



Board Agile interactif et collaboratif



Rapport de stage : Silvia SAMBOR PANOR

Concepteur Développeur Informatique

Octobre – Décembre 2016

SOMMAIRE

CHAPITRE 1 - Présentation générale	5
1.1 Introduction	6
1.2 Abstract	7
1.3 Remerciements	8
1.4 Présentation de l'entreprise	9
1.5 Le cadre général du projet	10
1.5.1 Les interlocuteurs et les acteurs du projet	10
1.5.2 Gestion du projet en mode SCRUM	10
CHAPITRE 2 - Cahier des charges	12
2.1 L'objectif principal du projet	13
2.2 Cahier des charges	13
2.2.1 Définition du contenu du Backlog produit : les USER STORIES	14
2.2.2 Priorisation du Backlog produit	16
2.2.3 Définition du Sprint	18
CHAPITRE 3 - Analyse et Conception	19
3.1 La démarche d'analyse avec UML	20
3.2 Le diagramme de cas d'utilisation	22
3.2.1 Fiche de description textuelle d'un cas d'utilisation	23
3.3 Les maquettes	24
3.3.1 Maquette 1 : Page d'accueil	24
3.3.2 Maquette 2 : Page d'inscription	25
3.3.3 Maquette 3 : Page Mot de passe oublié	25
3.3.4 Maquette 4 : Page réinitialiser mot de passe	26
3.3.5 Maquette 5 : Page Product Owner (PO)	26
3.3.6 Maquette 6 : Page profil utilisateur	27
3.3.7 Maquette 7 : Page Product Owner (PO) – créer nouveau Projet	27
3.3.8 Maquette 8 : Page Product Owner (PO) – consultation Projet	28
3.3.9 Maquette 9 : Page Product Owner (PO) – consultation Sprint	28
3.3.10 Maquette 10 : Page Equipe développement (DEV)	29
3.3.11 Maquette 11: Page profil utilisateur	29
3.3.12 Maquette 12 : Page Equipe développement – consultation Sprint	30
3.3.13 Maquette 13 : Page Equipe développement (DEV) – consultation projet	30
3.3.14 Maquette 14 : Page Equipe développement (DEV) – créer nouveau Sprint	31
3.3.15 Maquette 15 : Page Scrum Master (SM)	31
3.3.16 Maquette 16 : Page profil utilisateur	32
3.3.17 Maquette 17 : Page Scrum Master (SM) – consultation Projet	32

3.3.18 Maquette 18 : Page Scrum Master (SM) – consultation Sprint.....	33
3.4 Le diagramme de navigation	34
3.5 Le Diagramme de Séquence Système	36
3.5.1 Diagramme de séquence système : Authentification.....	37
3.5.2 Diagramme de séquence système : Ajout projet.....	38
3.6 Le Diagramme d'activité	39
3.6.1 Diagramme d'activité : Inscription	39
3.6.2 Diagramme d'activité : Statut User Story	40
CHAPITRE 4 - Conception de la Base de Données	41
4.1 La démarche utilisée.....	42
4.1.1 Les règles de gestion.....	43
4.1.2 Dictionnaire de données.....	43
4.2 Les diagrammes de conception de la base de données	45
4.2.1 Le diagramme de classes (DCL).....	45
4.2.2 Modèle conceptuel de données (MCD).....	47
4.2.3 Modèle logique de données (MLD).....	48
4.2.4 Modèle physique de données (MPD).....	49
4.3 Le schéma de la Base de Données	50
4.6 Le code SQL (LDD) de création de la Base de Données	51
CHAPITRE 5 - Conception de l'application	55
5.1 Le diagramme de séquence détaillé	56
5.1.1 Diagramme de séquence détaillé : Consultation d'un projet.....	56
5.1.2 Diagramme de séquence détaillé : Création d'un sprint.....	57
CHAPITRE 6 – Développement	58
6.1 Technologies utilisées.....	59
6.2 Introduction au langage Java EE	59
6.2.1 Environnement requis pour le développement.....	59
6.2.2 Architecture	60
6.2.2.1 Le modèle design pattern ECB (Entity-Boundary-Control)	60
6.2.3 Création et configuration de l'application	61
6.2.3.1 Arborescence de notre application.....	61
6.3 Interfaces web	62
6.3.1 Code HTML.....	62
6.3.1.1 Page Accueil (Welcome).....	62
6.3.1.2 Page Sign In.....	63
6.3.2 Aperçu dans le navigateur.....	65
6.3.2.1 La page Accueil (Welcome).....	65

6.3.2.2 La page Sign In	65
6.4 Code pour les Entities et DAO	66
6.4.1 Code User DAO	66
6.4.2 Code Entities - User POJO (JavaBean)	70
6.4.3 Code Entities - Class UserLoginBean	72
6.4.4 Code Factory de DAOs	74
6.4.5 Connexion vers la Base de données	76
6.5 Contrôleur	76
6.6 Code pour les Tests	77
6.6.1 Code Teste JUnit - User DAO	77
6.6.2 Script pour l'insertion de données de test dans la BD	79
CHAPITRE 7 - Déploiement	80
7.1 Le diagramme de déploiement	81
CHAPITRE 8 - Conclusion et perspectives	82
CHAPITRE 9 - Annexes	83
9.1 Correspondances Projet/Reac	84
9.2 Liste de mots-clés	85
9.3 Les langages	86
9.4 Les outils	87
CHAPITRE 10 - Bibliographie et Webographie	89

CHAPITRE 1 - Présentation générale

1.1 Introduction

Nous sommes dans la tendance du développement agile depuis maintenant plusieurs années, ce qui implique de respecter certaines contraintes de développement et des méthodologies. Malheureusement, certaines d'entre elles ne sont pas ou alors très peu mises en pratique.

L'une des quatre valeurs fondamentales du développement agile est la mise en avant des individus et leurs interactions plus que les processus et les outils. L'optique agile accorde en effet beaucoup plus d'importance à l'équipe. Il est préférable d'avoir une équipe soudée et qui communique, composée de développeurs, plutôt qu'une équipe composée d'experts fonctionnant chacun de manière isolée. La communication est une notion fondamentale.

On peut d'ailleurs trouver dans les douze principes généraux du développement agile une règle qui fait référence à cela : « La méthode la plus efficace pour transmettre l'information est une conversation en face à face. »

Le travail à distance des différentes équipes de nos jours est le problème auquel des nombreuses sociétés sont confrontées pour respecter cette règle susmentionnée. Afin de réduire virtuellement cette distance géographique, la société Sogeti High Tech a décidée d'élaborer une application web Board Agile de type Scrum interactif et collaboratif qui permettra à plusieurs équipes géographiquement distantes d'intervenir sur un board commun.

1.2 Abstract

We are in the trend of agile development for several years now, which implies respecting certain constraints of development and methodologies.

Unfortunately, some of them are not or very little put into practice. One of the four core values of agile development is the emphasis of individuals and their interactions more than processes and tools. Agile optics gives a lot more importance to the team.

It is better to have a tight, communicative team of developers rather than a team of experts working in isolation. Communication is a fundamental concept. One can also find in the twelve general principles of agile development a rule that refers to this: "The most effective method of conveying information is a face-to-face conversation."

The remote work of the various teams today is the problem that many companies face to comply with this rule. In order to reduce this geographical distance virtually, Sogeti High Tech has decided to develop an Interactive and collaborative Scrum-type Board Agile web application that will allow several geographically distant teams to intervene on a common board.

1.3 Remerciements

Je tiens à remercier la société **Sogeti High Tech** pour m'avