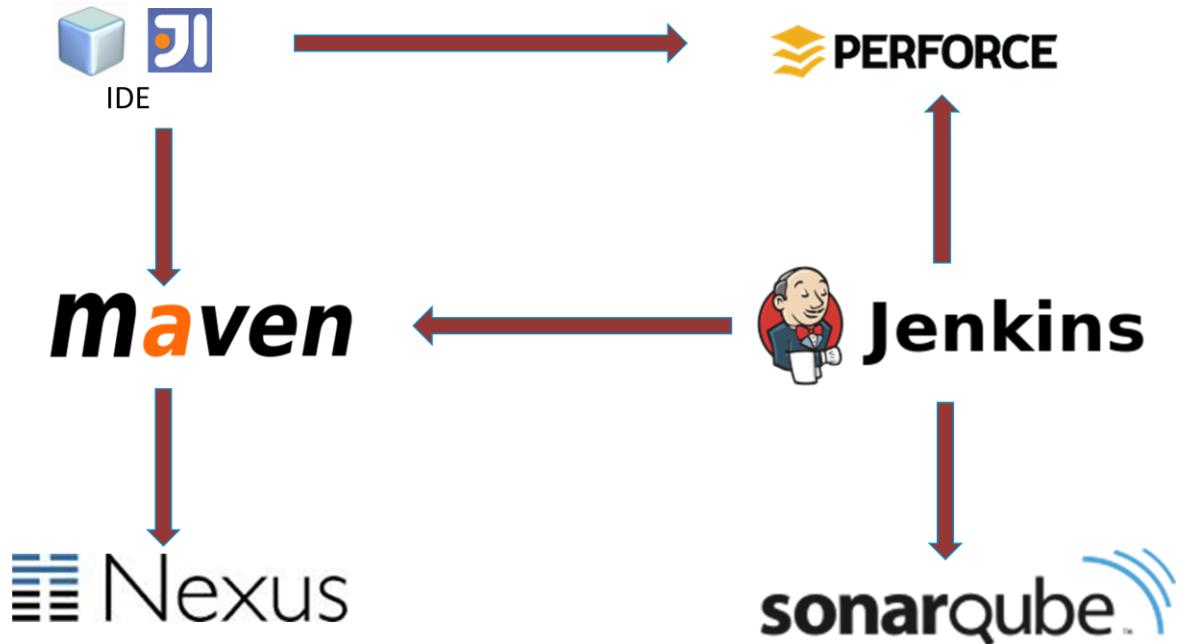
Ces outils leur permettent aussi d'expérimenter et d'ajuster les paramètres de leur séquence de traitement sur une petite quantité de données puis de relancer le traitement par lot sur la totalité des données du projet avec les nouveaux paramètres.

Les méthodologies et les bonnes pratiques sont la règle au sein du département des applications interactives.

Elles impliquent notamment une gestion de projet en mode SCRUM, l'application des bonnes pratiques, un environnement de développement normalisé et sécurisé, visant à garantir l'uniformisation des pratiques et la qualité de code afin de faciliter la maintenance des logiciels développés.

Environnement de développement	
Station de travail Linux	Pour le développement
Station de travail Windows 7	Pour la bureautique et les test
IntelliJ IDEA	EDI (environnement de développement intégré) ou en anglais <i>IDE (Integrated Development Environment)</i>
Maven	Gestionnaire de dépendances
Perforce	Gestion de configuration logicielle
Jenkins	Intégration continue
SONAR	Analyse de la qualité de code
JIRA Agile	Gestion de projet Agile en mode collaboratif
Confluence	Logiciel de travail collaboratif

Le diagramme suivant illustre l'interaction entre les divers outils de l'atelier logiciel :



1.5.2 Les interlocuteurs et acteurs du projet

Réalisatrice de la conception et du développement de l'application: Catherine Lellouche

Chef de projet, Scrum Master, et mon tuteur de stage : Eric Smith – Software Advisor.
Il est responsable de l'organisation de mon travail et valide mes choix.

Participants consultés pour la partie métier :
Stéphane Baris, de la Division Land, a joué le rôle de Product Owner.

Les interlocuteurs qui doivent être informés de l'évolution du projet :
M. Stéphane Bernard, Software Supervisor du département,
M. Gaël Youinou, Development Director du département.

1.5.3 Les outils de développement

Pour l'interface graphique :

Le projet consiste à explorer et exploiter une bibliothèque graphique développée en JavaScript 5, dédiée à la sismique, [GeoToolkit-js 2.0](#), dont l'entreprise INT est sous-traitante, et qui demeure inutilisée jusqu'ici.



INT fournit des composants logiciels graphiques pour la visualisation de données à la plupart des sociétés de services pétrolières et gazières.

"GeoToolkit-js delivers high-performance graphics capabilities for building data visualization applications needed in the Oil and Gas Exploration and Production industry. A bundled component suite consisting of Carnac, Gauges, Widgets and an optional combination of Seismic, WellLog, WellSchematic and Carnac 3D components, GeoToolkit-js allows developers to take advantage of high-level tools to rapidly deploy sophisticated data visualization and analysis technology while taking advantage of the simplicity of the JavaScript language."



Les pages Web statiques sont développées en HTML 5, et CSS.

Les parties dynamiques côté client sont donc développées en Javascript 5.

De l'AJAX (Asynchronous Javascript and XML) est utilisé pour exécuter des requêtes asynchrones vers le serveur Web, pour l'envoi et le chargement des données.

Côté serveur d'application Web:

Le serveur est développé en Java à partir du serveur d'applications Jetty.

Côté base de données :

L'entreprise m'a laissé le choix du SGBDR.

Dans le cadre du stage, nous avons donc sélectionné MySQL pour capitaliser sur ma connaissance de cet environnement acquise chez M2i, et afin d'optimiser les délais de développement.

1.5.4 Gestion du projet en mode SCRUM

La méthode Scrum est une approche produit basée sur le **Manifeste pour le développement Agile de logiciels** :

« Nous découvrons comment mieux développer des logiciels par la pratique et en aidant les autres à le faire.

Ces expériences nous ont amenés à valoriser :

- Les individus et leurs interactions** plus que les processus et les outils
- Des logiciels opérationnels** plus qu'une documentation exhaustive
- La collaboration avec les clients** plus que la négociation contractuelle
- L'adaptation au changement** plus que le suivi d'un plan

Nous reconnaissions la valeur des seconds éléments, mais privilégiions les premiers. »

Source : <http://www.agilemanifesto.org>

Cela signifie notamment :

- Privilégier le travail en équipe et les échanges au cours des réunions scrum,
- Des livraisons intermédiaires du produit ou de fonctionnalités opérationnelles,
- Un feedback direct sur le produit ou sur la fonctionnalité livrée grâce aux réunions qui incluent le client ou le Product Owner,
- Opérer des changements de direction si c'est souhaité par le client.

Mon projet a été intégré au début d'un sprint d'un projet en cours.

Mon tuteur de stage, Eric Smith, est Scrum Master et j'intervenais en tant que team member.

Mon projet s'est ainsi déroulé sur 3 sprints :

- 1 sprint de 3 semaines du 3 août au 21 août 2015,
- 1 sprint de 3 semaines du 24 août au 11 14 septembre 2015,
- 1 sprint de 2 semaines du 15 septembre au 28 septembre 2015.

Dans ce premier sprint, mon tuteur de stage a établi un backlog un peu particulier en termes de contenu puisqu'il comprenait des stories fonctionnelles et non fonctionnelles.

Il est détaillé dans le chapitre *Cahier des charges : Topaz-Sprint n°1*

1.5.5 La démarche d'analyse avec UML.

En tant que concepteurs/conceptrices développeurs logiciel, notre question fondamentale est :

Comment passer des besoins des utilisateurs au code de l'application ?

La modélisation grâce à l'UML nous permet de répondre à cette question.

J'ai adopté la démarche de modélisation d'une application Web basée sur l'utilisation de l'UML développée dans l'ouvrage de Pascal Roques UML2 – Modéliser une application Web (Editions Eyrolles).

Je l'ai enrichie avec le cours de M. Pascal Buguet, mais aussi adaptée par rapport au mode de gestion de projet Scrum de notre équipe.

Durant la phase d'analyse de mon application, les diagrammes réalisés m'ont permis de formaliser les besoins des utilisateurs définis avec le Product Owner et d'échanger avec les différents acteurs du projet à des fins de validation.

Durant la phase de conception de l'application, d'autres diagrammes, comme les diagrammes de séquence détaillés notamment, m'ont permis d'identifier les interactions de l'acteur avec les différentes couches du système (interface homme machine –IHM-, contrôleur, et entités - couche métier), selon le design pattern ECB (Entity - Control - Boundary) qui a été utilisé dans ce projet.

Cette étape m'a permis d'aboutir à l'architecture de l'application et enfin au code de l'application.

CHAPITRE 2 - CAHIER DES CHARGES

2.1 - PRÉSENTATION PRÉLIMINAIRE DU PROJET

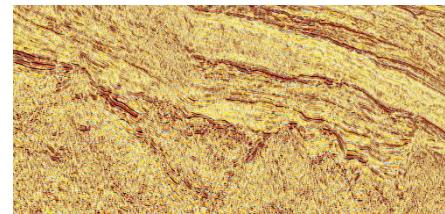
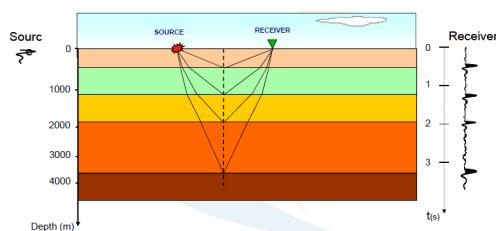
2.1.1 Le métier de CGG.

La CGG explore le sous-sol afin de fournir à ses clients, principalement l'industrie pétrolière, des images du sous-sol permettant de connaître la nature des couches géologiques et de déterminer la présence de nappes de pétrole.

En résumé, le métier de CGG comprend donc :

- l'acquisition de données par la mesure des signaux (ondes acoustiques). Des signaux sont émis à la surface de la terre, pour la faire trembler, puis mesurés par des capteurs et enregistrés.
- le traitement des ces signaux (traitement visant à les épurer du « bruit » parasite par exemple) et leur transformation en image la plus juste possible afin d'être interprétée.

(Pour une présentation plus détaillée et imagée, cf les annexes « Présentation du métier de CGG »)



2.1.2 Le client

Le département des Applications Interactives de CGG, au sein duquel je fais mon stage, travaille au développement et à l'évolution de logiciels internes destinés aux géophysiciens de CGG pour le traitement et la visualisation des données sismiques.

En effet, les géophysiciens qui travaillent sur le terrain à l'acquisition de données sismiques ou au traitement des données (data processing) ont besoin de contrôler la qualité de l'acquisition des données et la qualité des signaux eux-mêmes.

L'application sur laquelle je travaille est donc une demande du département des Applications Interactives lui-même, dans le cadre de sa mission d'évolution des outils logiciels mis à la disposition des géophysiciens.

2.2 - LE CAHIER DES CHARGES

2.2.1 Définir le contenu du backlog produit : les USER STORIES

Nom du projet : **TOPAZ**

Méthodologie de gestion de projet : SCRUM .

Mon tuteur de stage dirige les réunions.

L'outil JIRA permet de gérer les projets en mode Scrum en mode partagé au sein de l'équipe.

Première étape : constitution du backlog produit.

J'ai recensé les besoins des utilisateurs lors d'une interview de Stéphane Baris, qui est membre de la division LAND de l'entreprise CGG.

Travaillant aussi bien sur le terrain avec les équipes d'acquisition des données que dans les bureaux de CGG à Massy, il connaît bien les besoins des géophysiciens et est aussi très impliqué dans le développement des applications interactives.

Il représente les utilisateurs de l'application Web de visualisation de données sismiques et a incarné à ce titre le rôle du Product Owner d'un point de vue de la méthode Scrum.

A travers l'utilisation des widgets graphiques de GeoToolKit, les utilisateurs devront pouvoir paramétriser l'affichage des données dans le cross plot d'abord en sélectionnant la colonne de données qu'ils souhaitent affecter à chacun des axes X, Y et Z du cross plot, ensuite en paramétrant d'autres options ou en filtrant les données.

Ils devront ainsi avoir la possibilité d'enregistrer des configurations d'affichage par type de fichier (type d'en-tête) et d'y faire appel à chaque re(connexion).

Ils devront disposer d'un espace de travail utilisateur dans lequel stocker leurs données importées et mémoriser leurs préférences d'affichage.

Cela m'a permis de rédiger un backlog produit comprenant une liste de user stories fonctionnelles.

Ces user stories, une fois classées, se regroupent naturellement en catégories de fonctionnalités qui aboutiront aux pages de l'application Web :

- ➔ Importer mes données dans mon espace utilisateur [ID],
- ➔ Gérer ma liste de données dans des dossiers [GD],
- ➔ Afficher mes données dans un cross plot ou leurs valeurs dans un tableau [AD],
- ➔ Paramétriser les options d'affichage dans le cross plot selon les différents types de données importées et pouvoir les modifier [AD].

BACKLOG PRODUIT **TOPAZ** : User stories fonctionnelles

Légende du backlog:

ID : Importer les Données

GD : Gestion des Données dans mon espace utilisateur

AD : Affichage de mes Données

En tant que géophysicien, je veux...

1	[ID] importer dans mon "espace utilisateur" les données sismiques à partir d'un fichier csv	ID
2	[ID] importer des fichiers csv polymorphe (en-têtes différents) dans mon "espace utilisateur"	ID
3	[ID] importer dans mon "espace utilisateur" les données sismiques à partir d'un fichier SDS	ID
4	[ID] importer dans mon "espace utilisateur" les données sismiques à partir d'un data store	ID
5	[ID] gérer mes données importées : ajouter des fichiers de données à ma liste de fichiers importés	ID
6	[ID] gérer mes données importées : supprimer des fichiers de données de ma liste de fichiers importés	ID
7	[ID] pouvoir visualiser ma liste de données importées, naviguer dans la liste	ID
8	[GD] classer les données importées dans des répertoires : créer des répertoires	GD
9	[GD] classer les données importées dans des répertoires : supprimer les répertoires	GD
10	[GD] classer les données importées dans des répertoires : modifier les répertoires	GD
11	[GD] classer les données importées dans des répertoires : afficher les répertoires	GD
12	[GD] gérer mes données importées : modifier des fichiers importés (renommer)	GD
13	[AD] visualiser graphiquement les données sismiques dans un cross plot sur mon navigateur	AD
14	[AD] visualiser les valeurs (Values) des données affichées dans le cross plot (des axes X, Y et Z)	AD
15	[AD] pouvoir switcher entre l'affichage du Cross Plot et celui des Values en mode : onglet ou accolé	AD
16	[AD] personnaliser les options d'affichage des données dans le cross plot selon le type d'en-tête du fichier importé	AD
17	[AD] Options d'en-tête : définir, selon le type d'en-tête de fichier, les colonnes affectées aux axes X, Y, Z	AD
18	[AD] Options d'en-tête : pouvoir définir des valeurs MIN et MAX pour chaque axe (X, Y, Z) du cross plot afin d'afficher un sous-ensemble de données	AD
19	[AD] Options d'en-tête : définir une colormap par type d'en-tête de fichier	AD
20	[AD] Options d'en-tête : pouvoir sélectionner une gamme de valeurs du cross plot sur laquelle appliquer une palette de couleur	AD
21	[AD] Options d'en-tête : pouvoir accéder à mes options d'affichage enregistrées pour chaque type de fichier et les modifier	AD
22	[AD] Options d'en-tête : pouvoir filtrer les valeurs du fichier importé dans le tableau avec mise à jour dans le cross plot	AD
23	[AD] Options d'en-tête : pouvoir trier les données sur la valeur "Z" et en tenir compte dans	AD

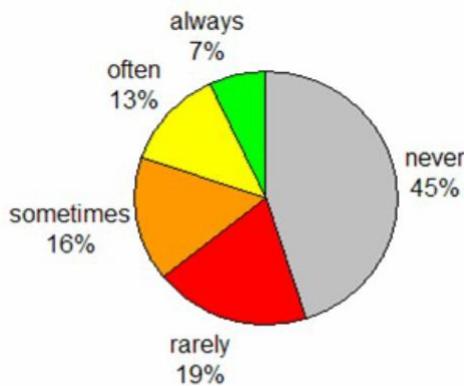
	I'affichage sur le cross Plot	
24	[AD] Options d'en-tête : personnaliser la légende de mon affichage (le label des axes) selon le type d'en-tête du fichier importé	AD
25	[AD] Options générales du cross plot : définir le pas de la grille d'affichage	AD
26	[AD] Options générales du cross plot : pouvoir activer ou non l'affichage de la grille dans le cross plot	AD
27	[AD] Options générales du cross plot : pouvoir changer la couleur de fond de la grille (blanc pour l'impression notamment)	AD
28	[AD] enregistrer plusieurs configurations d'affichage par type de fichier importé (multiconfiguration des Headers)	AD
29	[AD] pouvoir retrouver mes options d'affichage (préférences) à chaque (re)connexion	AD

2.2.2 Prioriser le backlog produit.

En gestion Scrum, le backlog produit :

- Doit être priorisé, afin de pouvoir extraire les user stories qui seront implémentées en priorité dans le sprint qui débute.

Cette étape est d'autant plus importante qu'une étude a démontré que 20% seulement des fonctionnalités demandées étaient « souvent » ou « toujours » utilisées.



Jim Johnson.

The Standish Group International Inc. 2002

L'objectif de la gestion de projet informatique en mode Scrum est donc de livrer en priorité des fonctionnalités opérationnelles utiles à l'utilisateur.

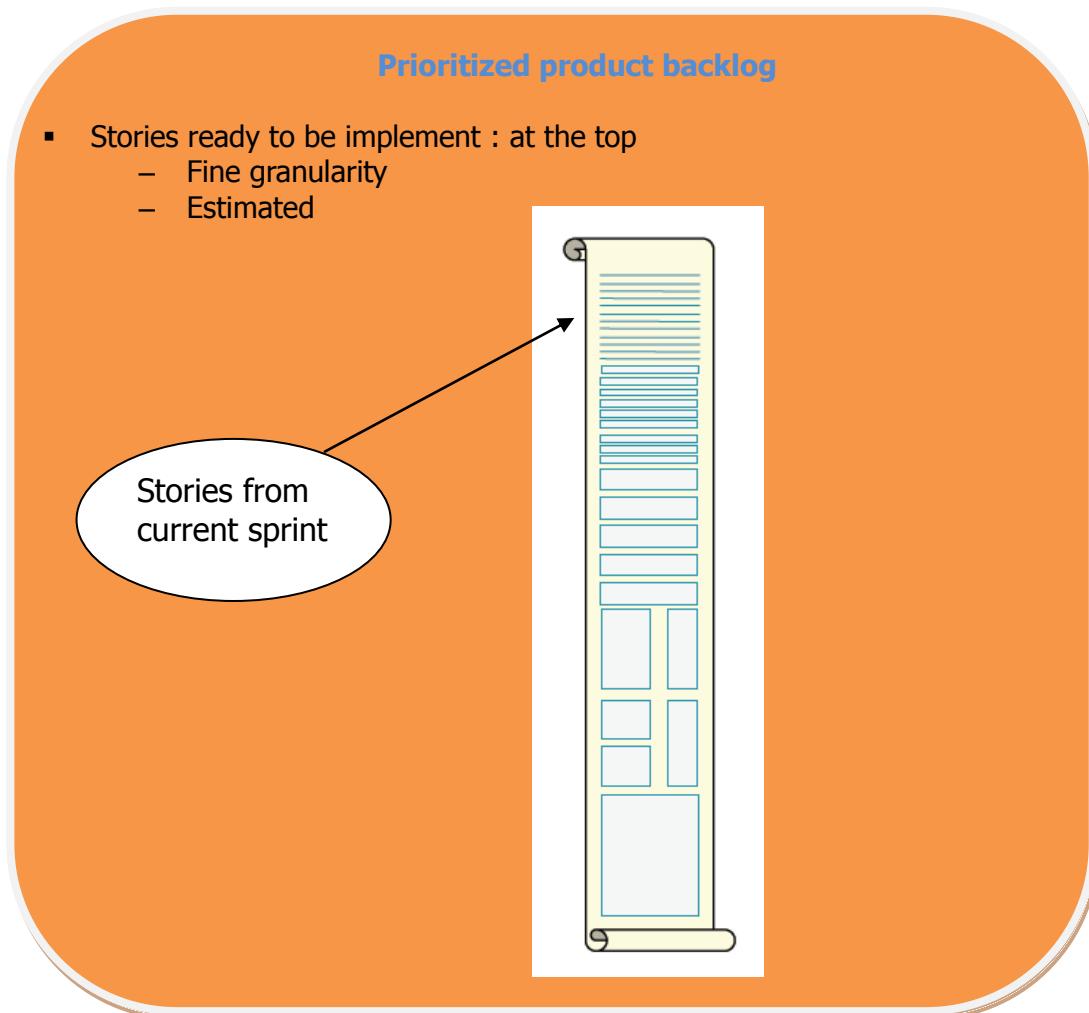
Même si le projet devait être interrompu faute de budget, le client est donc détenteur d'une application éventuellement partielle mais opérationnelle et surtout ayant les fonctionnalités essentielles.

- Doit avoir une granularité différente entre les stories prioritaires qui seront implémentées en premier, plus fines, et les suivantes.

En effet, les user stories sélectionnées pour le sprint courant seront détaillées en début de sprint, au cours d'une séance de planning poker, afin de décomposer la story en fonctionnalités plus fines, en tâches les plus simples possibles à implémenter.

Ainsi, les user stories du sprint en cours pourront facilement être estimées par l'équipe et être réalisables sur la durée de l'itération.

Volontairement, on ne perd pas de temps à détailler les stories moins prioritaires.



2.2.3 Définir le sprint backlog.

TOPAZ - Sprint n°1

Ce premier sprint est donc constitué d'un backlog adapté au contexte de mon stage.

Le but de l'application étant d'afficher dans un cross plot (sur un navigateur Web) les données qu'un géophysicien a importées dans son espace utilisateur, la condition préalable est que la base de données qui va stocker les données importées soit créée. C'est donc la première fonction à implémenter inclue dans le sprint 1.

Extrait d'un Sprint backlog

The screenshot shows a JIRA issue page for 'Training related epic' (COPSL-88). The top navigation bar includes 'Dashboards', 'Projects', 'Issues', 'Agile', and a 'Create' button. Below the header, there's a toolbar with 'Edit', 'Comment', 'Assign', 'More', 'To Do', 'In Progress', and 'Done' buttons. The main area is divided into sections: 'Details', 'Description', 'Issues in Epic', and 'Activity'. The 'Details' section shows the epic is an 'Epic' (Type), 'Trivial' (Priority), has no labels (Labels), and is named 'Web Viewer'. It is currently 'TO DO' (Status) and 'Unresolved' (Resolution). The 'Description' section contains a placeholder 'Click to add description'. The 'Issues in Epic' section lists 20 issues from COPSL-109 to COPSL-108, each with a status indicator (e.g., TO DO, IN PROGRESS, UNASSIGNED) and assignee (lellouche, catherine). The 'Activity' section at the bottom has tabs for 'All', 'Comments', 'Work Log', 'History', and 'Activity', with 'Comments' being the active tab. A note below states 'There are no comments yet on this issue.'

Backlog produit de TOPAZ priorisé

Cas d'utilisation	Priorité	Risque	Poids	Séquence
Importer mes données	Haute	Haut	6	1
Afficher mes données	Haute	Haut	6	2
Gérer mes données	Moyenne	Moyenne	4	3
Me connecter à mon espace utilisateur	Moyenne	Bas	3	4

Légende :

Haut = 3,
 Moyen = 2,
 Bas = 1.

Le risque est ici synonyme de complexité.

La décision de l'ordonnancement dépend de la combinaison Priorité/Risque.
 Un cas d'utilisation hautement prioritaire à haut risque est en haut de la pile.

Ce backlog du sprint 1 comprend donc :

- des stories portant sur la définition et la préparation du projet lui-même :
 - Appréhender la sismique et le métier de CGG,
 - définir des spécifications de l'application Web à développer,
 - appréhender la plateforme de développement et les outils existants (l'environnement de développement décrit en page 10 ainsi que l'ensemble des outils de production utilisés, dont la liste figure dans les annexes),
 - explorer la bibliothèque graphique sismique GeoToolKit que je dois utiliser pour l'IHM de l'application,
- et des user stories fonctionnelles concernant la conception de la base de données qui constituait le pré-requis avant de pouvoir développer les fonctionnalités.
 - la conception de la base de données.

CHAPITRE 3 - ANALYSE

Les étapes de l'analyse

La modélisation des cas d'utilisation et des scenarios (spécifications et architecture fonctionnelle) qui suit consiste à formaliser en UML ces cas d'utilisation.

Point de départ	Point d'arrivée	Contenu de l'étape
Besoin des utilisateurs : user stories du backlog produit Topaz	Diagramme de cas d'utilisation (DCU)	
DCU	Cas d'Utilisation (CU)	Diagramme d'un Cas d'utilisation Description textuelle d'un cas d'utilisation
Cas d'Utilisation	Diagramme de Séquence Système (DSS)	Diagramme sur 2 axes (temps –vertical, et espace – horizontal)
Cas d'Utilisation	Maquettage de l'interface utilisateur	Maquettes avec balsamiq
Maquettes	Navigation entre les pages de l'application	Diagramme de navigation (DNAV)
Présentation des maquettes et du diagramme de navigation au Product Owner	Validation de la partie statique de l'application	

3.1 - LES CAS D'UTILISATION (CU) ET LE DIAGRAMME DE CAS D'UTILISATION (DCU)

3.1.1 Représentation graphique

Définition et utilisation :

- Les **cas d'utilisation** décrivent exhaustivement les exigences fonctionnelles du système.

Chaque cas d'utilisation correspond à une fonction métier du système du point de vue d'un de ses acteurs.

La question à laquelle il faut répondre est "Pourquoi utilise-t-il le système ?".

Un Cas d'Utilisation permet donc d'illustrer une tâche fondamentale que l'utilisateur attend du système, il correspond à un de ses **besoins**. Il formalise les **interactions** des **acteurs** avec le **système**.

- Un **diagramme de cas d'utilisation** est un ensemble de cas d'utilisation. Il correspond à un processus métier.

Application :

Dans le projet TOPAZ, l'acteur principal est le géophysicien. C'est lui qui déclenche les cas d'utilisation de l'application.

3.1.2 Le DCU macroscopique

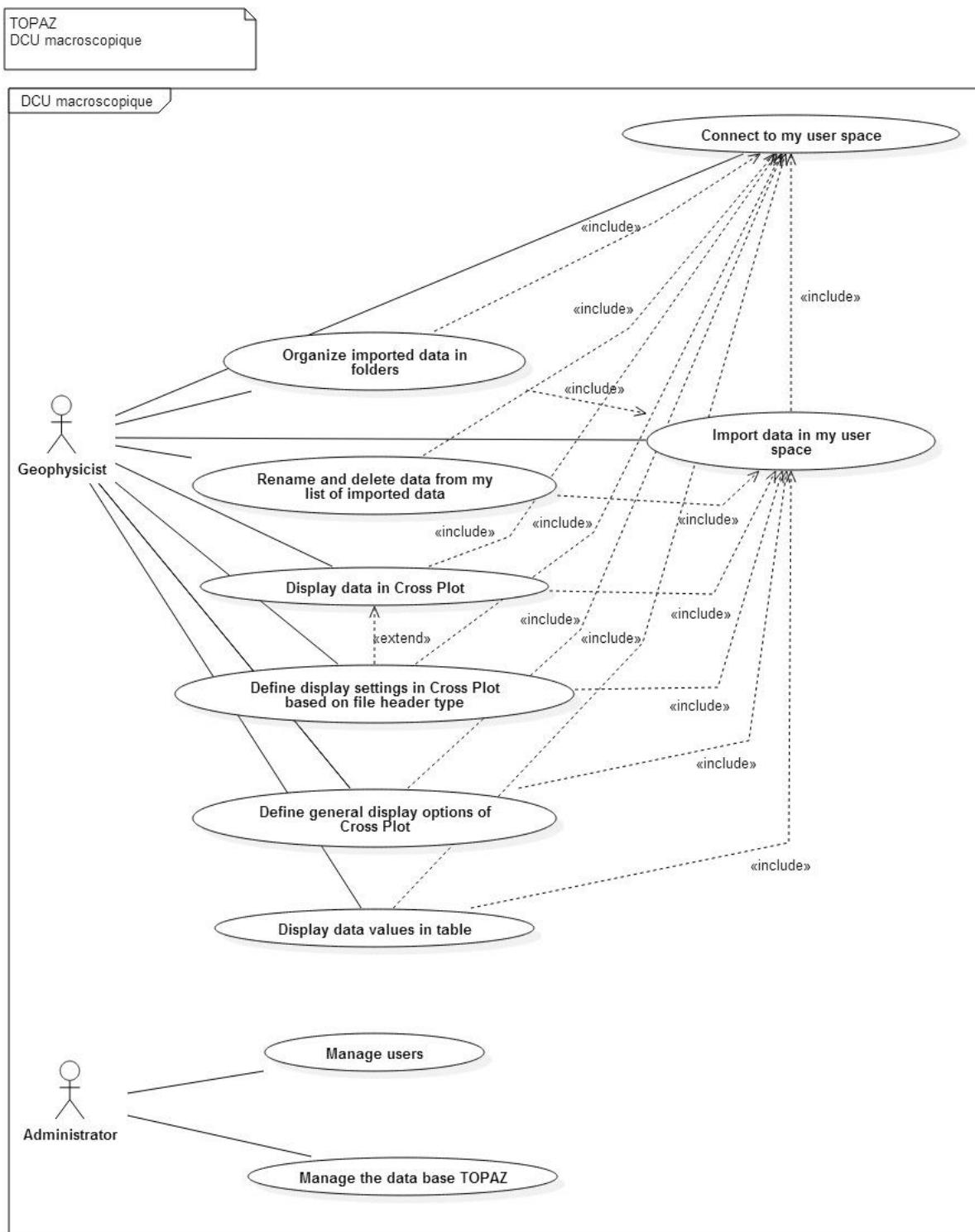


Figure DCUMacro : Le diagramme de cas d'utilisation macroscopique illustre les grandes fonctionnalités de cette application.

Notes :

La relation d'inclusion représente le fait qu'un Cas d'Utilisation **utilise** un autre Cas d'Utilisation via une méthode externe.

Cette relation permet de **factoriser** des processus et ainsi d'en éviter la duplication. Il y a un parallélisme avec le code.

Ici, tous les cas d'utilisation du géophysicien possédant cette relation <<include>> sont dépendants du CU « connect to my user space » et du CU « import data in my user space ».

La relation <<extend>> étend le cas "standard".

Ici, le Cas d'Utilisation « Define display settings in cross plot » ouvre des options au Cas d'Utilisation « Display data in cross plot ».

Il sera implémenté par une opération ou une fonction.

Dans le projet TOPAZ, les cas d'utilisation « s'inscrire » ou « s'authentifier » sont volontairement laissés de côté car la priorité est l'expérimentation de l'affichage dans le cross plot sur un navigateur.

Seulement si le projet est poursuivi, il sera opportun d'implémenter la connexion de l'utilisateur.

Conformément à la priorisation du backlog produit, le premier cas d'utilisation à implémenter est « import data in my user space » dans la mesure où tous les autres en dépendent.

Dans un second temps, il s'agira donc de « Display data in cross plot » et, en fonction du temps restant, d'implémenter le paramétrage des options d'affichage du user.

3.1.3 DCU Display data in cross plot and define display settings

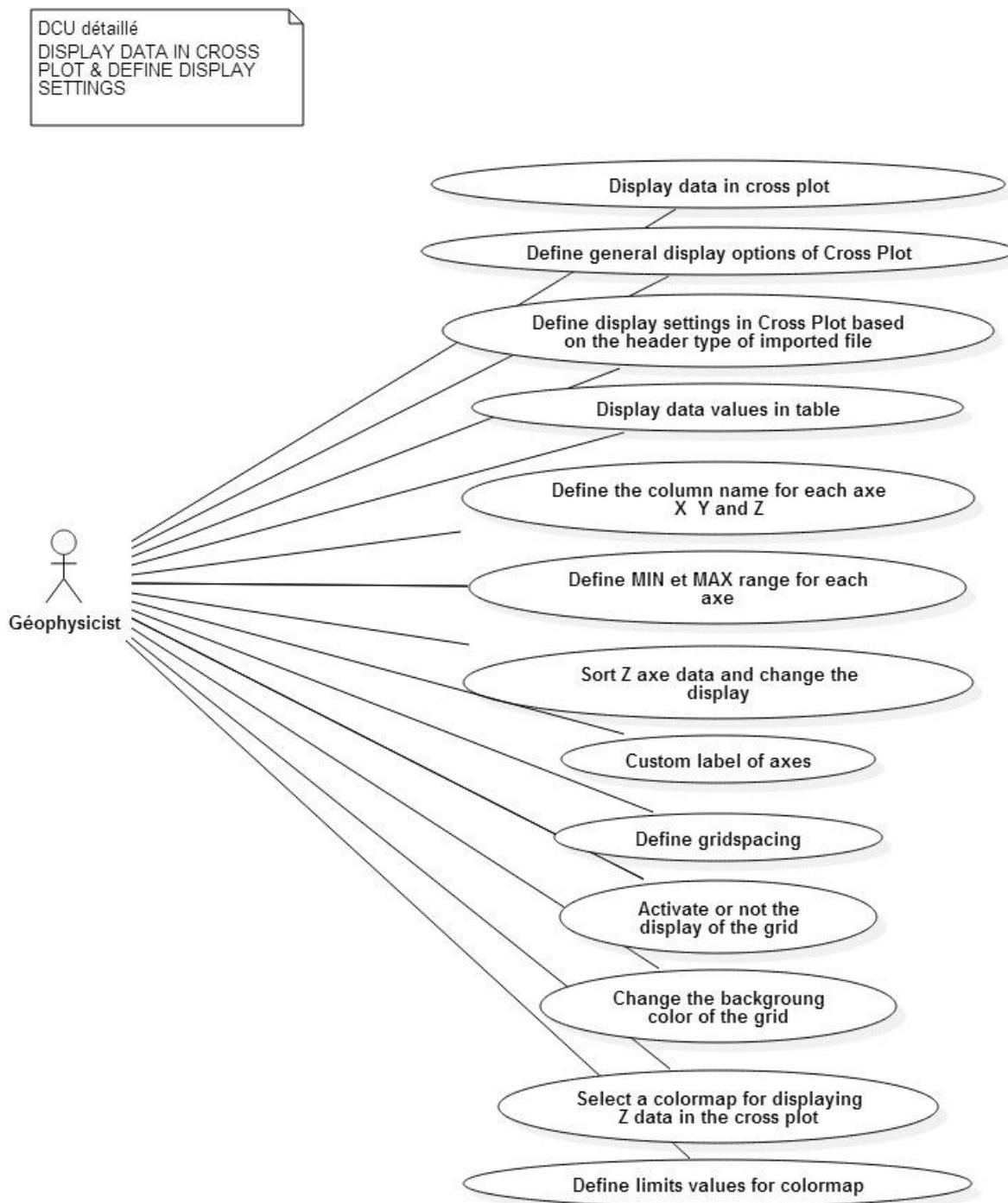


Figure DCUDisplaydata&DisplaySettings : Ce cas d'utilisation est le plus riche.

3.1.4 Fiche de description textuelle d'un cas d'utilisation

Ce cas d'utilisation **Import data** peut produire un ou plusieurs résultats. C'est la raison pour laquelle il peut être utile de le compléter par une **description textuelle**.

Il possède, dans le cas de TOPAZ, deux options, ainsi que le montre le diagramme d'activité en page 32.

- La première option est d'importer les données d'un fichier stocké localement sur l'ordinateur client ou sur une unité de stockage.
- La seconde option consiste à importer les données depuis les data store. Ces données sont accessibles via des serveurs et il faut passer par une API (Application Programming Interface) interne pour les importer.

Parce que cette 2^{nde} option est plus complexe à implémenter et pour ne pas retarder le développement du cas d'utilisation « Display datas in cross plot », mon tuteur de stage m'a demandé, dans un premier temps, de n'implémenter que l'import de données locales dans la base de données Topaz.

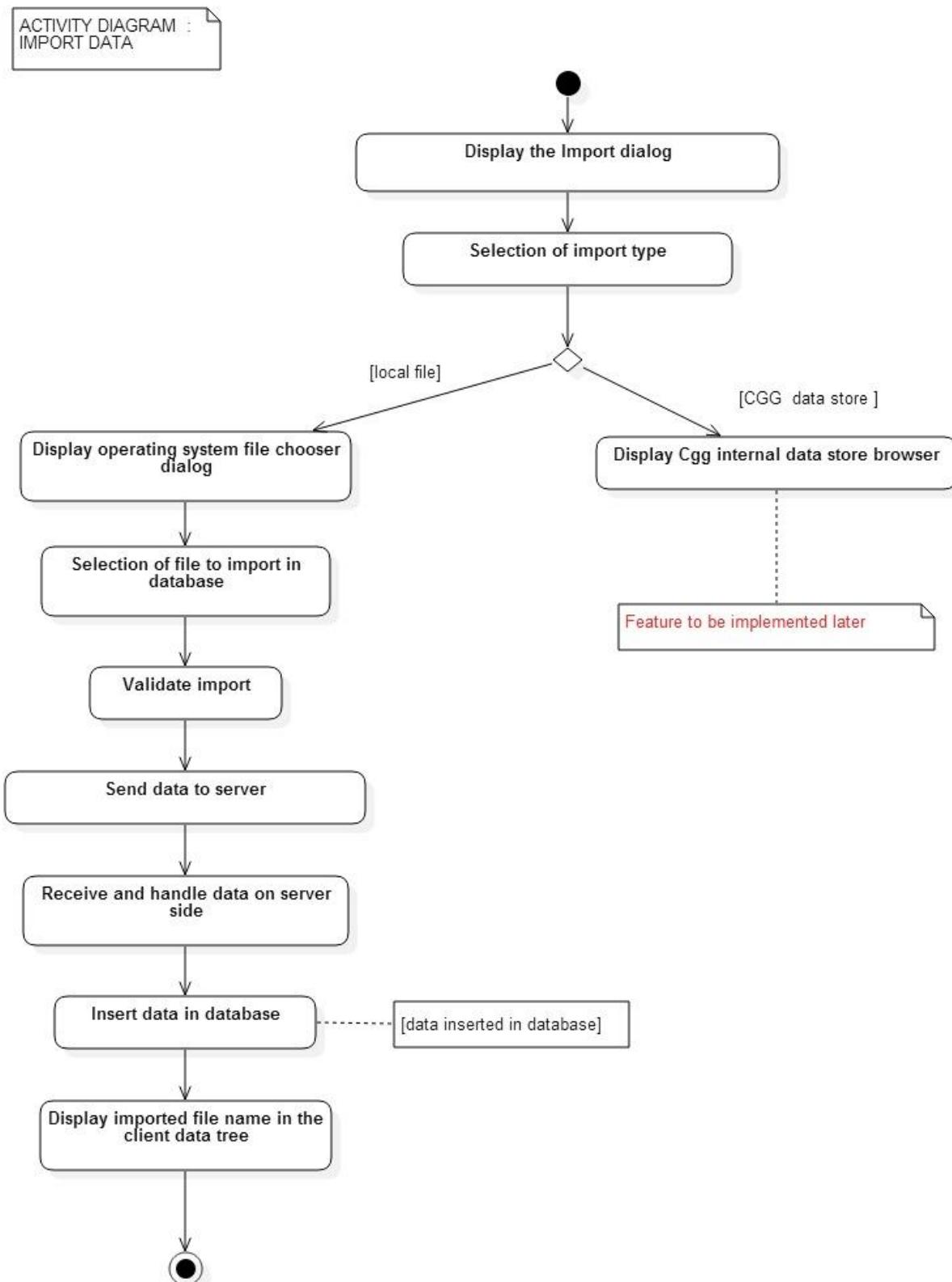
Ceci met en avant un des grands principes SCRUM. En effet, dans un projet géré en méthode Scrum, où nous cherchons à pouvoir faire une démonstration de la fonctionnalité intégrale, de l'interface à la base de donnée, la gestion de l'import de fichiers depuis les data store n'est pas une fonctionnalité prioritaire pour tester l'intérêt de l'application pour les utilisateurs.

C'est pourquoi la description textuelle du cas d'utilisation « **Import data** » qui suit ne décrit que la première option d'import (upload) de données locales.

Ce cas est instancié par un scenario (scenario nominal) ou plusieurs (scenario nominal +scenarii alternatifs + scenarii d'erreurs).

TOPAZ / CAS DUTILISATION : IMPORT DATA Description textuelle	
Etapes	Description
Identification du CU	Titre : Import data Résumé : Importer dans la BD des données issues de fichiers csv sélectionnés par l'utilisateur. Acteur : Géophysicien Date de création : 13/08/15 Date de dernière modification : 7/09/15 Version :1.0 Auteur : Catherine Lellouche
Pré-conditions	Boîte de dialogue disponible. Connexion à la BD disponible.
Scenario nominal	Sélection par l'utilisateur de l'option locale comme source du fichier. Ouverture de l'explorateur de fichiers sur le poste client. Sélection du fichier à importer. Validation de l'import par l'utilisateur. Lecture du fichier côté client (<i>en javascript</i>) Extraction des données et transfert (<i>au format JSON</i>) au serveur WEB. Traitement de la requête côté serveur. Insertion des datas dans la BD. Affichage d'un message validant l'insertion dans la base.
Scenarii alternatifs	L'utilisateur n'a pas sélectionné de fichier ou a sélectionné un fichier d'un autre format que csv avant de valider l'import. Ouverture d'une boîte de dialogue pour lui indiquer.
Scenarii d'erreurs du cas nominal	Echec de la lecture du fichier ou échec de l'extraction des données au format JSON ou échec de l'envoi de l'objet JSON au serveur provoquant un échec d'insertion des datas dans la BD Un message d'erreur est envoyé à l'utilisateur.
Post-conditions	Insertion réussie et ajout du nom du fichier dans le data tree (arborescence des données) sur la page « data view » de l'utilisateur.
Exigences non fonctionnelles	
Besoins d'IHM	Boîte de dialogue

3.1.5 Diagramme d'activité de « Import data »



3.2 - LES MAQUETTES

Les maquettes sont réalisées avec Balsamiq qui est l'outil utilisé chez CGG.

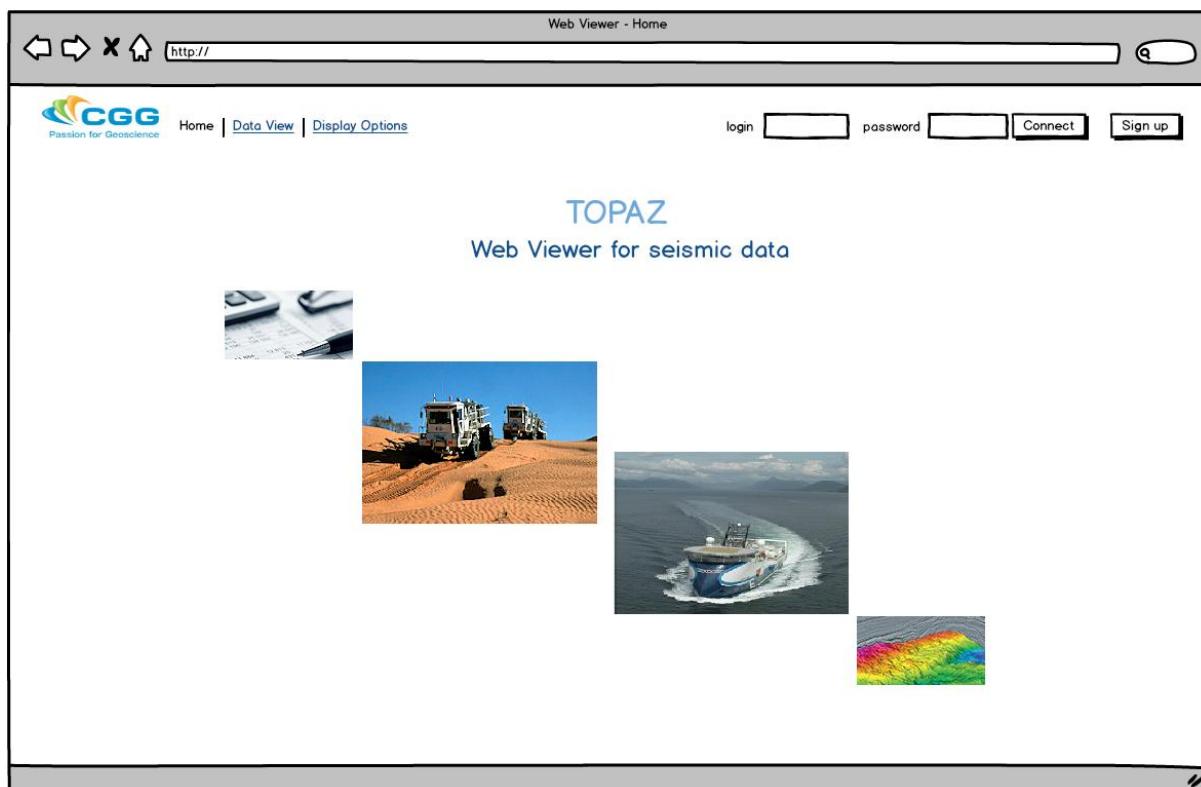
La liste des cas d'utilisation m'a permis de réaliser ces maquettes.
Elles ont été validées par le Product Owner et par mon tuteur de stage.

Nous aboutissons ainsi à 3 pages principales :

- L'accueil avec la connexion,
- La page principale comprenant 2 frames :
 - l'affichage dans l'arborescence des fichiers importés par l'utilisateur dans son espace,
 - l'affichage des données elles-mêmes (dans le cross plot et en valeur : [frames 1 et 2])
 - les paramètres d'affichage du cross plot (valeurs affectées aux axes X, Y et Z du cross plot et nom de la colormap appliquée pour afficher les valeurs Z)
 - ainsi que les boutons :
 - « Import » pour importer d'autres fichiers dans son espace,
 - « Create folder » pour créer des dossiers dans lesquels classer ses fichiers,
 - « Delete » pour supprimer des dossiers,
 - « Rename » pour renommer ses fichiers,
 - « Display » pour afficher un fichier sélectionné.
- la page « Display Header Settings » qui permet de visualiser les paramètres mémorisés pour l'affichage des différents type de fichiers importés, de les modifier et de les supprimer.

Les maquettes sont complétées par le diagramme de navigation qui sert à représenter le cheminement, la cinématique de l'application (ou couche Control), entre les différents écrans (IHM) de l'application (ou couche Dialogue - Boundary).

3.2.1 La page d'accueil et de connexion

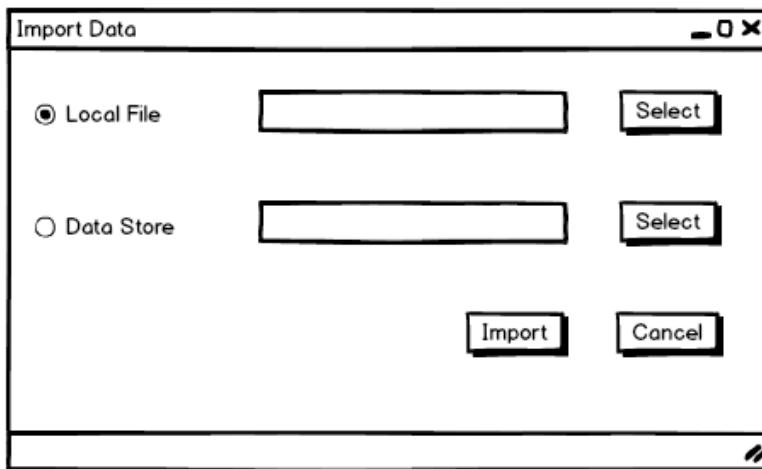


3.2.2 La boîte de dialogue « Sign up »

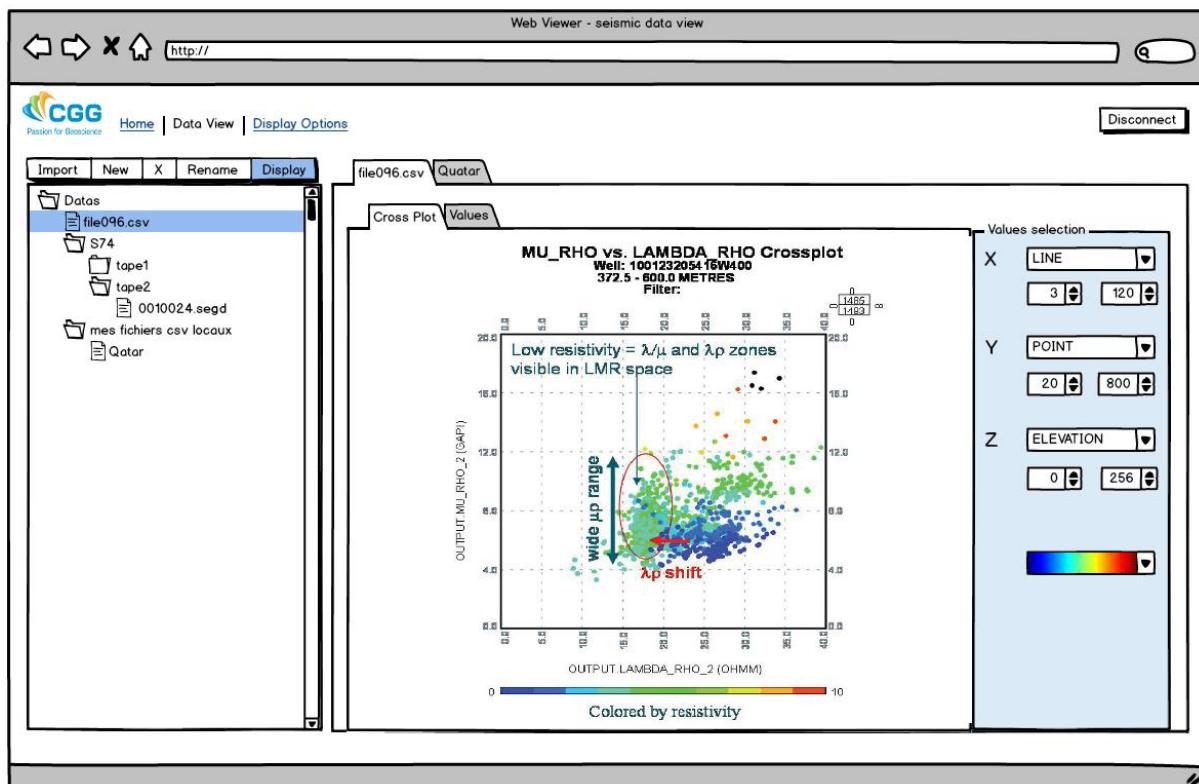
Create my user account

Firstname	<input type="text"/>
Lastname	<input type="text"/>
Enter an email	<input type="text"/>
Enter a password	<input type="text"/>
Confirm your password	<input type="text"/>
<input type="button" value="cancel"/> <input type="button" value="Create my account"/>	

3.2.3 La boîte de dialogue « Import Data »



3.2.4 La page « Display datas (in cross plot) »



3.2.5 La page « Display datas (values) »

Web Viewer - seismic data values

<http://>

CGG Home | Data View | [Display Options](#) Disconnect

New X Rename Import Display file096.csv Quatar

Datas S74 file096.csv tape1 tape2 0010024.segd mes fichiers csv locaux Quatar

Cross Plot Values

Type	Line	Point	Index	Code	X	Y	Z	Vertical Time	Point depth	Water Depth
S	1924.00	2915	1	V1	156867.3	374132.7	3.1	0	0	0
S	1924.00	2913	1	V1	156866.9	374117.5	2.9	0	0	0
S	1924.00	2911	1	V1	156866.6	374102	3.2	0	0	0
S	1924.00	2909	1	V1	156866.8	374087.3	3.1	0	0	0
S	1936.00	2915	1	V1	156956.8	374131.8	4.1	0	0	0
S	1936.00	2913	1	V1	156957.2	374117.7	4.3	0	0	0
S	1936.00	2911	1	V1	156956.9	374102.2	4.4	0	0	0
S	1936.00	2909	1	V1	156956.9	374086.9	4.3	0	0	0
S	1924.00	2914	1	V1	156867.1	374124.9	3	0	0	0
S	1924.00	2912	1	V1	156867.2	374109.7	3	0	0	0
S	1924.00	2910	1	V1	156866.5	374094.1	3.2	0	0	0
S	1924.00	2908	1	V1	156866.8	374080.3	3	0	0	0
S	1924.00	2905	1	V1	156866.9	374057.2	2.9	0	0	0

Values selection

X LINE 3 12

Y POINT 20 80

Z Z 0 25

Colorbar

3.2.6 La page « Display Options »

Web Viewer - viewer options

<http://>

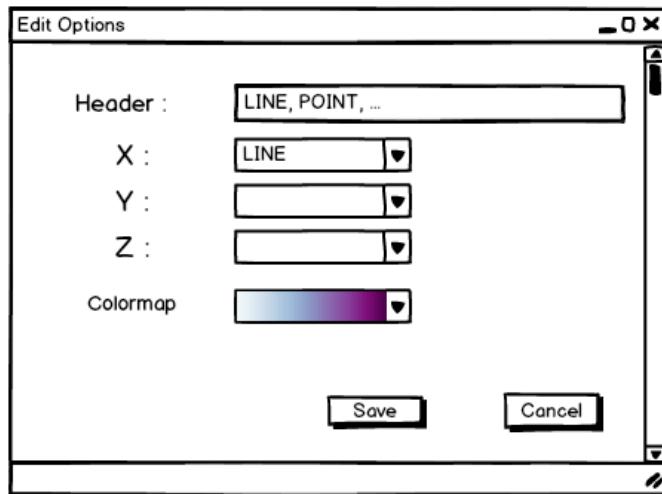
CGG Home | Data View | [Display Options](#) Disconnect

HEADER LINE, POINT, INDEX, NB_ACQ, SWEEP_LENGTH Type, Line, Point, Index, SHOT_NUMBER Line, Num

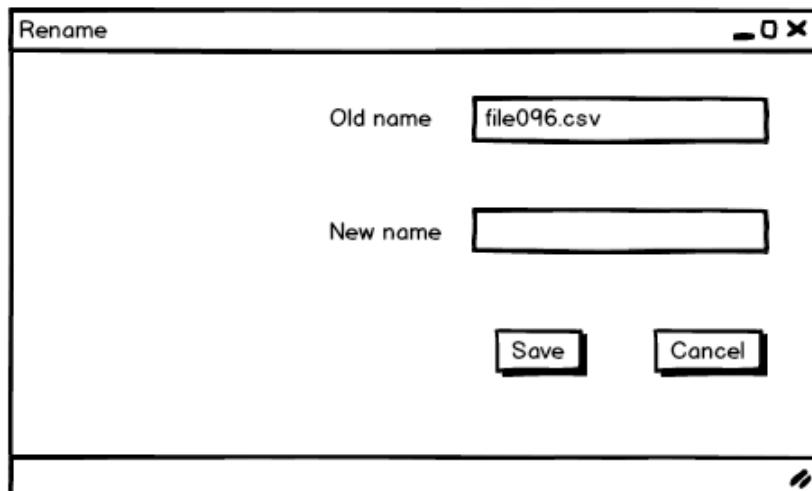
X	Y	Z	Colormap
LINE	POINT	INDEX	Colorbar
Line	Point	Index	Index
Line	Num		

Edit Delete

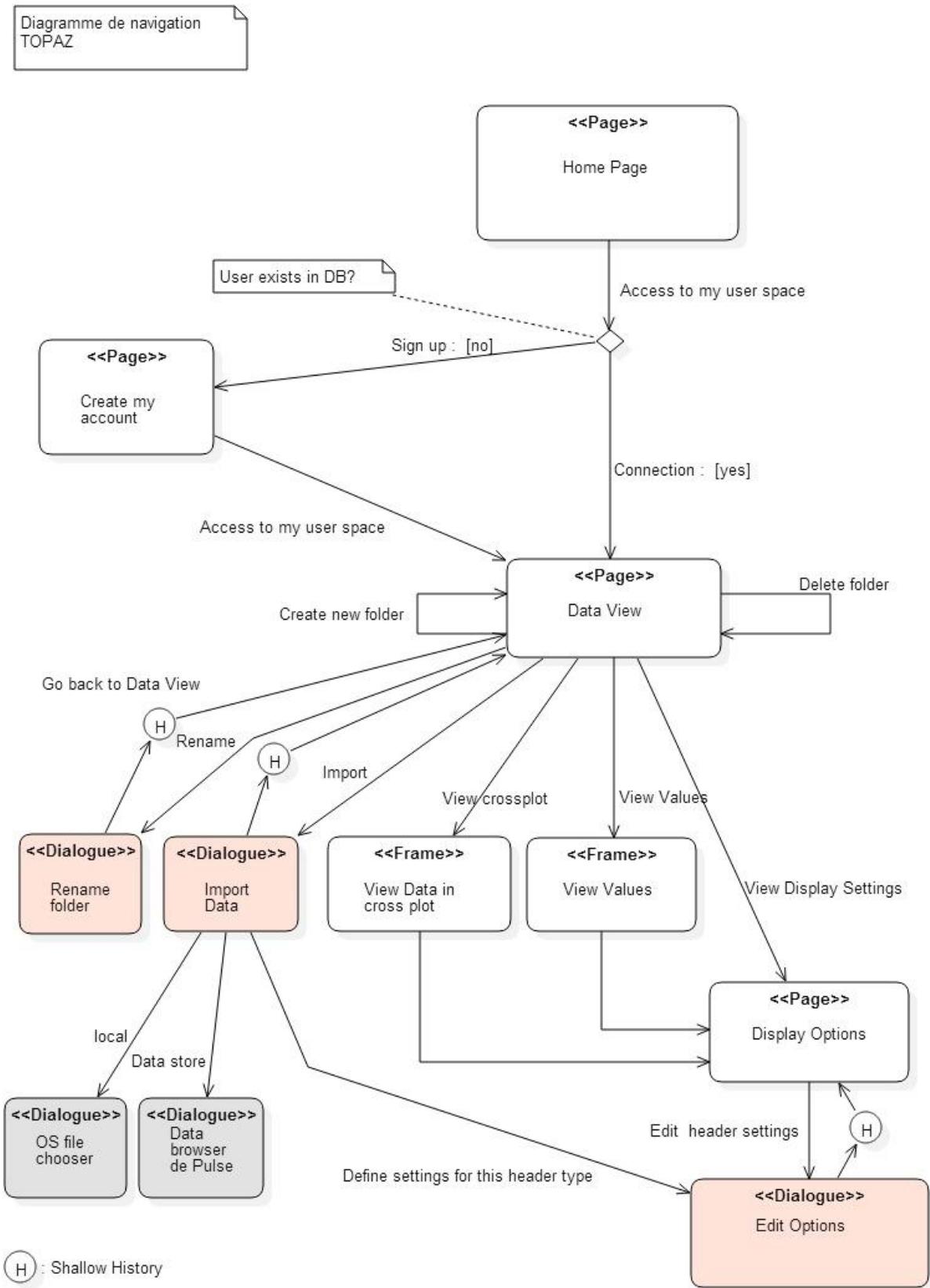
3.2.7 La page « Edit Options »



3.2.8 La page « Rename»



3.3 - LE DIAGRAMME DE NAVIGATION



CHAPITRE 4

CONCEPTION DE LA BASE DE DONNEES

4.1 - LA DEMARCHE UTILISEE

L'intérêt de la démarche utilisée, décrite ci-dessous, est d'obtenir un modèle de données normalisé au niveau de la 3^{ème} forme normale de Boyce Codd.

La normalisation a pour objectif d'obtenir des données cohérentes par rapport au réel, d'**éliminer les redondances** dans la base de données ainsi que les **anomalies de mise à jour**.

- Pour aboutir au « Modèle Conceptuel des Données » :
 - Etablir le dictionnaire des données à partir des user stories m'est apparu comme la manière la plus adaptée pour déterminer de manière exhaustive quelles étaient les données devant être stockées dans la base de données TOPAZ.
 - Faire la matrice des dépendances fonctionnelles :

Rappel : il y a dépendance fonctionnelle entre deux attributs lorsque la **connaissance d'une valeur** d'un attribut permet de déterminer **une et une seule** valeur d'un autre attribut.

 - Répertorier toutes les données recensées dans le dictionnaire.
 - Analyser les dépendances fonctionnelles entre les données et les noter.

Cette méthode permet d'être exhaustif.
 - Faire le graphe des dépendances fonctionnelles (GDF),
Le graphe permet de visualiser plus facilement :
 - les données structurées en ensemble (qui deviendront des « entités » puis des « tables » dans les étapes ultérieures de la modélisation),
 - et les dépendances entre ces données.
 - Réaliser le diagramme de classes entités où sont représentées les classes qui sont les entités, les associations entre les classes-entités et les cardinalités de ces associations.
- Pour l'élaboration du Modèle Physique des Données (MPD) :

Je l'ai réalisé à partir du diagramme de classes avec MySQL WorkBench.

 - La génération du script SQL de création de la base avec MySQL WorkBench,
- Pour la création de la base de données TOPAZ :

J'ai créée la base de données TOPAZ (son schéma) dans le SGBDR MySQL en exécutant le script SQL de création de la base avec MySQL Query Browser.

4.2 LE MODELE CONCEPTUEL DES DONNEES

Cela consiste à faire une représentation graphique et structurée des informations qui seront mémorisées dans le système d'information de TOPAZ.

4.2.1 Le dictionnaire des données.

Il s'agit d'une version simplifiée qui vise à classifier les données, identifiées comme devant être enregistrées dans la base de données, à partir des user stories.

LEGENDE :

Ensemble de données : fournit les classes

Données atomiques : fournit les titres de colonnes

Valeurs : des valeurs contenues dans les colonnes

UTILISATEUR GEOPHYSICIEN :

[AD]me connecter à mon "espace utilisateur"

[AD] importer dans mon "espace utilisateur" les données sismiques à partir d'un fichier csv

[AD] importer dans mon "espace utilisateur" les données sismiques à partir d'un fichier SDS

[AD] importer dans mon "espace utilisateur" les données sismiques à partir d'un data store

[AD] visualiser les données sismiques dans des cross plots sur mon navigateur

[AD] pouvoir naviguer dans la liste des données/fichiers que j'ai importées.

[GD] classer les données importées dans des répertoires : créer des répertoires.

[GD] classer les données importées dans des répertoires : supprimer les répertoires.

[GD] classer les données importées dans des répertoires : modifier les répertoires.

[GD] classer les données importées dans des répertoires : afficher les répertoires.

[GD] gérer mes données importées : ajouter des fichiers de données à ma liste de fichiers importés

[GD] gérer mes données importées : supprimer des fichiers de données de ma liste de fichiers importés

[GD] gérer mes données importées : modifier des fichiers importés (renommer)

[OA] personnaliser les options(propriétés) d'affichage (des données) par type de fichier importé

[OA] pouvoir retrouver mes options d'affichage (préférences) à chaque (re)connexion

[OA] pouvoir définir, lors de l'affichage dans le cross plot, les colonnes du fichier importé affectées aux axes X, Y et Z

[OA] que le système mémorise, par type de fichier, les colonnes affectées à X, Y, Z et mes autres options d'affichage(range, colormap, etc...)

[OA] pouvoir modifier les affectations de colonne aux axes X, Y et Z et les options d'affichage pour un type de fichier

[OA] pouvoir afficher les valeurs des données des axes X, Y et Z

[OA] pouvoir définir des bornes minimum et maximum pour chaque axe X, Y, Z afin d'afficher un sous-ensemble des données

[OA] pouvoir trier les données sur la valeur "Z" et en tenir compte dans l'affichage sur le X-Plot

[OA] personnaliser la légende de mon affichage (le label et le nom des axes)

[OA] définir le pas de la grille d'affichage

[OA] pouvoir activer ou non l'affichage de la grille dans le cross plot

[OA] pouvoir changer la couleur de fond de la grille (blanc pour l'impression notamment)

[OA] pouvoir sélectionner une palette de couleur (colormap) pour l'affichage des cross plots

[OA] pouvoir sélectionner une gamme de valeurs (range) pour chaque axe du cross plot (range) sur laquelle appliquer une palette de couleur

UTILISATEUR ADMIN

gérer les comptes utilisateurs

DICTIONNAIRE DES DONNEES

re)connexion

utilisateurs/comptes utilisateurs

espace utilisateur

données sismiques

la liste des données/fichiers que j'ai importées par utilisateur

type de fichier importé

fichier csv

fichier SDS

data store

les colonnes du fichier importé

des répertoires

des axes X, Y et Z

des bornes minimum et maximum pour chaque axe X, Y, Z

propriétés d'affichage des données par type de fichier importé

préférences utilisateur

définir Range/ gamme de valeurs

colormap (palette de couleur)

légende : le label des axes

définir le pas de la grille d'affichage

changer la couleur de fond de la grille

4.2.2 Le modèle conceptuel des données.

Ce dictionnaire m'a permis de commencer à modéliser les données et d'aboutir à la structure suivante :

(utilisateurs) Users	(Répertoire) Folder	(Fichier) File	(Type de fichier) Header Type	(Options d'affichage d'un header) HeaderDisplayOptions	Colonnes du fichier	Données du fichier
Id Utilisateur	Id répertoire	Id Fichier	Id Header	Id Options		
Nom	NomFolder	NomFichier	Header definition	Axe X value		
Prenom		CheminFichier(url)		Axe Y value		
Login				Axe Z value		
Mot de Passe				X label		
				Y label		
				Z label		
				Colormap		
				MIN value X		
				MAX value X		
				MIN value Y		
				MAX value Y		
				MIN value Z		
				MAX value Z		
				Pas de la grille		
				Couleur de fond de la grille		
				IdHeader		

4.2.3 Les règles de gestion.

Les règles de gestion sont les règles « métier » qui permettent de contraindre, contrôler et influencer un aspect du métier.
Il est nécessaire de concevoir la base de données en appliquant ces règles.

J'ai déterminé les règles de gestion suivantes grâce à l'interview du Product Owner et en les validant avec mon tuteur de stage

- Un utilisateur géophysicien possède un répertoire principal (son espace utilisateur) dans lequel seront stockés ses fichiers importés, éventuellement classés dans des répertoires.
- Un répertoire peut contenir 0 à n autres dossiers créés par l'utilisateur pour classer ses données importées.
- Un fichier de données sismiques peut être contenu dans plusieurs répertoires (le répertoire de plusieurs utilisateurs).
- L'utilisateur pourra supprimer des données de son espace.
- Au moment de l'import d'un fichier dans son espace, l'utilisateur pourra renommer le fichier.
- Un fichier importé est caractérisé par 1 et 1 seul en-tête (type de header). Un type de header correspond à 0 ou n fichiers.
- Un fichier contient de 0 à n datas. Des datas appartiennent à un seul fichier.
- Un type de header sera associé à des paramètres d'affichage des données dans le cross plot (options d'affichage) pour un utilisateur.
- L'utilisateur pourra accéder aux préférences d'affichage qu'il aura défini pour ses headers, et pourra les modifier et en supprimer.

4.2.4 La matrice des dépendances fonctionnelles.

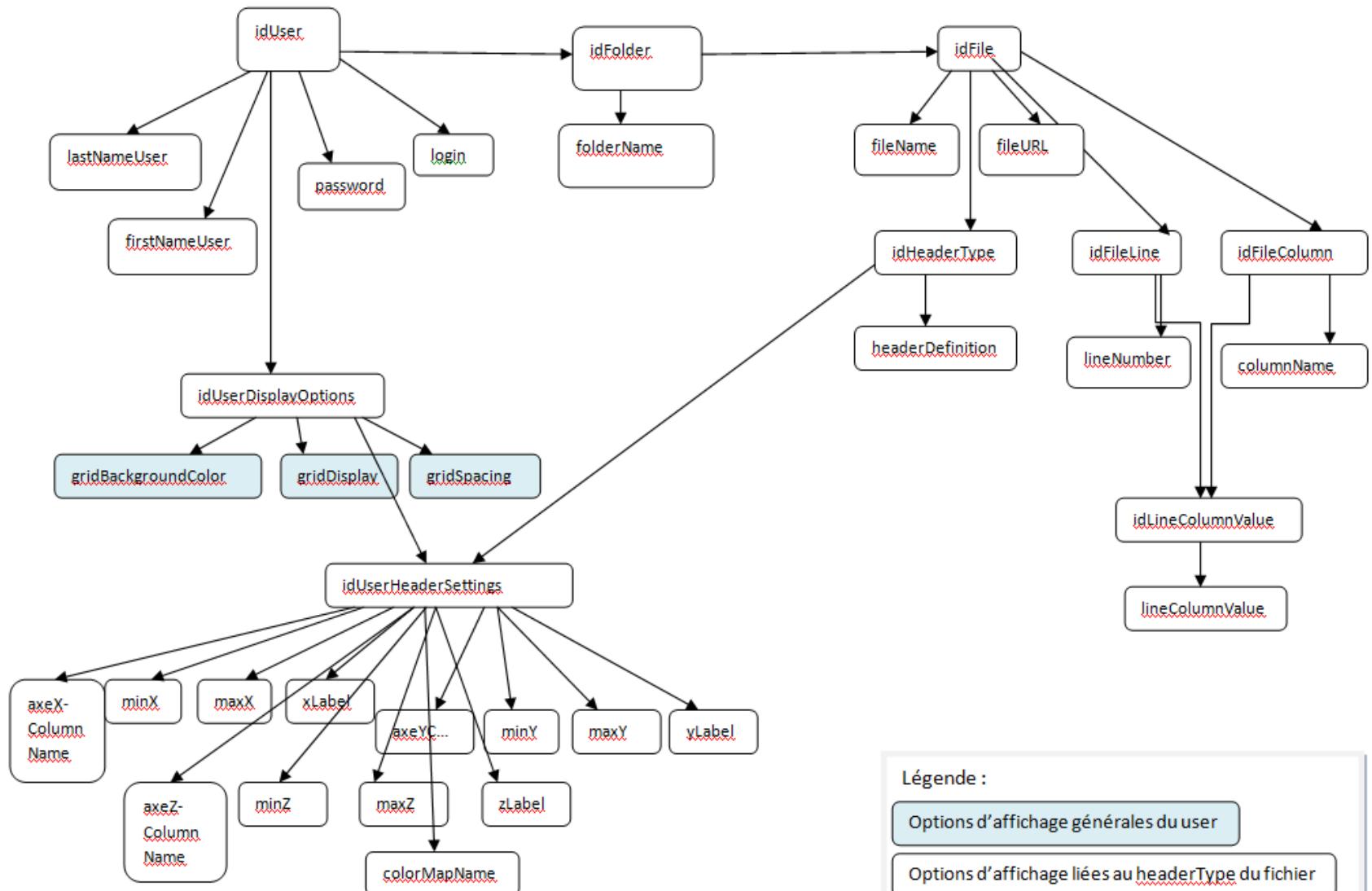
PROJET TOPAZ
MATRICE DES DEPENDANCES FONCTIONNELLES

Attributs	N°	1	2	3	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	29	30	31	32	33	34	35	36	37	38		
idUser	1	X	X	X	X	X	X																																		
lastnameUser	2		X																																						
firstnameUser	3			X																																					
login	4				X																																				
mdp	5					X																																			
idFolder	6						X	X																																	
folderName	7							X																																	
idFile	8								X	X	X																														
fileName	9									X	X																														
urlFile	10										X																														
idFileLine	11										X	X																													
lineNumber	12											X																													
idFileColumn	13											X	X	X	X																										
columnName	14												X																												
idColumnType	15													X	X																										
columnType	16														X																										
idLineColumnName	17															X	X																								
lineColumnName	18																X																								
idHeader	19																	X	X																						
headerDefinition	20																		X																						
idUserDisplayOption	21																			X	X	X	X	X	X																
gridBackgroundColor	22																				X																				
gridDisplay	23																				X																				
gridSpacing	24																					X																			
idUserHeaderSettings	25																						X																		
xValue	26																							X																	
minX	27																								X																
maxX	28																									X															
xLabel	29																										X														
yValue	30																											X													
minY	31																												X												
maxY	32																													X											
yLabel	33																														X										
zValue	34																															X									
minZ	35																																	X							
maxZ	36																																		X						
zLabel	37																																			X					
Colormap	38																																						X		

4.2.5 Le Graphe des Dépendances Fonctionnelles.

**Règles de Transformation de la matrice des dépendances fonctionnelles en
graphe des dépendances Fonctionnelles**

1. l'attribut, source du plus grand nombre de DF, est situé en haut du graphe,
2. une dépendance fonctionnelle est représentée par une flèche,
3. les dépendances qui s'obtiennent par transitivité sont éliminées (couverture minimale),
4. pour les attributs orphelins, il faut rechercher les sources multiples de dépendances fonctionnelles.



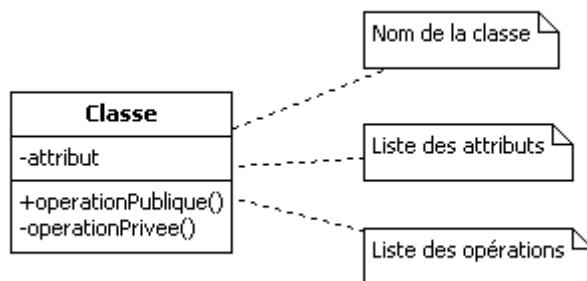
4.3 - LE DIAGRAMME DE CLASSES (DCL)

- Généralités :

C'est un diagramme statique normalisé de l'UML, qui représente :

- des classes entités,
- les relations entre les classes,
- et les multiplicités des relations.

Définition de la classe : une classe est une représentation abstraite d'un ensemble d'objets réels ou virtuels qui possèdent une structure, un comportement et des relations similaires.



Le modèle E-C-B d'Ivar Jacobson.

Ivar Jacobson propose 3 stéréotypes pour Entity, Control et Boundary.

Boundary : classe de frontière. Les objets de ces classes sont visibles et ils fournissent un moyen de communication entre le système et l'extérieur (Interface Homme Machine).

Entity : classe entité. Les objets de ces classes représentent les données du domaine de l'étude. Elles sont généralement persistantes.

Control : classe contrôle. Les objets de ces classes sont internes au système. Elles contrôlent le comportement des cas d'utilisation. Les classes représentent une activité de contrôle, une règle de gestion, un « chef d'orchestre », ...

- Application :

L'objectif ici est de représenter les **classes entités** pour modéliser les données du domaine de l'application TOPAZ.
C'est un schéma entité-association.

Méthodologie :

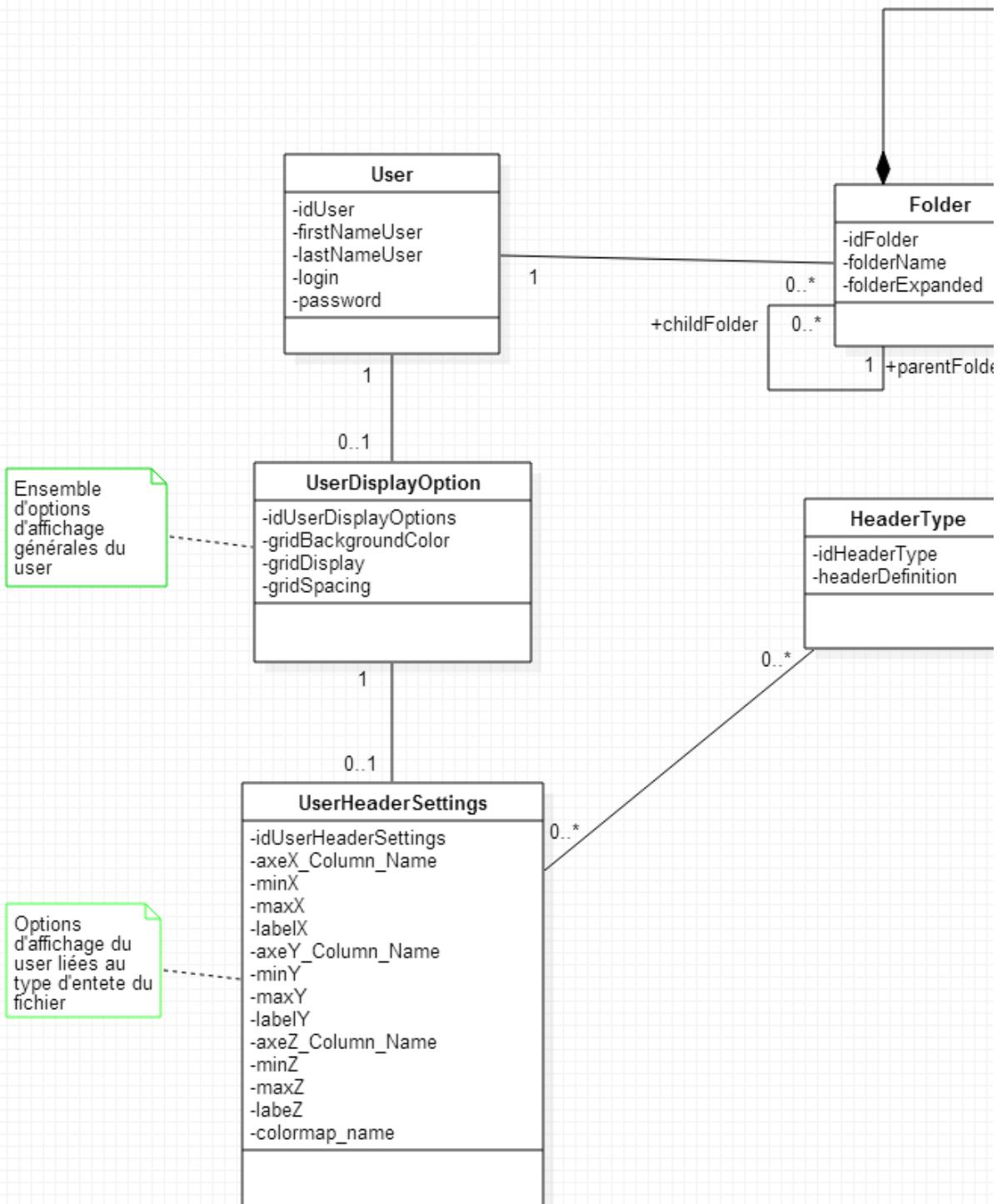
J'ai défini les classes et élaboré ce diagramme de classes entités à partir du Graphe des Dépendances Fonctionnelles et en appliquant les règles de transformation suivantes :

Règles de Transformation du GDF en DCL

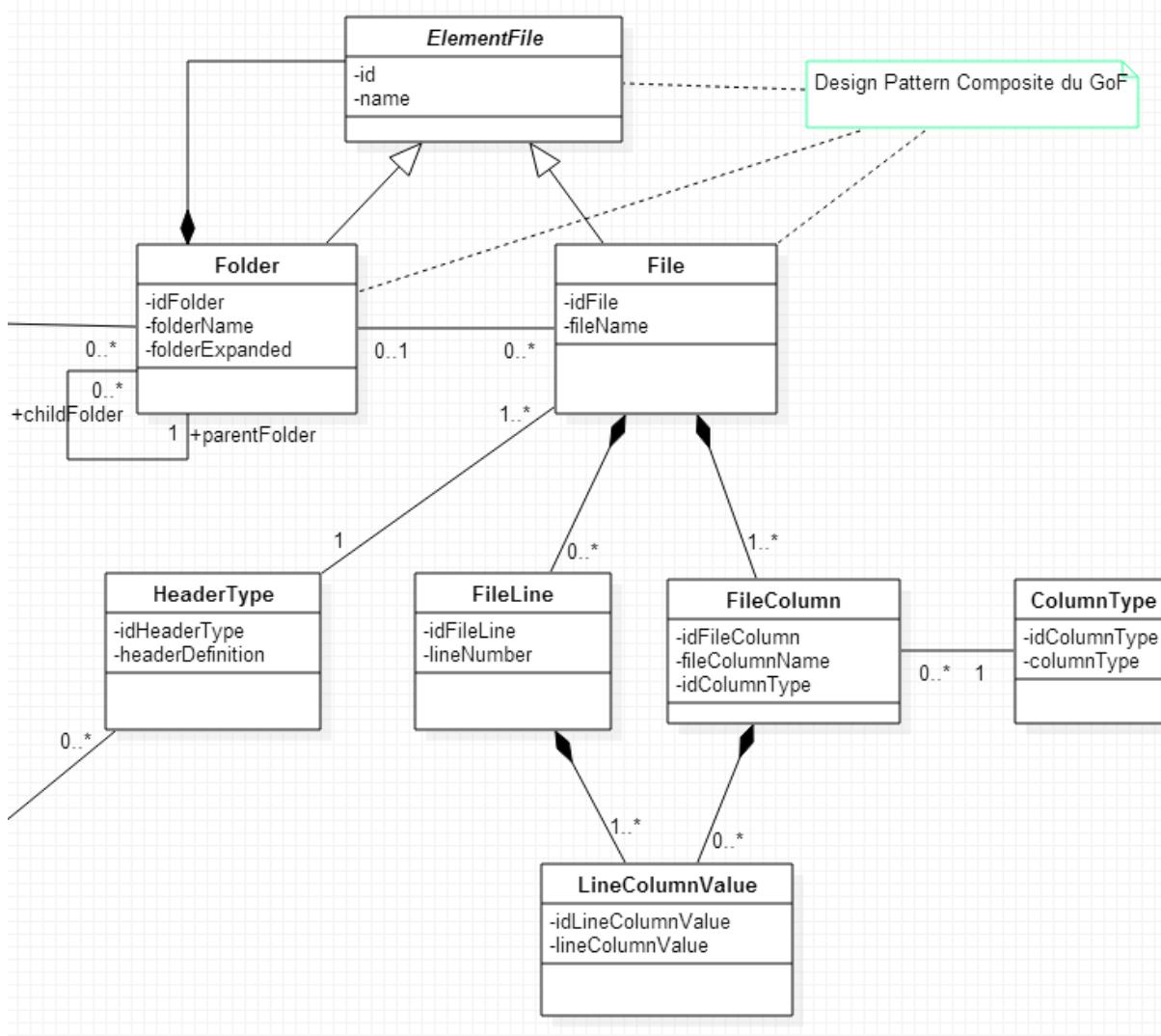
GDF	DCL
Arbre	Classe
Sommet d'un arbre	Attribut identifiant d'une classe
Feuille terminale d'un arbre	Attribut non-identifiant d'une classe
Concaténation (***)	Association de type non Père-Fils (multiplicités de type * - *)
Feuille d'une concaténation →	Classe association → Attribut d'une classe-association
DF inter-sommets	Association inter-classes avec au moins une multiplicité maxi de 1 (**)
Un attribut ayant plusieurs sommets	Une classe et 2 associations (*)

4.3.1 DCL entités : l'utilisateur et ses options d'affichage du cross plot.

TOPAZ
Diagramme de classes ENTITIES



4.3.2 Le DCL entités : le fichier (file).



La problématique était la suivante : les fichiers csv importés sont polymorphes avec un nombre de colonnes variable.

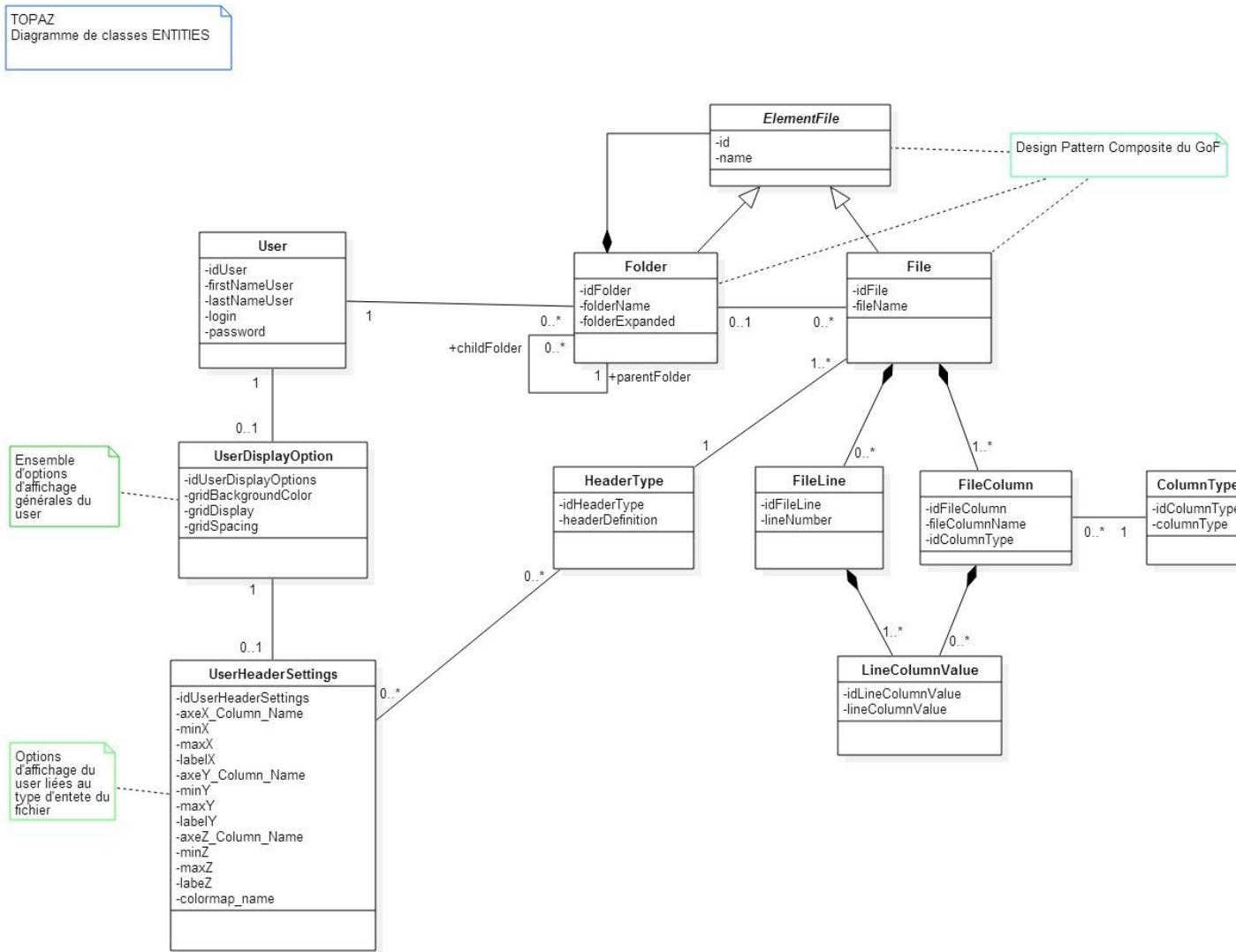
J'ai utilisé ici le Design pattern Composite du GOF car il se prête à la hiérarchisation des dossiers (Folder) et des fichiers (File).

En effet, ce **design pattern** est un arrangement caractéristique de classes reconnu comme bonne pratique pour les structures d'objet arborescentes ayant des relations de composition. Ici, il y a une relation de composition entre les classes ElementFile, Folder, et File entre lesquelles il y a une relation de contenance. La composition est une **agrégation forte**.

Il y a aussi d'autres agrégations fortes entre les classes FileLine et File, FileColumn et File et entre LineColumnValue et FileLine & FileColumn.

La multiplicité du côté du composite est de 1. Le composant appartient au composite. La destruction du composite entraîne la destruction des composants.

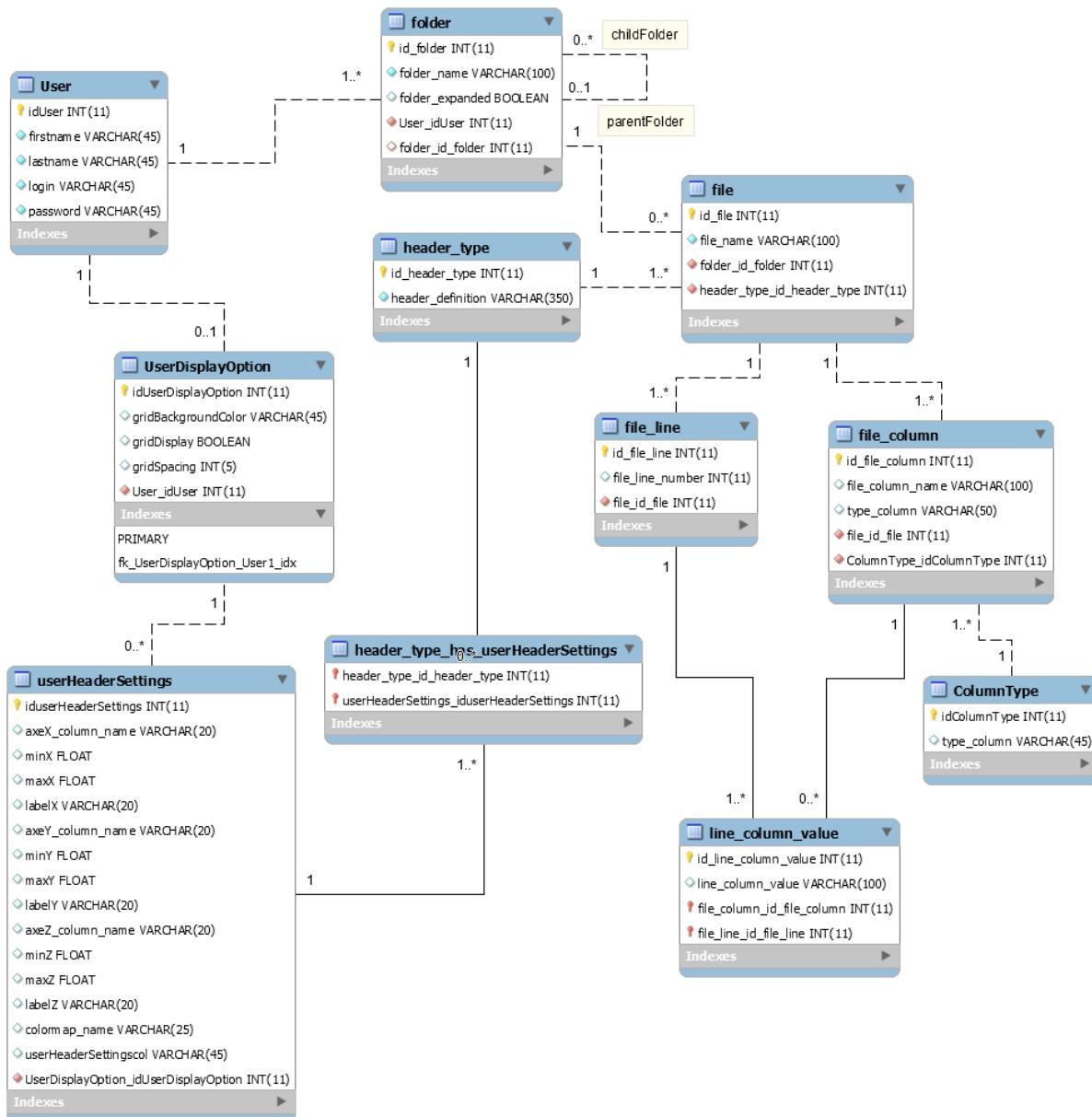
4.3.3 Diagramme de classes Entités complet :



4.4 - LE MODELE PHYSIQUE DES DONNEES

Il représente le schéma de la base de données.

J'ai réalisé ce Modèle physique de données (MPD) avec MySQL WorkBench.



4.5 - LES PROCÉDURES STOCKÉES

Présentation

La plupart des SGBDR disposent d'un [langage procédural](#) qui permet d'étendre les capacités de SQL. Ce code contiendra des instructions procédurales (gestion de variables, conditionnelles, boucles...) et du code SQL.

MySQL fournit depuis la version 5 un langage procédural qui permet de créer des procédures stockées.

Objectifs des procédures stockées :

1. Séparer le code client du code serveur.

Dans le cas de Topaz, le client est l'application JSP, le serveur est le serveur de BD, c'est-à-dire MySQL.

Les procédures stockées servent à stocker sur le serveur de BD des instructions SQL pré-compilées.

La dynamique de la base de données (le CRUD) est ainsi encapsulée dans la BD et rendue disponible pour n'importe quelle application.

2. Faciliter la maintenance.

Les ordres SQL sont disponibles depuis n'importe quel programme client (lourd/léger). Il n'y a pas de duplication du code SQL dans l'application.

3. Optimiser la charge Client/serveur.

L'avantage d'une telle implémentation est de gagner en productivité réseau et serveur.

4. Accroître la sécurité.

D'un point de vue de la sécurité, cela permet d'éviter que le code SQL circule sur le réseau. Les injections SQL deviennent impossibles.

Application au projet TOPAZ :

Ce sont les raisons pour lesquelles, dans la phase de développement ultérieure de l'application, des procédures stockées seront créées en remplacement du code SQL actuellement présent dans le code java.

Syntaxe de création d'une procédure stockée faite dans MySQL Query browser :

```
DELIMITER $$  
  
CREATE PROCEDURE [bd.]nomDeLaProcedure ([[DIRECTION] argument1 TYPE [, [DIRECTION] argument2 TYPE]])  
BEGIN  
    Instruction SQL ou autre;  
    Instruction SQL ou autre;  
END $$  
  
DELIMITER;
```

Elles seront appelées depuis le code java par la syntaxe :

```
CALL nomDeLaProcedure[[paramètres]];
```

4.6 - LE CODE SQL (LDD) DE CREATION DE LA BASE DE DONNEES

SQL (Structured Query Language) est langage standardisé pour communiquer avec les SGBDR (Systèmes de Gestion de Base Données Relationnelle).

Le sous-langage LDD permet de créer (CREATE), modifier, (ALTER) et supprimer (DROP) les objets de la base de données, notamment la DataBase, les Users, les tables, les Index, les procédures et les fonctions.

Voici le script de création de la base de données de test TOPAZTEST sur laquelle je travaille en développement :

```
*****  
-- Schema topaztest  
-----  
CREATE SCHEMA IF NOT EXISTS `topaztest` DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci ;  
USE `topaztest` ;  
  
-- Table `topaztest`.`User`  
-----  
DROP TABLE IF EXISTS `topaztest`.`User` ;  
  
CREATE TABLE IF NOT EXISTS `topaztest`.`User` (  
  `idUser` INT(11) UNSIGNED NOT NULL COMMENT "",  
  `firstname` VARCHAR(45) NOT NULL COMMENT "",  
  `lastname` VARCHAR(45) NOT NULL COMMENT "",  
  `login` VARCHAR(45) NOT NULL COMMENT "",  
  `password` VARCHAR(45) NOT NULL COMMENT "",  
  PRIMARY KEY (`idUser`) COMMENT "",  
  UNIQUE INDEX `login_UNIQUE` (`login` ASC) COMMENT "")  
ENGINE = InnoDB;  
  
-- Table `topaztest`.`folder`  
-----  
DROP TABLE IF EXISTS `topaztest`.`folder` ;  
  
CREATE TABLE IF NOT EXISTS `topaztest`.`folder` (  
  `id_folder` INT(11) UNSIGNED NOT NULL AUTO_INCREMENT COMMENT "",  
  `folder_name` VARCHAR(100) NOT NULL COMMENT "",  
  `folder_expanded` TINYINT(1) NULL COMMENT "",  
  `User_idUser` INT(11) UNSIGNED NOT NULL COMMENT "",  
  `folder_id_folder` INT(11) UNSIGNED NULL COMMENT "",  
  PRIMARY KEY (`id_folder`) COMMENT "",
```

```

INDEX `fk_folder_User1_idx`(`User_idUser` ASC) COMMENT '',
INDEX `fk_folder_folder1_idx`(`folder_id_folder` ASC) COMMENT '',
CONSTRAINT `fk_folder_User1`
    FOREIGN KEY (`User_idUser`)
        REFERENCES `topaztest`.`User`(`idUser`)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION,
    CONSTRAINT `fk_folder_folder1`
        FOREIGN KEY (`folder_id_folder`)
        REFERENCES `topaztest`.`folder`(`id_folder`)
        ON DELETE CASCADE
        ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```
-- Table `topaztest`.`header_type`
```

```
DROP TABLE IF EXISTS `topaztest`.`header_type` ;
```

```

CREATE TABLE IF NOT EXISTS `topaztest`.`header_type` (
    `id_header_type` INT(11) UNSIGNED NOT NULL AUTO_INCREMENT COMMENT '',
    `header_definition` VARCHAR(350) NOT NULL COMMENT '',
    PRIMARY KEY (`id_header_type`) COMMENT ''
ENGINE = InnoDB;

```

```
-- Table `topaztest`.`file`
```

```
DROP TABLE IF EXISTS `topaztest`.`file` ;
```

```

CREATE TABLE IF NOT EXISTS `topaztest`.`file` (
    `id_file` INT(11) NOT NULL AUTO_INCREMENT COMMENT '',
    `file_name` VARCHAR(100) NOT NULL COMMENT '',
    `folder_id_folder` INT(11) UNSIGNED NOT NULL COMMENT '',
    `header_type_id_header_type` INT(11) UNSIGNED NOT NULL COMMENT '',
    PRIMARY KEY (`id_file`) COMMENT '',
    UNIQUE INDEX `id_file_UNIQUE`(`id_file` ASC) COMMENT '',
    INDEX `fk_file_folder_idx`(`folder_id_folder` ASC) COMMENT '',
    INDEX `fk_file_header_type1_idx`(`header_type_id_header_type` ASC) COMMENT '',
    CONSTRAINT `fk_file_folder`
        FOREIGN KEY (`folder_id_folder`)
        REFERENCES `topaztest`.`folder`(`id_folder`)
        ON DELETE CASCADE
        ON UPDATE NO ACTION,
    CONSTRAINT `fk_file_header_type1`
        FOREIGN KEY (`header_type_id_header_type`)
        REFERENCES `topaztest`.`header_type`(`id_header_type`)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```
-- -----
-- Table `topaztest`.`file_line`  

--  

DROP TABLE IF EXISTS `topaztest`.`file_line` ;  

CREATE TABLE IF NOT EXISTS `topaztest`.`file_line` (  

    `id_file_line` INT(11) UNSIGNED NOT NULL AUTO_INCREMENT COMMENT "",  

    `file_line_number` INT(11) NULL COMMENT "",  

    `file_id_file` INT(11) NOT NULL COMMENT "",  

PRIMARY KEY (`id_file_line`) COMMENT "",  

INDEX `fk_file_line_file1_idx` (`file_id_file` ASC) COMMENT "",  

CONSTRAINT `fk_file_line_file1`  

    FOREIGN KEY (`file_id_file`)  

    REFERENCES `topaztest`.`file` (`id_file`)  

ON DELETE CASCADE  

    ON UPDATE NO ACTION)  

ENGINE = InnoDB;
```

```
-- -----
-- Table `topaztest`.`ColumnType`  

--  

DROP TABLE IF EXISTS `topaztest`.`ColumnType` ;  

CREATE TABLE IF NOT EXISTS `topaztest`.`ColumnType` (  

    `idColumnType` INT(11) NOT NULL AUTO_INCREMENT COMMENT "",  

    `type_column` VARCHAR(45) NULL COMMENT "",  

PRIMARY KEY (`idColumnType`) COMMENT "")  

ENGINE = InnoDB;
```

```
-- -----
-- Table `topaztest`.`file_column`  

--  

DROP TABLE IF EXISTS `topaztest`.`file_column` ;  

CREATE TABLE IF NOT EXISTS `topaztest`.`file_column` (  

    `id_file_column` INT(11) NOT NULL AUTO_INCREMENT COMMENT "",  

    `file_column_name` VARCHAR(100) NULL COMMENT "",  

    `type_column` VARCHAR(50) NULL COMMENT "",  

    `file_id_file` INT(11) NOT NULL COMMENT "",  

    `ColumnType_idColumnType` INT(11) NOT NULL COMMENT "",  

PRIMARY KEY (`id_file_column`) COMMENT "",  

UNIQUE INDEX `id_file_column_UNIQUE` (`id_file_column` ASC) COMMENT "",  

INDEX `fk_file_column_file1_idx` (`file_id_file` ASC) COMMENT "",  

INDEX `fk_file_column_ColumnType1_idx` (`ColumnType_idColumnType` ASC)  

COMMENT "",  

CONSTRAINT `fk_file_column_file1`  

    FOREIGN KEY (`file_id_file`)  

    REFERENCES `topaztest`.`file` (`id_file`)
```

```

ON DELETE CASCADE
    ON UPDATE NO ACTION,
    CONSTRAINT `fk_file_column_ColumnType1` 
        FOREIGN KEY (`ColumnType_idColumnType`)
            REFERENCES `topaztest`.`ColumnType`(`idColumnType`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

-- -----
-- Table `topaztest`.`line_column_value`
-- -----
DROP TABLE IF EXISTS `topaztest`.`line_column_value` ;

```

```

CREATE TABLE IF NOT EXISTS `topaztest`.`line_column_value` (
    `id_line_column_value` INT(11) UNSIGNED NOT NULL AUTO_INCREMENT COMMENT '',
    `line_column_value` VARCHAR(100) NULL COMMENT '',
    `file_column_id_file_column` INT(11) NOT NULL COMMENT '',
    `file_line_id_file_line` INT(11) UNSIGNED NOT NULL COMMENT '',
    PRIMARY KEY (`id_line_column_value`, `file_column_id_file_column`,
    `file_line_id_file_line`) COMMENT '',
    INDEX `fk_line_column_value_file_column1_idx`(`file_column_id_file_column` ASC)
COMMENT '',
    INDEX `fk_line_column_value_file_line1_idx`(`file_line_id_file_line` ASC) COMMENT '',
    CONSTRAINT `fk_line_column_value_file_column1` 
        FOREIGN KEY (`file_column_id_file_column`)
            REFERENCES `topaztest`.`file_column`(`id_file_column`)
    ON DELETE CASCADE
        ON UPDATE NO ACTION,
    CONSTRAINT `fk_line_column_value_file_line1` 
        FOREIGN KEY (`file_line_id_file_line`)
            REFERENCES `topaztest`.`file_line`(`id_file_line`)
    ON DELETE CASCADE
        ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

-- -----
-- Table `topaztest`.`UserDisplayOption`
-- -----
DROP TABLE IF EXISTS `topaztest`.`UserDisplayOption` ;

```

```

CREATE TABLE IF NOT EXISTS `topaztest`.`UserDisplayOption` (
    `idUserDisplayOption` INT(11) UNSIGNED NOT NULL COMMENT '',
    `gridBackgroundColor` VARCHAR(45) NULL COMMENT '',
    `gridDisplay` TINYINT(1) NULL COMMENT '',
    `gridSpacing` INT(5) NULL COMMENT '',
    `User_idUser` INT(11) UNSIGNED NOT NULL COMMENT '',
    PRIMARY KEY (`idUserDisplayOption`) COMMENT '',
    INDEX `fk_UserDisplayOption_User1_idx`(`User_idUser` ASC) COMMENT '',
    CONSTRAINT `fk_UserDisplayOption_User1` 

```

```

FOREIGN KEY (`User_idUser`)
REFERENCES `topaztest`.`User` (`idUser`)
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

```

-- -----
-- Table `topaztest`.`userHeaderSettings`
-- -----
DROP TABLE IF EXISTS `topaztest`.`userHeaderSettings` ;

```

```

CREATE TABLE IF NOT EXISTS `topaztest`.`userHeaderSettings` (
`iduserHeaderSettings` INT(11) UNSIGNED NOT NULL AUTO_INCREMENT COMMENT '',
`axeX_column_name` VARCHAR(20) NULL COMMENT '',
`minX` FLOAT NULL COMMENT '',
`maxX` FLOAT NULL COMMENT '',
`labelX` VARCHAR(20) NULL COMMENT '',
`axeY_column_name` VARCHAR(20) NULL COMMENT '',
`minY` FLOAT NULL COMMENT '',
` maxY` FLOAT NULL COMMENT '',
`labelY` VARCHAR(20) NULL COMMENT '',
`axeZ_column_name` VARCHAR(20) NULL COMMENT '',
`minZ` FLOAT NULL COMMENT '',
`maxZ` FLOAT NULL COMMENT '',
`labelZ` VARCHAR(20) NULL COMMENT '',
`colormap_name` VARCHAR(25) NULL COMMENT '',
`userHeaderSettingscol` VARCHAR(45) NULL COMMENT '',
`UserDisplayOption_idUserDisplayOption` INT(11) UNSIGNED NOT NULL COMMENT '',
PRIMARY KEY (`iduserHeaderSettings`) COMMENT '',
INDEX `fk_userHeaderSettings_UserDisplayOption1_idx`(
(`UserDisplayOption_idUserDisplayOption` ASC) COMMENT '',
CONSTRAINT `fk_userHeaderSettings_UserDisplayOption1`
FOREIGN KEY (`UserDisplayOption_idUserDisplayOption`)
REFERENCES `topaztest`.`UserDisplayOption` (`idUserDisplayOption`)
ON DELETE NO ACTION
ON UPDATE NO ACTION)
ENGINE = InnoDB;

```

CHAPITRE 5 - CONCEPTION DE L'APPLICATION

-

5.1 - LES DIAGRAMMES DE SEQUENCE DETAILLÉS

En UML, le diagramme de séquence (DSEQ) permet de décrire et de formaliser les interactions entre les acteurs et le système ou les classes, par l'intermédiaire de messages, d'un point de vue chronologique et spatial.

Le temps est représenté à la verticale et l'espace à l'horizontal.

Dans la phase de conception de l'application, ce diagramme est détaillé et les messages correspondent aux messages des langages informatiques.

Le diagramme de séquence détaillé représente, selon le design pattern (patron de conception) ECB (Entity - Control - Boundary) que je vais utiliser pour le développement de l'application TOPAZ :

- les utilisateurs qui interagissent avec le système au moyen des Boundaries,
- les Boundaries qui envoient des instructions aux contrôleurs,
- les Contrôleurs qui font l'intermédiaire entre les Boundaries et les entités et qui orchestrent l'exécution des commandes,
- Les modèles (Entities et les DAOs) qui représentent les données du système.

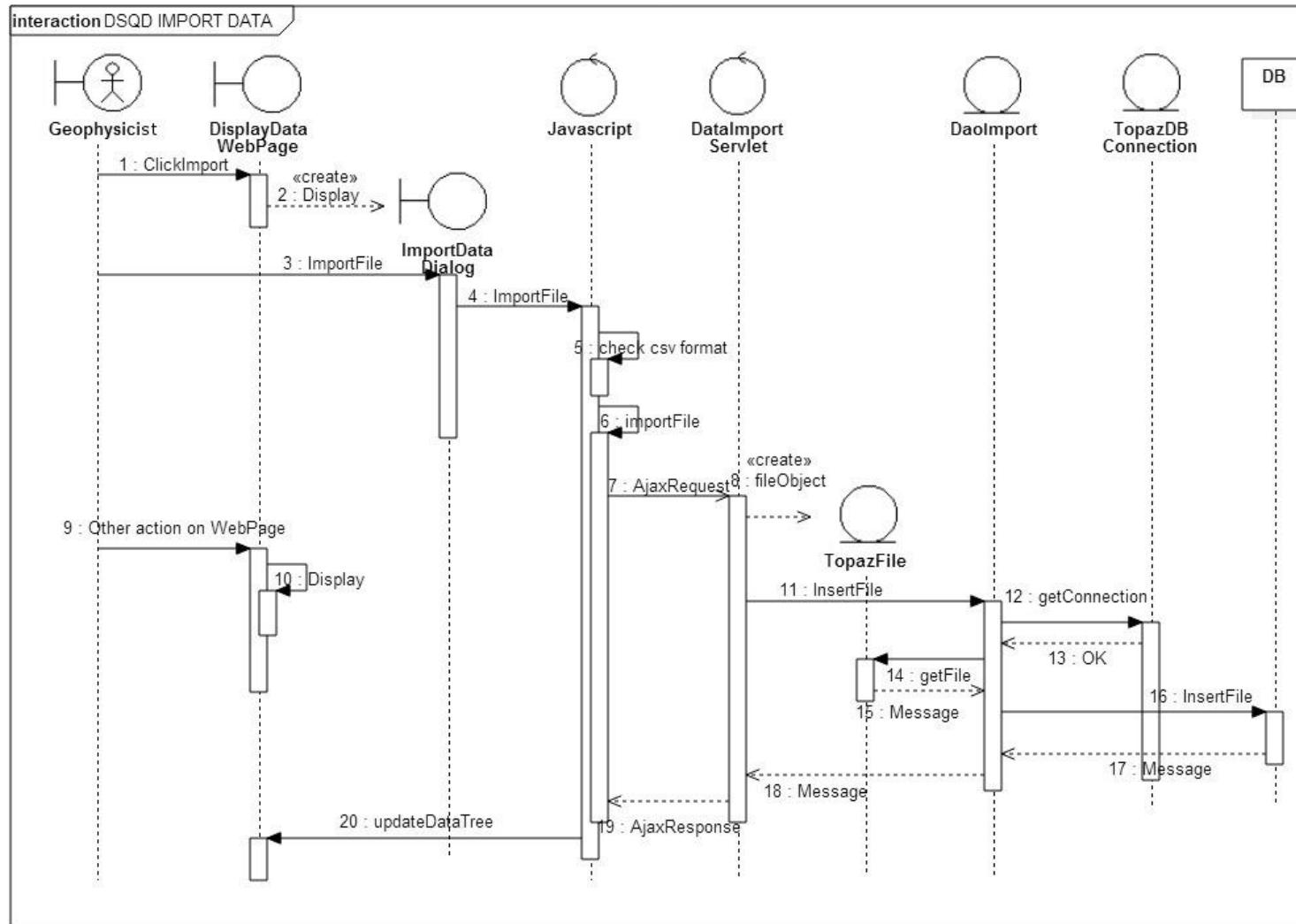
Dans l'application TOPAZ :

Boundaries : Pages Web

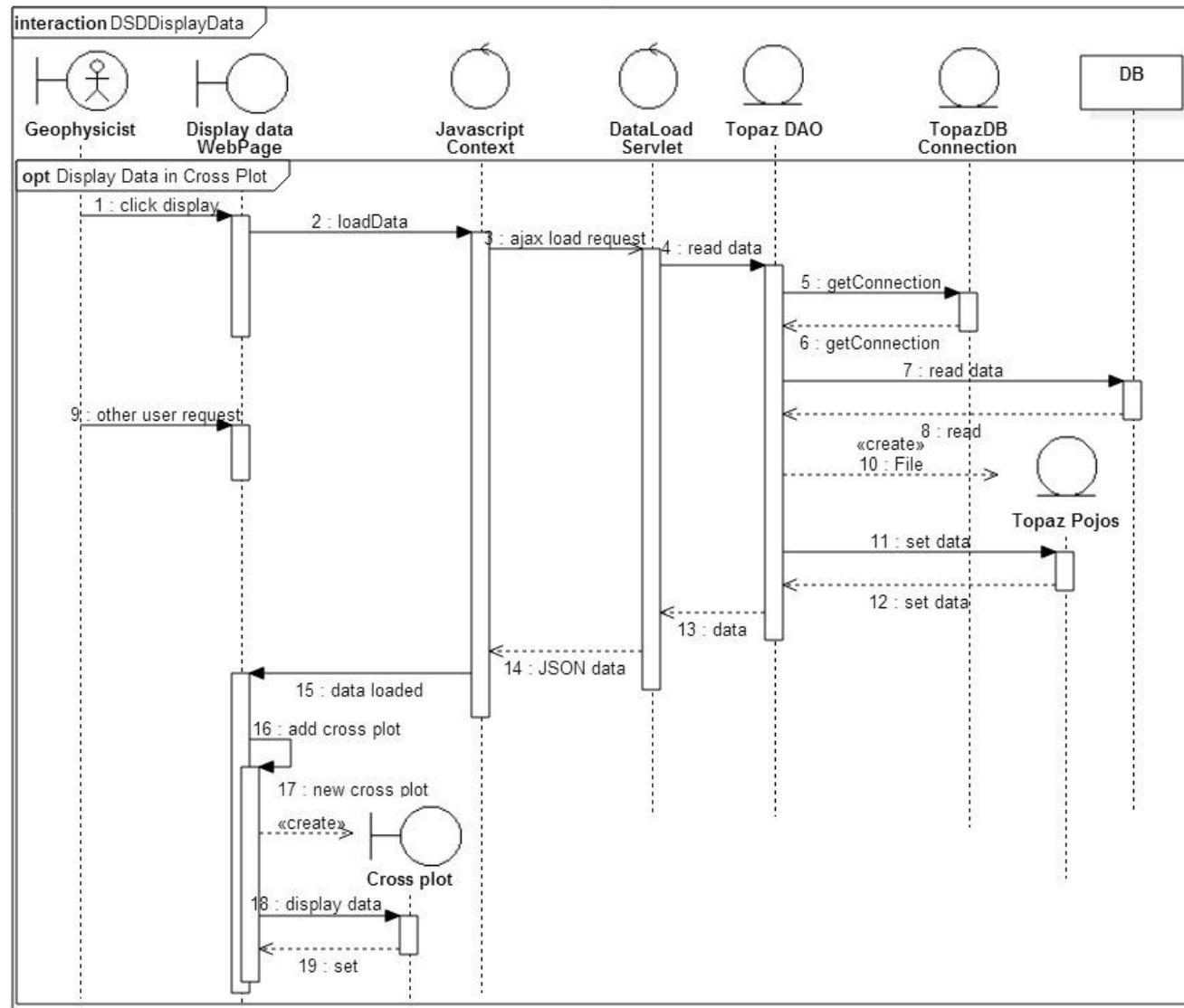
Controller : JSP (Java Servlet Page)

Entities : classes POJOS en Java

5.1.1 Le Diagramme de Séquence Détailé « IMPORT DATA »



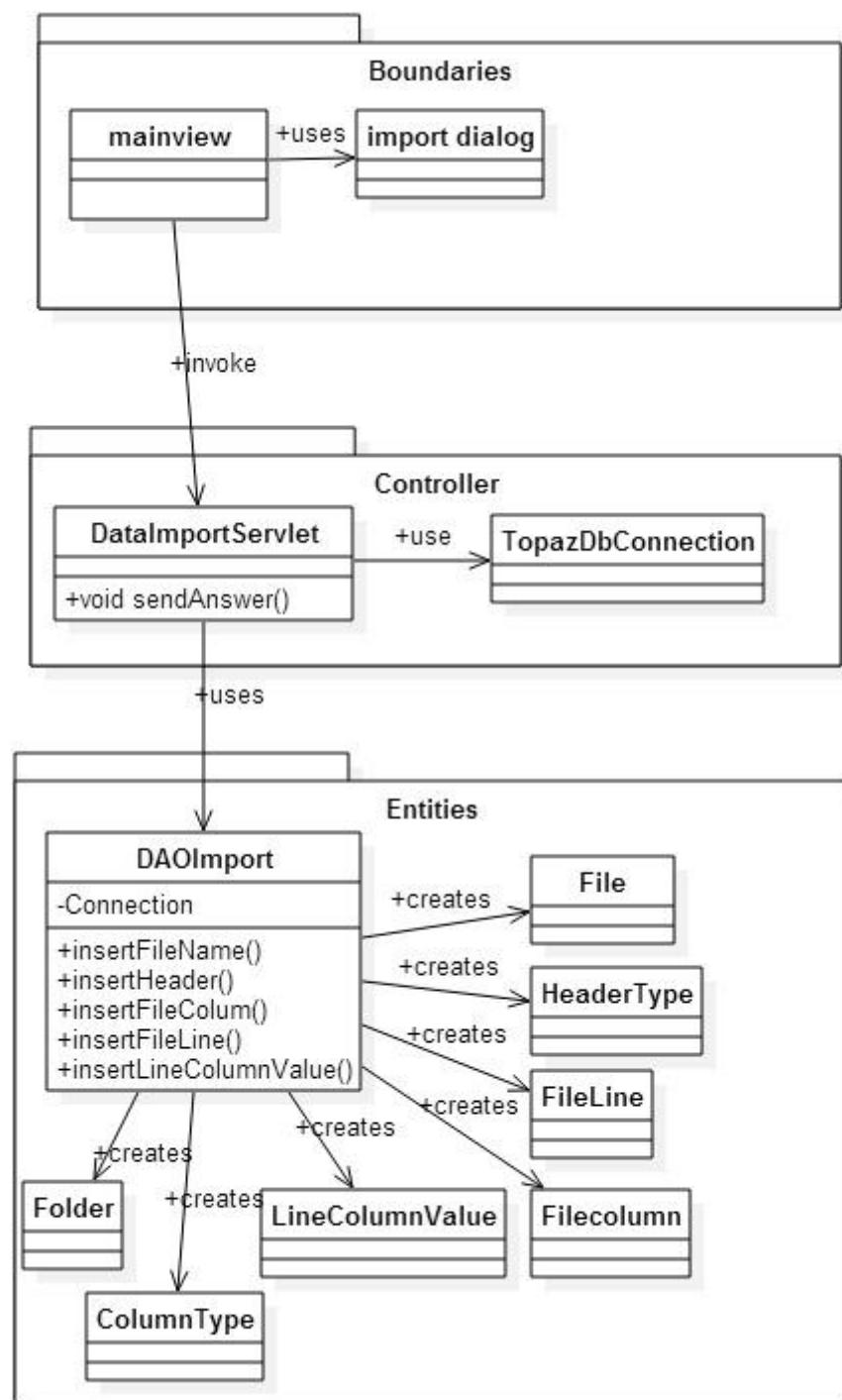
5.1.2 Le Diagramme de Séquence Détailé « DISPLAY DATA »



5.2 - LES DIAGRAMMES DE CLASSES PARTICIPANTES

5.2.1 Le Diagramme de classes participantes « IMPORT DATA »

Les classes du diagramme de séquence détaillé « Import Data » y sont représentées.



CHAPITRE 6 - DEVELOPPEMENT

6.1 - DONNEES D'ENTREE

Il est intéressant d'avoir un aperçu du type et du format des données importées. Voici ci-après 2 exemples de début de fichier (les fichiers contiennent en général plus de 10 000 lignes) :

On peut voir leur forme très différente.

Exemple 1.

```
Type,Line,Point,Index,Code,X,Y,Z,Vertical Ttime,Point depth,Water Depth,Julian Day,Hours,Minutes,Seconds
S, 1924.00,2915,1,V1,156867.3,374132.7,3.1,0,0,0,76,0,0,0
S, 1924.00,2913,1,V1,156866.9,374117.5,2.9,0,0,0,76,0,0,0
S, 1924.00,2911,1,V1,156866.6,374102,3.2,0,0,0,76,0,0,0
S, 1924.00,2909,1,V1,156866.8,374087.3,3.1,0,0,0,76,0,0,0
S, 1936.00,2915,1,V1,156956.8,374131.8,4.1,0,0,0,76,0,0,0
S, 1936.00,2913,1,V1,156957.2,374117.7,4.3,0,0,0,76,0,0,0
```

Exemple 2.

```
Line,Num,Idx,SPSX,SPSY,SPSZ,COGX,COGY,COGZ,COGState,COGDev,REALDEV,Grp,Vib,Acq,Drive,Avg
_Pha,Avg_Dst,Avg_Frc,Avg_GS,Avg_GV,Pik_Pha,Pik_Dst,Pik_Frc,Status,X,Y,Z,Insec,JDay,Time,Hacc,Pa
ttern
83061,68423,1,730287.5,3188262.5,0,730287.6,3188261.4,241.5,3,1.1,1.10453610180937,2,24,1,7
5,1,13,75,28,51,-4,17,81,1,730281.5,3188261.3,241.5,29373,321,080933,0.8,
83061,68423,1,730287.5,3188262.5,0,730287.6,3188261.4,241.5,3,1.1,1.10453610180937,2,23,1,7
5,1,9,74,25,44,-3,15,79,1,730293.7,3188261.6,241.5,29373,321,080933,0.8,
82589,68513,1,731412.5,3182362.5,0,731405,3182361.6,242.5,4,7.6,7.55380698719741,5,4,1,75,2,
18,74,37,48,4,29,80,1,731416.9,3182362.3,242.5,29465,321,081105,0.7,
82589,68513,1,731412.5,3182362.5,0,731405,3182361.6,242.5,4,7.6,7.55380698719741,5,1,1,75,2,
12,74,37,35,4,18,83,1,731405,3182361.6,242.5,29465,321,081105,0.7,
82589,68513,2,731412.5,3182362.5,0,9999,9999,9999,9999,9999,9999,9999,5,4,1,75,0,0,0,0,0,0,14,0,
0,0,29499,321,081139,0,
82589,68513,2,731412.5,3182362.5,0,9999,9999,9999,9999,9999,9999,9999,5,1,1,75,0,0,0,0,0,0,14,0,
0,0,29499,321,081139,0,
```

6.2 - INTERFACE : PAGE « DISPLAY DATA »

6.2.1 Page HTML principale

Très simpliste, la majeure partie des interfaces est codée en javascript . On notera d'ailleurs que la plupart des lignes sert à importer les feuilles de style des libraires utilisées.

```
<!DOCTYPE html>

<html>
<head>
    <meta charset="UTF-8"/>
    <title>Topaz Web Viewer</title>

    <script data-main="js/config.js" src="thirdparty/js/requirejs/requirejs-2.1.11.js"></script>
    <script type="text/javascript">
        require(['js/mainview.js']);
    </script>

    <link href="thirdparty/js/jquery-tagit/tagit.css" rel="stylesheet">
    <link href="thirdparty/js/w2ui/1.4.2/w2ui.min.css" rel="stylesheet">
    <link href="thirdparty/js/contextMenu/jquery.contextMenu-1.6.5.css" rel="stylesheet">
    <link href="thirdparty/js/toastr/toastr-2.0.3.css" rel="stylesheet">
    <link href="thirdparty/js/fontawesome/4.1.0/css/font-awesome.css" rel="stylesheet">

    <link href="css/main.css" rel="stylesheet"/>

</head>

<body>
    <header>
        <h1>Welcome to Topaz</h1>
    </header>

    <div class="row">
        <div class="span12">
            <div id="data-tree" style="height: 1000px;"></div>
        </div>
    </div>

    <div id="geotoolkit_ui_open_file_dialog"></div>
    <input id="hiddenFileInput" type="file" style="visibility: hidden; position: absolute; top: 0;" >

</body>
</html>
```

6.2.2 Feuille de style

Peu pertinent dans ce contexte très orienté Javascript avec l'utilisation des widgets de GeoToolkit, la feuille de style reste assez sommaire.

La version complète est disponible en annexe.

En voici un extrait :

```
/* HTML tag Definition */
body {
    margin: 0 auto;
    width: 98%;
    font-family: "Century Gothic", Arial;
}

header {
    background-color: #f0deef;
    padding-bottom: 4px;
}

footer {
    height: 50px;
    background-color: #d7d7d7;
}

footer nav ul li {
    display: inline-block;
    padding-top: 15px
}

header nav ul li a {
    text-decoration: none;
    color: white;
    font-weight: bold;
}

aside {
    background-color: #CCC;
    float: right;
    margin: 15px 0 15px 0;
    width: 195px;
    padding: 10px;
    text-align: justify;
}
```

6.2.3 Classes Javascript

Voici des extraits de méthodes javascript les plus représentatives.
Ici du jQuery ou presque... du w2ui utilisé chez CGG.

Page « Display Data » : Initialisation de la vue principale

```
// on window onload :
function init() {
    // first define a layout
    $('#data-tree').w2layout({
        name: 'w2ui-main-layout',
        // define the number and size of panels of the layout
        panels: [
            {type: 'top', size: 28},
            {type: 'left', size: 300, resizable: true, style: 'background-color: #F5F6F7;', content: 'left'},
            {type: 'main', style: 'background-color: #F5F6F7; padding: 5px;'}
        ]
    });
}

// then define the toolbar, the sidebar and the tabs
w2ui['w2ui-main-layout'].content('top', createToolbar());
w2ui['w2ui-main-layout'].content('left', createDataTree());
w2ui['w2ui-main-layout'].content('main',
    "<div id='tabs' style='width: 100%;'></div>" +
    "<div id='selected-tab' style='padding: 4px 4px'><canvas id='crossplot' /></div>"
);

$('#tabs').w2tabs({
    name: 'data-tabs',
    // if several tabs are open, onClick on the label of the tab, display the content of the
    // selected tab on the panel
    onClick: function (event) {
        var widget = crossplots[event.target];
        crossplot.addShapeToCanvas("crossplot", widget);
    }
});

// Function that update the list of imported files in data tree via a json query to the server Web
updateDataTreeContent();

// Request the HTML input type FILE "hiddenFileInput" to invoke the function onFileImport
// when a change event occurs (that is when a file is selected by the user in the OS File chooser
// and he
// clicked on OK/Open).
$('#hiddenFileInput').change(onFileImport);

} // init
```

Création de la toolbar horizontale

```

function createToolbar() {
    return $(()).w2toolbar({
        name: 'main_toolbar',
        items: [
            {type: 'button', id: 'mtb-import-data', caption: 'Import', icon: 'fa fa-database'},
            {type: 'button', id: 'mtb-display-data', caption: 'Display', icon: 'fa fa-eye'},
            {type: 'break', id: 'break1'},
            {type: 'button', id: 'mtb-new-folder', caption: 'New Folder', icon: 'fa fa-folder'},
            {type: 'button', id: 'mtb-rename-item', caption: 'Rename', icon: 'fa fa-text-height'},
            {type: 'button', id: 'mtb-delete-item', caption: 'Delete', icon: 'fa fa-trash-o'},
            {type: 'spacer'},
            {type: 'button', id: 'mtb-home', caption: 'Home', icon: 'fa fa-home'}
        ],
        onClick: function(event) {
            onClickToolbar (event);
        }
    });
} // createToolbar
    
```

Fonctions appelées sur le onClick des boutons de la toolbar

```

function onClickToolbar(event) {
    // onClick event of every button, a function is called.
    switch (event.target) {
        // onClick on the "Import" button of the toolbar, it calls the function onClickImport
        case 'mtb-import-data':
            onClickImport(event);
            break;
        // OnClick on display, open a new tab which displays the content (graphic & values) of the
        // selected file
        case 'mtb-display-data': // DOING
            break;
        case 'mtb-new-folder': // TO DO
            break;
        case 'mtb-rename-item': // TO DO
            break;
        case 'mtb-delete-data': // TO DO
            break;
        case 'mtb-home': // TO DO
            break;
    }
} // onClickToolbar
    
```

```

// Function that opens the OS file chooser. The <input type="file"> HTML control gives the web page
// (and the server) permission to access to the information (name, type, size) of the selected file.
function onClickImport(evt) {
    // connect the HTML input (which is hidden) and Click on it to open the OS file chooser on client
    $('#hiddenFileInput').click();
}
    
```

Import (upload) du fichier vers le serveur Web

```

// Invoked when a user has selected a file in the OS file chooser and click on OK
// Function that adds the file selected in a list and check if it's a csv file before calling the function
importFile.

function onFileImport(evt) {
    evt.stopPropagation();
// cancel default action of event
    evt.preventDefault();
// Put the selected files in the File Chooser in a FileList.
// var "files" is a FileList object (it's a HTML5 <input type file> property which allows to add the files
selected in the control in a list).
    var files = evt.target.files;
// Get the first selected file in the FileList
    var file = files[0];
// Check if the extension of the file is csv
    if (file.name.substring(file.name.length-4) == ".csv") {
        importFile(file);
    }
// TO DO : Else : Add a dialog "import that file not allowed"
}

// Read the file provided and send it as parameter to the server
function importFile(file) {
// instantiate a javascript FileReader
    var reader = new FileReader();
// read the file as text
    reader.readAsText(file);
// onLoadend event, call the function that creates a json object and sends jsonData to the server
    reader.onloadend = function(evt){
        var content = evt.target.result;
// Replace Carriage Return by ""
        content = content.replace(/\r/g, "");
// split the content (text) on the \n separator in a "lines" variable which is a table.
        var lines = content.split('\n');

// instantiate a json object
        var jsonData = {};
// add the name of the file (a key-value pair)
        jsonData["name"] = file.name;
// add the number of datalines (= lines - header)
        jsonData["dataLinesCount"] = lines.length-1;
// add the header of the file (the first line of the table "lines")
        jsonData["header"] = lines[0];
// loop on lines to add each line of data in the json object
        for(i = 1; i<lines.length; i++) {
            if(lines[i] != "") {
                jsonData[i] = lines[i];
            }
        }

// send the jsonData to the server (data-import servlet)
        _queryJsonApi("/data-import", null, jsonData);
    };
} // importFile

```

Méthode générique AJAX pour exécuter une requête asynchrone vers le serveur (servet /Control).

```
// Generic method used to invoke AJAX request on our topaz server
function _queryJsonApi(query, jsonProcessor, postData, onErrorCallback) {
// Defines jQuery AJAX common settings
    var settings = { contentType: 'application/json', cache: false };
    if (postData) {
        // POST method case : define the method and set the posted data
        settings.type = 'POST';
        // convert data as JSON string
        settings.data = JSON.stringify(postData);
    }
    else {
        // GET method case : no data to be provided
        settings.type = 'GET';
    }
// Creates the AJAX URL by combining topaz server address and provided AJAX URL
var url = "/topaz-servlet" + query;

// Handle the case where there is no AJAX processing method (this may happen in the POST case)
if (!jsonProcessor) {
    jsonProcessor = function () {
    };
}

// Invoke JQuery AJAX base on URL and setting that have been defined earlier
$.ajax(url, settings).error(function (e) {
// Call the error method if provided by the user
    if (onErrorCallback) {
        onErrorCallback(e);
    }
// Request JQuery to call "jsonProcessor" method upon sucess
    }).success(jsonProcessor);
} //_queryJsonApi
```

Création de la vue Cross Plot, widget de la bibliothèque JavaScript INT

Voici un extrait du code de génération et d'affichage d'un crossplot

```

function init(canvasName, title, tool) {
    var crossplotWidget = createWidget(canvasName, title, tool);
    setupTools(crossplotWidget);
    populateData(crossplotWidget);
    updateScrollBars(crossplotWidget);
    return crossplotWidget;
} // init

// Create a cross plot widget with its attributs
function createWidget(canvasName, title, tool) {
// define the style of its header by creating a "new geotoolkit.attributes" objet
    var styleHeader = new geotoolkit.attributes.TextStyle({
        // key : value pairs
        'color': "blue",
        'font': "20px Arial"
    });
// define the cross plot
    var widget = new geotoolkit.widgets.CrossPlot({
// define the shape which will contain the crossplot
        'bounds': new geotoolkit.util.Rect(0, 0, plotWidth, plotHeight),
// define the tool of the cross plot (scrollbars)
        'tools': {
            'horizontalscroll': { 'visible': false },
            'verticalscroll': { 'visible': false }
        },
// define the header of the cross plot (title and annotationsize)
        'header': {
            'title': {
                'text': title,
                'textstyle': styleHeader
            },
            'annotationsize': 30
        },
// define the X axe of the cross plot (label and annotationsize)
        'x': {
            'label': {
                'text': "x axis",
                'textstyle': Helpers.axisColor()['style']
            },
            'annotationsize': 50
        },
// define the Y axe of the cross plot (label and annotationsize)
        'y': {
            'label': {
                'text': "y axis",
                'textstyle': Helpers.axisColor()['style']
            },
            'annotationsize': 65
        },
}
    
```

```
// define the Z axe of the cross plot (annotationSize)
    'z': { 'annotationSize': 20 }
}); // var widget

addShapeToCanvas(canvasName, widget);

if (tool){
    //Connect Widget Tools to Plot (one toolsContainer can be used for several widgets inside the same plot)
    var toolsContainer = new geotoolkit.controls.tools.ToolsContainer(plot);
    toolsContainer.add(widget.getTool());
}
return widget;
} // createWidget

// Create a cross plot widget with its attributs
function addShapeToCanvas(canvasName, shape) {
    var canvas = document.getElementById(canvasName);
    // Create a new Plot object from the canvas and group
    plot = new geotoolkit.plot.Plot({
        'canvasElement' : canvas,
        'root' :
            new geotoolkit.scene.Group()
                .setBounds(new geotoolkit.util.Rect(0, 0, plotWidth, plotHeight))
                .setAutoModelLimitsMode(true)
                .addChild(shape),
        'autoUpdate' : true
    });
    plot.setSize(plotWidth, plotHeight);

    return plot;
}
```

6.3 - CONTROLLER

Les requêtes en provenance de l'IHM (le bouton « import » de la page « Display Data ») aboutissent sur cette servlet qui, en l'état actuel du développement, récupère les données du fichier envoyées par le client, puis instancie un DAO et appelle la méthode insertFile afin de faire persister ces données dans la base de données.

```
/*
 * @Author LELLOUCHE Catherine
 * Date: 11-Aug-15
 */
@WebServlet(name = "DataImportServlet") // The servlet name defined in web.xml

public class DataImportServlet extends HttpServlet {

    // Method invoked on POST request
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {
        sendAnswer(request, response);
    }

    // Method invoked on GET request : does the same for test purpose
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {
        sendAnswer(request, response);
    }

    private void sendAnswer(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {

        // Defines some tooling to retrieve posted JSON data
        ObjectMapper mapper = new ObjectMapper();
        JsonFactory factory = mapper.getFactory();

        // Create the JSON parser based on the request input stream ("reader")
        JsonParser jp = factory.createParser(request.getReader());

        // Retrieve the JSON node from the mapper and factory
        JsonNode node = mapper.readTree(jp);

        // We are now able to retrieve the data sent from the web client (the name of the file, the header line
        // and the data in a list)
        String fileName = node.get("name").textValue();
        String header = node.get("header").textValue();
        Integer dataLinesCount = node.get("dataLinesCount").intValue();
        List<String> dataLines = new ArrayList<>();
    }
}
```

```
// extract all lines of data from the JSON node and add them to the list
for (int i = 1; i < dataLinesCount; i++) {
    String key = String.valueOf(i);
    JsonNode jsonNode = node.get(key);
    String aLine = jsonNode.textValue();
    dataLines.add(aLine);
}

// Retrieve the SQL connection parameters
String propertiesFilePath = DataImportServlet.class.getResource("/topaz-
db.properties").getPath();

// Create the connection to TOPAZ database
Connection sqlConnection = TopazDbConnection.connect(propertiesFilePath);

// Instantiate a DAO with a connection parameter
DaoImportData daoImportData = new DaoImportData(sqlConnection);

// Call insertFile method on DAOImportData object
daoImportData.insertFile(fileName);

}
} // class DataImportServlet
```

6.4 - ENTITIES/MODELS ET DAO

La partie serveur basée sur Jetty est implémentée en Java.

6.4.1 Classes de base commune

Les classes TopazDataFile et TopazFolder héritent de la classe abstraite AbstractTopazDataItem en application du pattern composite du GOF que nous avons choisi d'utiliser.

Un dossier et un fichier ont en commun un label et un id.

```
/**  
 * @Author Lellouche Catherine  
 * Date: 27-Aug-15  
 */  
public abstract class AbstractTopazDataItem {  
    private String label;  
    private int idItem;  
  
    public String getLabel() {  
        return label;  
    }  
  
    public void setLabel(String label) {  
        this.label = label;  
    }  
  
    public int getIdItem() {  
        return idItem;  
    }  
  
    public void setIdItem(int idItem) {  
        this.idItem = idItem;  
    }  
}  
  
/**  
 * @Author Lellouche Catherine  
 * Date: 27-Aug-15  
 */  
public class TopazDataFile extends AbstractTopazDataItem {  
    private String fileUrl;  
  
    public String getFileUrl() {  
        return fileUrl;  
    }  
  
    public void setFileUrl(String fileUrl) {  
        this.fileUrl = fileUrl;  
    }  
} // class AbstractTopazDataItem
```

6.4.2 Exemple d'entité avec UserDisplayOptions

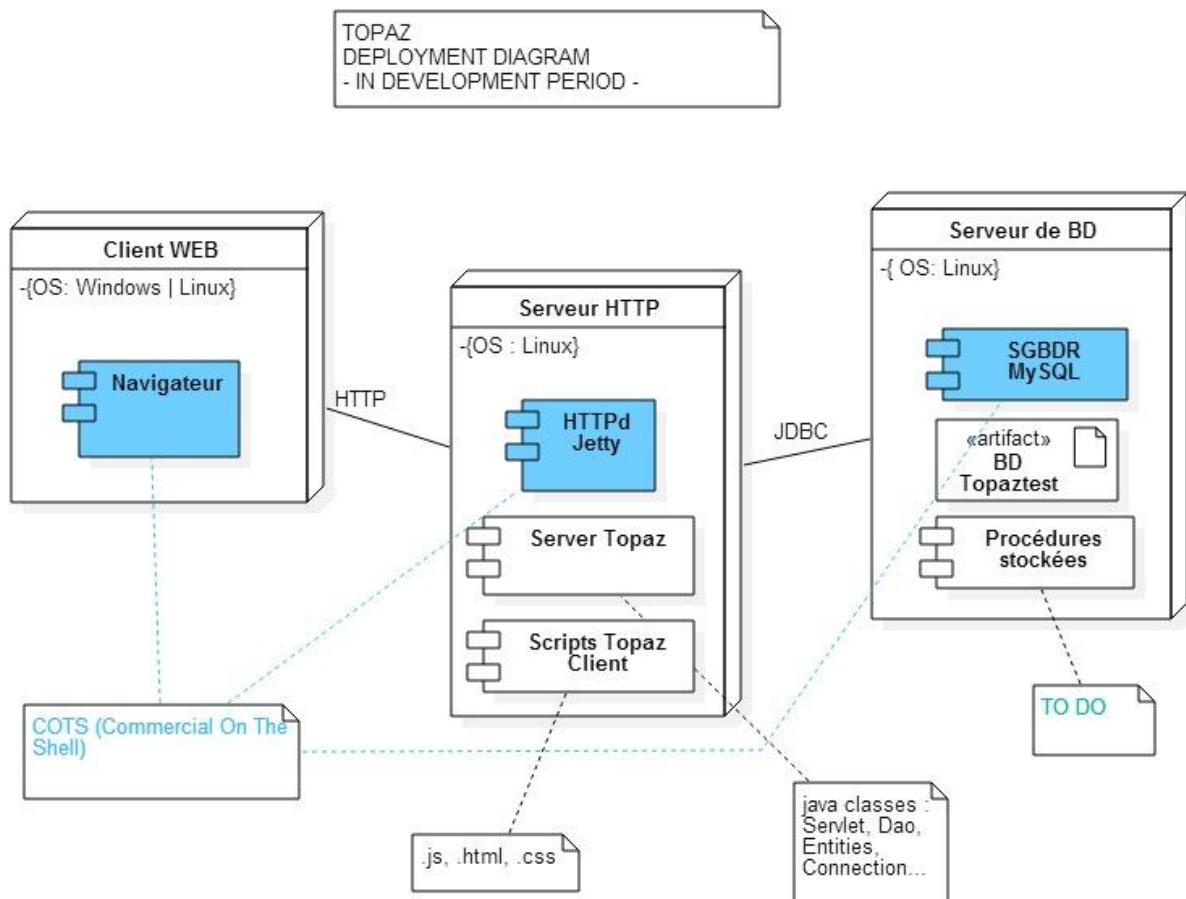
```
/**  
 * @Author Lellouche Catherine  
 * Date: 27-Aug-15  
 */  
public class TopazUserDisplayOptions {  
    private int idUserDisplayOption;  
    private Color gridBackgroundColor;  
    private boolean gridDisplayed;  
    private int gridSpacing;  
    private List<TopazUserHeaderSettings> topazUserHeaderDisplayOptionList = new  
ArrayList<>();  
  
    public int getIdUserDisplayOption() {  
        return idUserDisplayOption;  
    }  
  
    public void setIdUserDisplayOption(int idUserDisplayOption) {  
        this.idUserDisplayOption = idUserDisplayOption;  
    }  
  
    public Color getGridBackgroundColor() {  
        return gridBackgroundColor;  
    }  
  
    public void setGridBackgroundColor(Color gridBackgroundColor) {  
        this.gridBackgroundColor = gridBackgroundColor;  
    }  
  
    public boolean isGridDisplayed() {  
        return gridDisplayed;  
    }  
  
    public void setGridDisplayed(boolean gridDisplayed) {  
        this.gridDisplayed = gridDisplayed;  
    }  
  
    public int getGridSpacing() {  
        return gridSpacing;  
    }  
  
    public void setGridSpacing(int gridSpacing) {  
        this.gridSpacing = gridSpacing;  
    }  
  
    public List<TopazUserHeaderSettings> getTopazUserHeaderDisplayOptionList() {  
        return topazUserHeaderDisplayOptionList;  
    }  
} // class TopazUserDisplayOptions
```

6.4.3 DAO d'import des données dans la base de données Topaz

```
public class DaoImportData {  
  
    private Connection connection;  
  
    public DaoImportData(Connection aConnection) {  
        connection = aConnection;  
    }  
  
    /**  
     * insert in file table  
     * @param fileName      //FileName parameter comes from the DataImportServlet  
     */  
    public void insertFile(String fileName) {  
  
        try {  
            // The string request for insert  
            String insertName = "INSERT INTO topaztest.file (file_name, folder_id_folder) VALUES(?, ?)";  
  
            // create Statement with SQL string request  
            PreparedStatement pstInsertFileName = connection.prepareStatement(insertName,  
Statement.RETURN_GENERATED_KEYS);  
  
            //Give value to the parameter  
            pstInsertFileName.setString(1, fileName);  
            pstInsertFileName.setInt(2, 1);  
  
            // Execute the SQL request and get the count of inserted line  
            int resultInsertFileName = pstInsertFileName.executeUpdate();  
  
            // get the new generated key of inserted file to use it in the next SQL request for inserting the  
            header  
            ResultSet insertedFile = pstInsertFileName.getGeneratedKeys();  
            insertedFile.next();  
            int idAddedFile = insertedFile.getInt(1);  
            insertedFile.close();  
            pstInsertFileName.close();  
            connection.commit();  
            System.out.println("INserted File : "+idAddedFile);  
  
        } catch (SQLException e) {  
            e.printStackTrace();  
        }  
    }  
} // class DaoImportData
```

CHAPITRE 7 - DEPLOIEMENT

7.1 - LE DIAGRAMME DE DEPLOIEMENT



7.2 - LE DEPLOIEMENT

Présentation :

Le diagramme de déploiement sert à représenter l'utilisation de l'infrastructure physique par le système, la manière dont les **composants** du système sont répartis ainsi que leurs relations entre eux.

Les éléments utilisés par un **diagramme de déploiement** sont principalement les **nœuds**, les **composants**, les **associations** et les **artefacts**.

Les nœuds (*nodes*), représentés par les cubes en trois dimensions, sont des composants mécaniques de l'infrastructure tel un [routeur](#), un [ordinateur](#), un [assistant personnel](#)...

Les **composants**, représentés par des boîtes rectangulaires avec deux rectangles sortant du côté gauche, sont les différentes parties du système étudié.

Les **associations**, représentées par de simples lignes sont des liens de communication entre les différents **composants** du système.

Un **artefact**, dans ce contexte, est une manière de définir un fichier, un programme, une bibliothèque ou une base de données construite ou modifiée dans un projet.

Ces **artefacts** mettent en œuvre des collections de composants qui sont consommées ou créées durant l'une des étapes du processus de déploiement.

Application :

Pour l'application Topaz, nous avons le déploiement suivant :

- un poste de travail côté client léger qui comprend le navigateur Web avec lequel le géophysicien va accéder à l'application.
- Un serveur Web dédié à la consultation distante des clients,
- Un serveur de base de données qui contient le schéma de la base de données, les données elles-mêmes, ainsi que les procédures stockées.

CHAPITRE 8 - CONCLUSION

Le développement d'un projet informatique est un processus évolutif, assimilable à de la recherche, et je suis très fière d'avoir eu la chance de travailler avec une équipe très qualifiée qui m'a fait partager ses compétences et son envie de faire toujours mieux : de mieux organiser le travail, dé répondre aux besoins des utilisateurs tout en leur facilitant la tâche, de faciliter la maintenance ultérieure des logiciels livrés et de réduire la dette technique.

Le sujet de mon stage, « : Web viewer de données sismiques », pouvait me paraître au premier abord « léger » car il tenait en quelques mots.

J'ai aujourd'hui pris conscience de la richesse du projet sur lequel j'ai travaillé.

Cette expérience a enrichi mes connaissances, a ouvert mes horizons de conceptrice développeur informatique et m'a donné de toute évidence envie de les approfondir.

Je pense que ces ingénieurs et développeurs que j'ai rencontrés ne sont pas « bons » dans l'absolu, ils sont « bons » parce qu'ils travaillent et qu'ils remettent en question leur savoir en permanence, en toute humilité.

Je tends vers cet objectif.

Mes premiers résultats m'ont attirés les compliments de mes collègues et j'ose ainsi penser que je suis sur la bonne voie.

Néanmoins il est évident, à l'heure où j'écris ces lignes, que la durée du stage était trop courte pour terminer d'implémenter ce projet. C'est la raison pour laquelle j'ai décidé de m'investir davantage et que j'ai proposé à l'entreprise CGG, qui l'a accepté, de prolonger mon stage pour continuer le développement de TOPAZ.

CHAPITRE 9 - ANNEXES

9.1 - CORRESPONDANCES PROJET/REAC.

COMPETENCES DU REAC	CORRESPONDANCES DANS LE PROJET
Développer des composants d'interface	
Maquetter une application	UML (Diagrammes de Cas d'Utilisation, d'activité et de navigation) avec StarUml Maquettes de l'IHM avec Balsamiq
Développer une interface utilisateur	
Développer des composants d'accès aux données	Langage serveur Java
Développer des pages web en lien avec une base de données	HTML5, CSS3, Javascript 5, W2ui (jQuery), AJAX
Développer la persistance des données	
Concevoir une base de données	Modèle Conceptuel des Données (MCD) UML (Diagramme de classes) avec StarUml Modèle Physique des Données (MPD) avec MySQL WorkBench
Mettre en place une base de données	SQL (LDD) : Création de la BD, Création des tables et mise en place des contraintes (FK) avec MySQL Query Browser
Développer des composants dans le langage d'une base de données	Procédures stockées avec P/SQL, le langage procédural de MySQL
Utiliser l'anglais dans son activité professionnelle en informatique	Diagrammes UML, maquettes, développement, et documentation en anglais
Développer une application n-tiers.	
Concevoir une application	UML (Diagramme de Séquence Détailé, Diagramme de Classes participantes)
Collaborer à la gestion d'un projet informatique	Gestion du projet avec la méthode SCRUM et avec JIRA
Développer des composants métier	Classes « entités » avec Java (POJO)
Construire une application organisée en couches	Application du design pattern ECB (cf les diagrammes de séquences détaillés)
Développer une application de mobilité numérique	
Préparer et exécuter les plans de tests d'une application	
Préparer et exécuter le déploiement d'une application	UML (Diagramme de Déploiement) avec StarUml Installation de la BD Test (topaztest)

9.2 - SYNTHESE DES OUTILS UTILISES.

Outils de développement	
Outil	Fonction
IntelliJ IDEA	IDE
Maven	gestionnaire de dépendances
Perforce	Travail en versioning
Jenkins	Intégration continue
SONAR	Analyse de la qualité de code
JIRA	Gestion de projet Agile en mode collaboratif
CONFLUENCE	Logiciel de travail collaboratif
Balsamiq	Outil de maquettage
StarUML	Modéliser en UML
MySQL Workbench	Elaboration du modèle physique de données (MPD) et génération du script de création de la base de données.
MySQL Query Browser	Execution du script SQL de création de la DB sur la station Linux. Gestion de la BD.
Serveur d'application Web	Jetty
SGBDR	MySQL
Langages et librairies	
Nom	Commentaires
Langage	
HTML 5, CSS3, Javascript 5	Côté IHM (Interface Homme Machine)
AJAX (Asynchronous javascript and XML)	Permet des requêtes asynchrones vers le serveur et de mettre à jour une portion de page dans le navigateur sans recharger toute la page depuis le serveur
Java (JDK 8)	Côté serveur d'application Web
Librairies	
GeoToolkit.Js	Librairie de widgets développée en javascript 5 fournie par la société INT
W2ui	The w2ui library is a set jQuery plugins for front-end development of data driven web applications. It is not a adhoc port to jQuery, but was initially developed with jQuery in mind. The library heavily uses HTML5 and CSS3 and yet supports all major modern browsers. Latest Chrome, FireFox 7+, Safari 5+ and IE 9+ are among supported browsers.
Require.js	It's a JavaScript file and module loader which improve the speed and quality of the code.

9.3 - LE METIER DE CGG.

LE METIER DE CGG

Notions sur l'exploration sismique

SOMMAIRE

- 9.3.1 - EXPLORATION SISMIQUE : but et fondamentaux
- 9.3.2 - ACQUISITION DES DONNEES
 - 9.3.2.1 - Acquisition terrestre
 - 9.3.2.2 - Acquisition maritime
- 9.3.3 - TRAITEMENT DES DONNEES
 - 9.3.3.1 - Les ondes sismiques
 - 9.3.3.2 - Data Processing
- 9.3.4 - INTERPRETATION

9.3.1 - EXPLORATION SISMIQUE : but et fondamentaux

Définition de la sismique : La recherche, à visée commerciale, de réserves souterraines de pétrole brut (et dans une moindre mesure de gaz naturel et de minéraux) par l'enregistrement, le traitement et l'interprétation d'ondes de choc artificiellement provoquées.

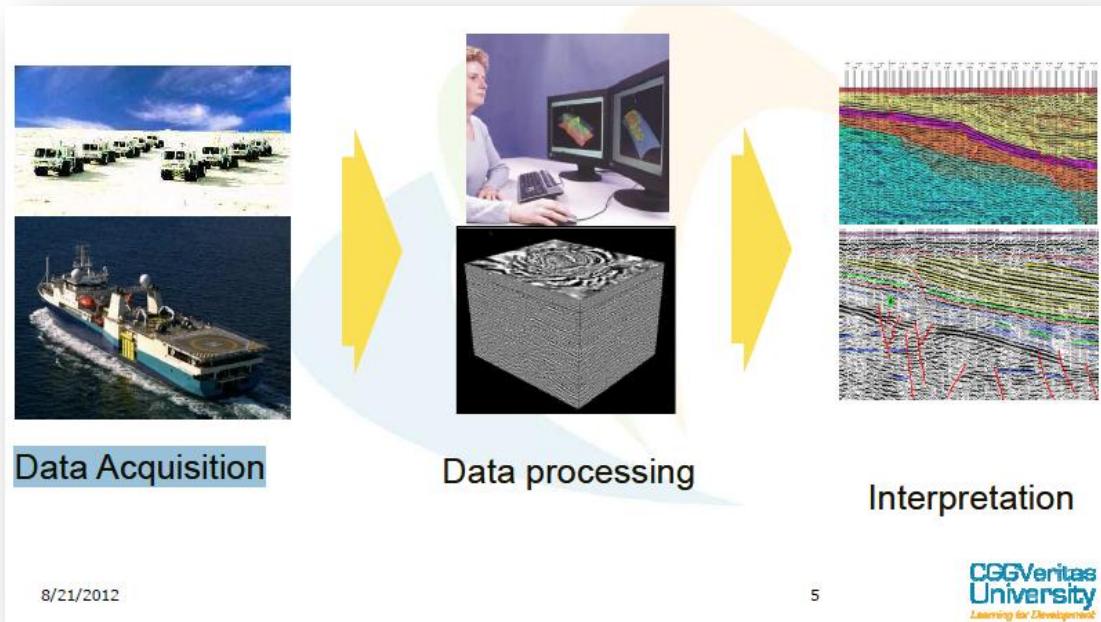
Quel est le but de l'exploration sismique ? C'est de « scanner » la terre pour obtenir une image du sous-sol.

A la demande de leurs clients de l'industrie pétrolière, la CGG explore le sous-sol afin de fournir des images permettant de connaître la nature des couches géologiques et de déterminer la présence de nappes de pétrole.

Comment ?

En résumé, ce métier de CGG consiste à :

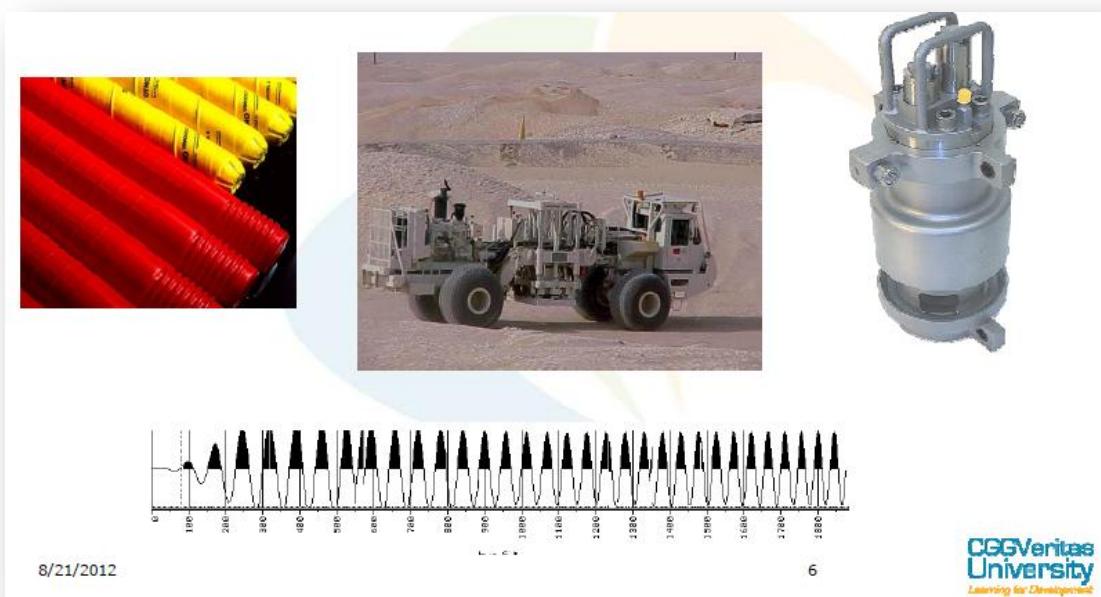
- émettre des signaux, sous forme d'ondes acoustiques, qui vont se propager dans le sous-sol en faisant trembler de la terre,
- effectuer la mesure des signaux à la surface du sol après qu'ils aient été réfléchis par les couches géologiques, c'est l'**ACQUISITION DES DONNEES** ou **Data Acquisition**.
- et de les traiter afin d'obtenir une image la plus précise possible des couches traversées, c'est le **TRAITEMENT DES DONNEES** ou **Data Processing**.



Légende : The seismic method.

9.3.2 – ACQUISITION DES DONNEES

CGG utilise différentes sources d'émission du signal (des ondes acoustiques) selon le site d'exploration : terre ou mer.



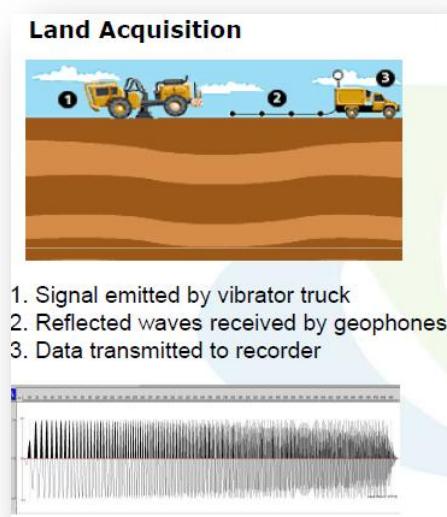
9.3.2.1 - Acquisition terrestre

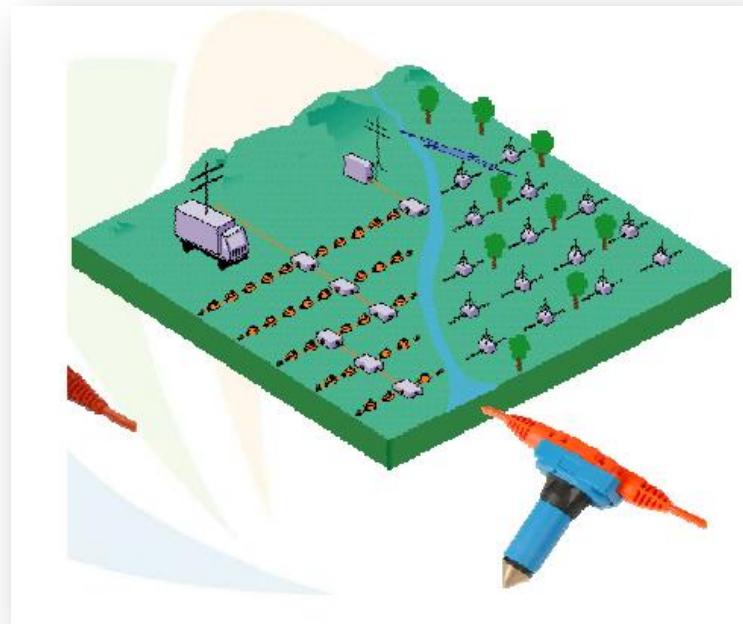
- Sources du signal
 - la dynamite,
 - les camions « Vibroseis », produits par la filiale de CGG SERCEL, font vibrer le sol à l'aide de leur plaque de plus d'1 tonne,



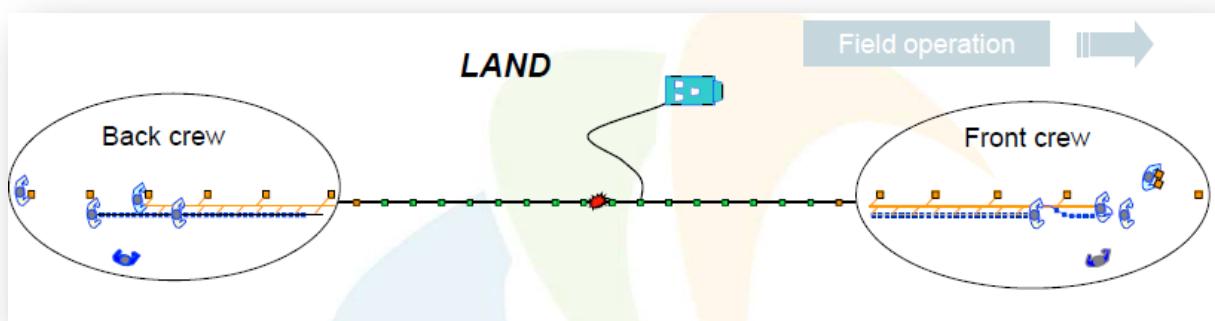
- Acquisition des données

Après l'impact sur la couche géologique, le signal est réfléchi puis remonte à la surface où il est reçu par des capteurs placés dans le sol (géophones) puis enregistré.





Légende : Schéma d'un dispositif terrestre d'acquisition avec installation de lignes de capteurs (géophones) au sol qui sont déplacées au fur et à mesure.

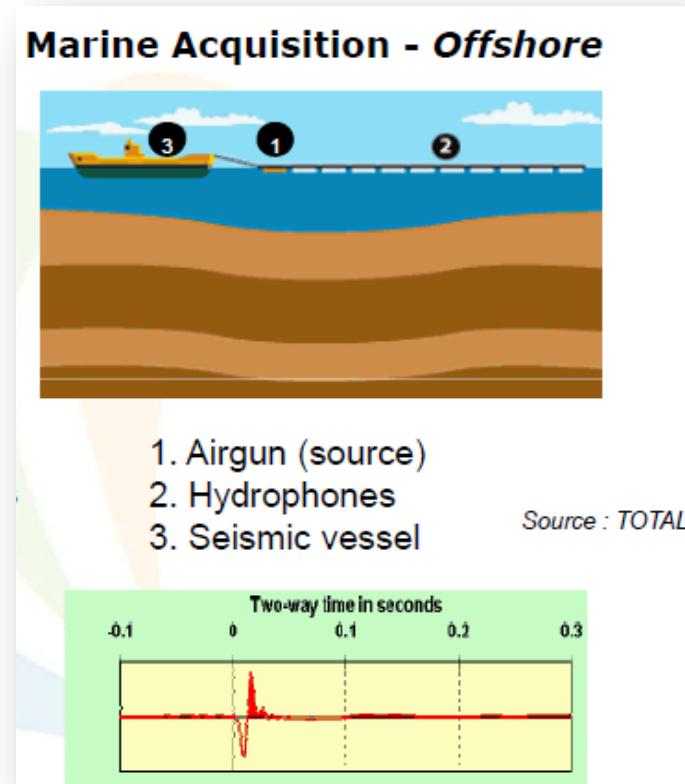


Légende : Les techniciens démontent l'installation à l'arrière du lieu d'acquisition pendant que d'autres installent les lignes de géophones à l'avant.

9.3.2.2 - Acquisition maritime

- Sources du signal
 - les airguns, embarqués sur les bateaux, émettent des explosions d'air comprimé.
- Acquisition des données

Après l'impact sur la couche géologique, le signal est reçu par des capteurs (hydrophones) placés dans les câbles de 10 à 12 kms de longs, tirés par des bateaux sur la zone à explorer, puis enregistré.





Légende : CGG Geo Coral vessel.

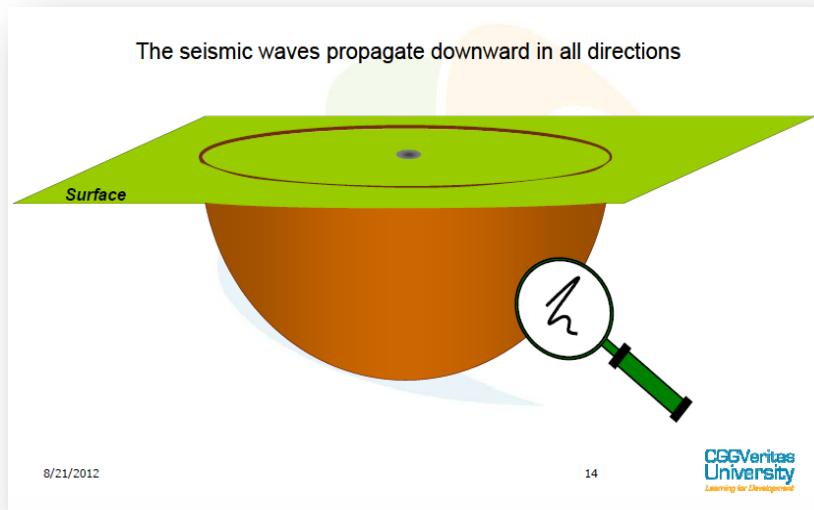
with Courtesy of CGG



Légende : Hydrophones placés dans les câbles tirés par les bateaux (et géophones).

9.3.3 - TRAITEMENT Des données

9.3.3.1 - Les ondes sismiques :



- Vitesse de propagation des ondes différente selon la roche :**

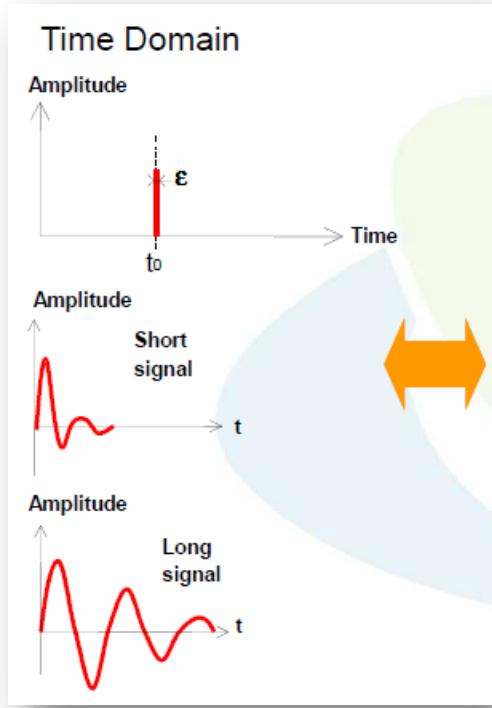
Il existe 2 types d'ondes qui se propagent différemment : P-WAVES et S-WAVES

WAVES PROPAGATION - Rocks Velocities			
	P-WAVES (Vp)	S-WAVES (Vs)	
Sand	1000-1800 m/s	400- 500 m/s	
Chalk	2100- 4200 m/s	1000- 1150 m/s	
Granite	5500- 6000 m/s	2800- 3100 m/s	
Limestone	3400- 6400 m/s	2400- 3100 m/s	

- Propriétés des ondes :

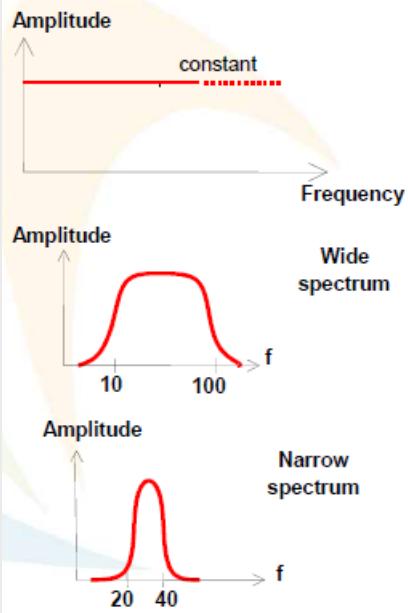
Les domaines TEMPS et FREQUENCE.

Une onde est caractérisée par son amplitude dans le temps ...

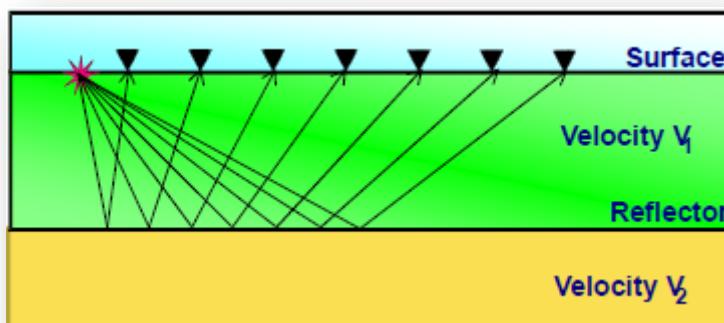
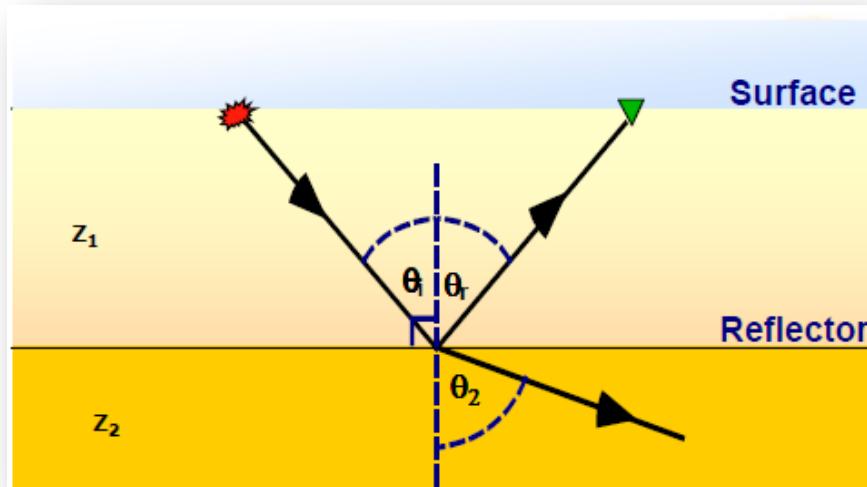


... et par sa fréquence

Frequency Domain



- Réflexion et transmission :



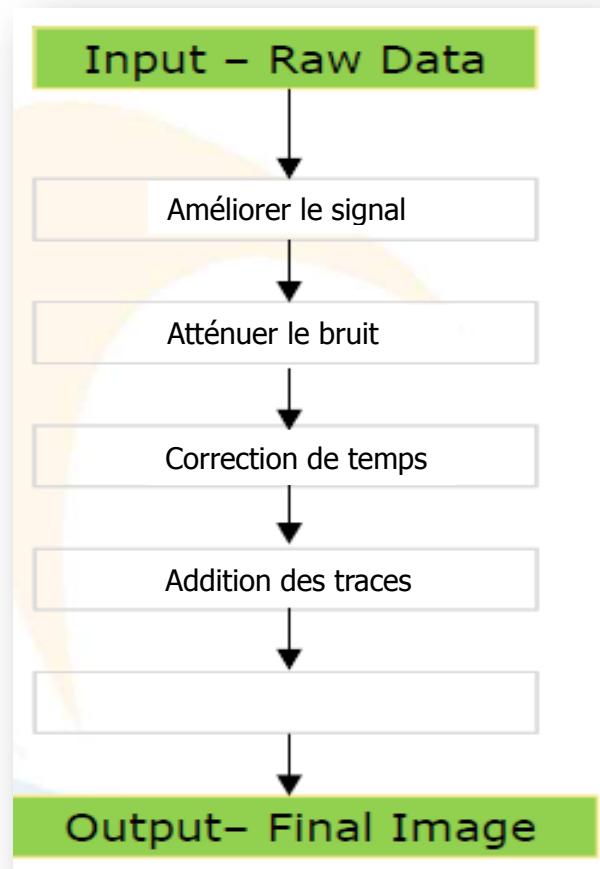
- L'ordre de couverture ou FOLD COVERAGE :

C'est le nombre d'impacts du signal émis sur une couche géologique.
Plus la couverture est grande, plus il est facile d'éliminer le bruit et d'épurer le signal

9.3.3.2 - Data Processing

"The challenge of data processing is to produce the truest possible image of the Earth's subsurface." CGG

Quelques processus de traitement séquentiels pour épurer le signal enregistré et redresser les courbes des ondes :



- **Le « Bruit »**

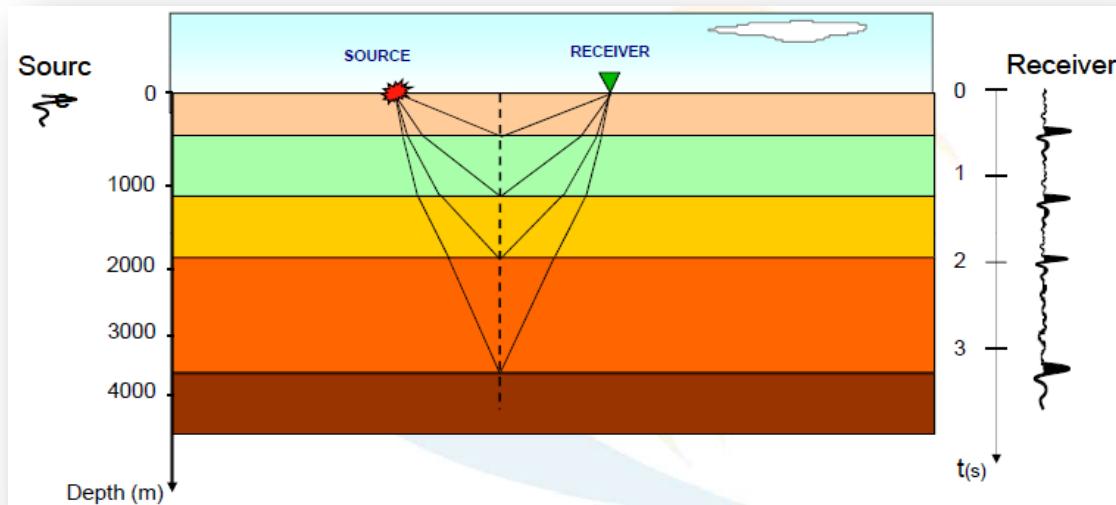
Le traitement du signal consiste, en résumé, à éliminer des signaux enregistrés « le bruit », c'est-à-dire les ondes parasites qui viendraient brouiller l'image finale, pour ne conserver que le signal effectivement réfléchi par la couche géologique.

Exemple de bruit :

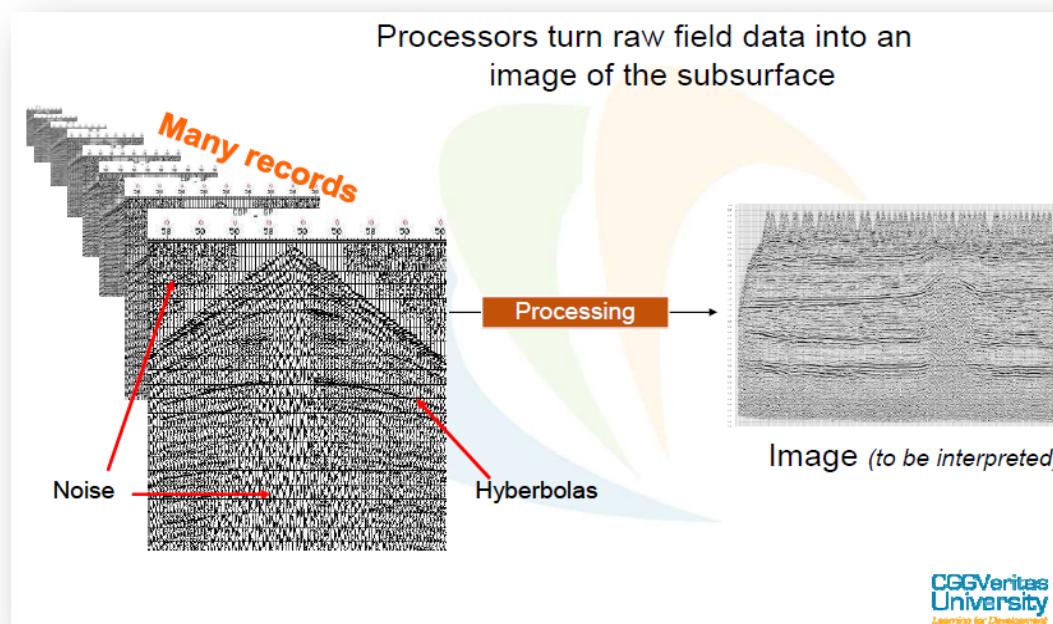
- bruit ambiant capté par les récepteurs,
- certaines ondes générées par la source (dynamite etc ...) mais qui ne correspondent pas à la pure réflexion du signal émis par la couche géologique.
 - En mer, il s'agit notamment des ondes multiples – les rebondissements d'ondes- dues à la surface de l'eau,
 - Sous terre, il y a notamment les Ground Roll (ondes de surface).

- **La trace**

Définition : Somme des signaux enregistrés par un capteur.

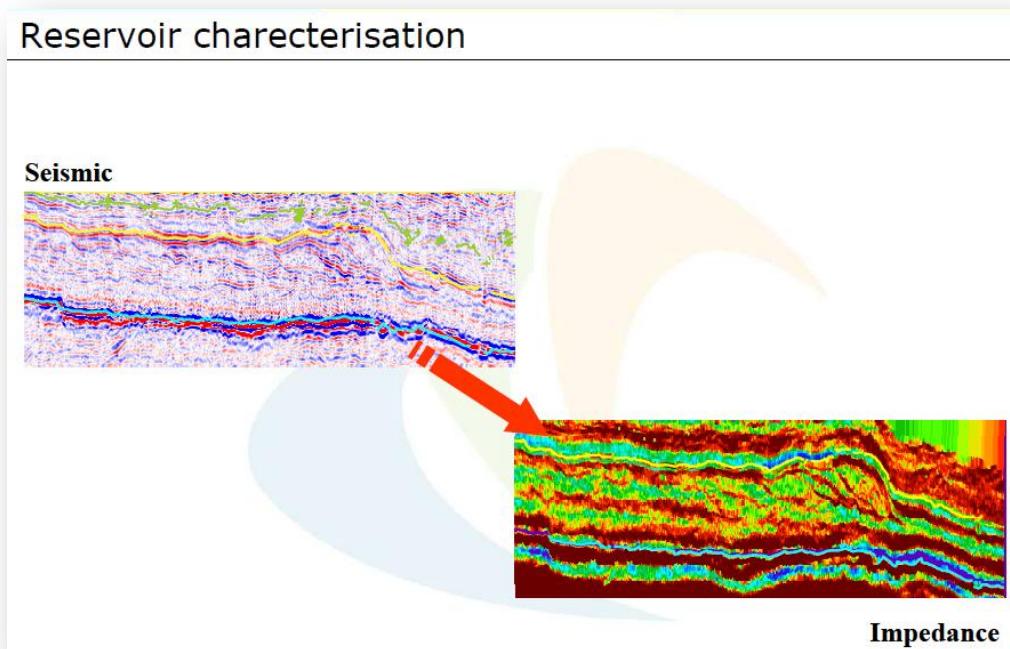


- Les datas enregistrées et traitées



9.3.4 - INTERPRETATION

Les images obtenues après traitement du signal et amélioration de l'image.



9.4 - LE CODE DE L'APPLICATION.

9.4.1 La feuille de style css complète

```
/*
 * ****
 * Classification: Unclassified
 *
 * Copyright Notice:
 *   Copyright (c) 2014 - CGG Interactive application
 *   Unpublished Work. All rights reserved.
 */

/* HTML tag Definition */
body {
    margin: 0 auto;
    width: 98%;
    font-family: "Century Gothic", Arial;
}

header {
    background-color: #f0deef;
    padding-bottom: 4px;
}

footer {
    height: 50px;
    background-color: #d7d7d7;
}

header h1 {
    font-family: 'bpdotsbold';
    color: #575757;
    font-size: 200%;
    margin: 0;
    padding: 3px 0 0 10px;
}

h2 {
    background-color: #178C93;
    margin: 0;
    padding: 15px;
}

h3 {
    background-color: #AAAAAA;
    margin: 0;
    padding: 15px;
    color: #444444;
}

footer nav ul li {
    display: inline-block;
```

```
padding-top: 15px
}

footer nav ul li:after {
    content: " | ";
}

header nav ul li:hover {
    background-color: white;
    cursor: pointer;
}

header nav ul li:hover a {
    color: black;
}

header nav ul li a {
    text-decoration: none;
    color: white;
    font-weight: bold;
}

aside {
    background-color: #CCC;
    float: right;
    margin: 15px 0 15px 0;
    width: 195px;
    padding: 10px;
    text-align: justify;
}

hr {
    display: block;
    clear: both;
    height: 0;
    padding: 0;
    border: 0;
    font-family: arial;
    text-align: center;
    font-size: 20px;
    line-height: 1;
}

hr:after {
    content: "\273D \273D \273D";
    height: 0;
    letter-spacing: 1em;
    color: #aaa;
}

mark {
    background-color: #FF4CD2;
    padding: 5px;
}

input[type="range"] {
    position: relative;
    margin-left: 1em;
```

```
}
```

```
input[type="range"]::after, input[type="range"]::before {  
    position: relative;  
    top: 1em;  
    color: #aaa;  
}
```

```
input[type="range"]::before {  
    left: 0em;  
    content: attr(min);  
}
```

```
input[type="range"]::after {  
    right: 0em;  
    content: attr(max);  
}
```

```
/*from INT samples*/
```

```
.toolbar {  
    position: relative;  
    top: 0px;  
    left: 0px;  
    right: 0px;  
    height: 40px;  
    border: 0px solid black;  
    padding: 0px;  
    width: 300px;  
    /* W3C Markup, IE10 Release Preview */  
    background-image: linear-gradient(to bottom, #fafafa 0%, #c4c4c4 100%);  
}
```

```
.toolbutton {  
    border: 1px solid transparent;  
    width: 32px;  
    height: 32px;  
    bottom: 0px;  
    top: 0px;  
    float: left;  
    margin-right: 2px;  
    margin-left: 2px;  
    margin-top: 2px;  
    border-radius: 3px;  
    -webkit-border-radius: 3px;  
    padding: 2px;  
    line-height: 36px;  
    border: 1px solid transparent;  
    outline-color: #aaa;  
    outline-width: 0px;  
}
```

```
.toolbutton:hover {  
    background-color: #aaa;  
    border: 1px solid transparent;  
}
```

```
.row h2 {
```

```

color: #525252;
font-size: 24px;
font-weight: 300;
letter-spacing: -.01em;
line-height: 1.3;
margin: 15px 0 15px;
}

.row h3 {
color: #525252;
font-size: 20px;
font-weight: 300;
letter-spacing: -.01em;
line-height: 1.3;
margin: 20px 0 20px;
}

.row h3:after {
background: #ddd;
border: 0;
content: "";
display: block;
height: 1px;
overflow: auto;
width: 800px;
}

```

9.4.2 Le code complet de création et d'affichage d'un cross plot

```

/**
 * Created by clellouc on 8/20/15.
 */

define(['helpers', 'geotoolkitWidgets', 'highlightPack'], function (Helpers) {
    hljs.initHighlighting();

    var plotWidth = 800;
    var plotHeight = 800;

    function init(canvasName, title, tool) {
        var crossplotWidget = createWidget(canvasName, title, tool);
        setupTools(crossplotWidget);
        populateData(crossplotWidget);
        updateScrollBars(crossplotWidget);
        return crossplotWidget;
    } // init
}

```

```

function createWidget(canvasName, title, tool) {
    var styleHeader = new geotoolkit.attributes.TextStyle({
        'color': "blue",
        'font': "20px Arial"
    });

    var widget = new geotoolkit.widgets.CrossPlot({
        'bounds': new geotoolkit.util.Rect(0, 0, plotWidth, plotHeight),
        'tools': {
            'horizontalscroll': { 'visible': false },
            'verticalscroll': { 'visible': false }
        },
        'header': {
            'title': {
                'text': title,
                'textstyle': styleHeader
            },
            'annotationsize': 30
        },
        'x': {
            'label': {
                'text': "x axis",
                'textstyle': Helpers.axisColor()['style']
            },
            'annotationsize': 50
        },
        'y': {
            'label': {
                'text': "y axis",
                'textstyle': Helpers.axisColor()['style']
            },
            'annotationsize': 65
        },
        'z': { 'annotationsize': 20 }
    });
    addShapeToCanvas(canvasName, widget);

    if (tool){
        //Connect Widget Tools to Plot (one toolsContainer can be used for several widgets inside the same plot)
        var toolsContainer = new geotoolkit.controls.tools.ToolsContainer(plot);
        toolsContainer.add(widget.getTool());
    }

    return widget;
} // createWidget

```

```

function populateFakeData(crossplotWidget) {

    crossplotWidget.setData({
        'header': {
            'annotationsize': 20
        },
        'x': { 'data' : [1,2,3,4,5,6,7,8,9,9,2,4,4,5,5,5,5,6,6,7,8,9,9,9] },
        'y': { 'data' : [1,3,5,7,9,7,5,3,1,3,6,4,7,2,3,6,8,5,9,4,6,2,4,8] },
        'z': {
            'data': [1,3,1,5,2,4,7,3,6,1,2,4,2,6,3,5,5,1,2,1,4,1,3,6],
            'label': {
                'text': "z color",
                'textstyle': Helpers.axisColor()['style']
            },
            'legendvisible' : true,
            'annotationsize': 90
        },
        'colorprovider' : Helpers.getColorProvider(1,7)
    });
}

// populateFakeData

function setupTools(crossplotWidget) {
    var crosshair = {
        'horizontal': {
            'color': '#525252',
            'width': 1
        },
        'vertical' : {
            'color': "#525252",
            'pattern':geotoolkit.attributes.LineStyle.Patterns.Solid,
            'width': 1
        },
        'north': {
            'visible': true,
            'textstyle': { 'color':'#525252' }
        },
        'east': {
            'visible': true,
            'textstyle': { 'color':'#525252' }
        },
        'west': { 'visible': false },
        'south': { 'visible': false }
    };

    crossplotWidget.setData({
        'tools': {
            'crosshair': crosshair,
            'horizontalscroll': { 'visible': true },
            'verticalscroll': { 'visible': true }
        }
    });
}

// setupTools

```

```
function addShapeToCanvas(canvasName, shape) {
    var canvas = document.getElementById(canvasName);
    // Create a new Plot object from the canvas and group
    plot = new geotoolkit.plot.Plot({
        'canvasElement' : canvas,
        'root' :
            new geotoolkit.scene.Group()
                .setBounds(new geotoolkit.util.Rect(0, 0, plotWidth, plotHeight))
                .setAutoModelLimitsMode(true)
                .addChild(shape),
        'autoUpdate' : true
    });
    plot.setSize(plotWidth, plotHeight);

    return plot;
} // addShapeToCanvas

function updateScrollBars(widget){
    var epsilon = geotoolkit.util.Math.epsilon; //0
    var model = widget.getModel();
    var vml = model.getModelLimits();
    var ml = model.getVisibleModelLimits();

    vBar = Math.abs(vml.getX() - ml.getX()) > epsilon || Math.abs(vml.getHeight() - ml.getHeight()) > epsilon;
    hBar = Math.abs(vml.getY() - ml.getY()) > epsilon || Math.abs(vml.getWidth() - ml.getWidth()) > epsilon;

    widget.setToolsOptions({
        'tools':{
            'verticalscroll': {
                'visible': vBar
            },
            'horizontalscroll': {
                'visible': hBar
            }
        }
    });
} // updateScrollBars
```

```
function zoomIn(widget){  
    widget.zoomIn();  
    updateScrollBars(widget);  
} // zoomIn  
  
function zoomOut(widget){  
    widget.zoomOut();  
    updateScrollBars(widget);  
} // zoomOut  
  
function fitToBounds(widget){  
    widget.fitToBounds();  
    updateScrollBars(widget);  
} // fitToBounds  
  
return {  
    'init': init,  
    'zoomIn' : zoomIn,  
    'zoomOut' : zoomOut,  
    'fitToBounds' : fitToBounds,  
    'addShapeToCanvas' : addShapeToCanvas  
};  
});
```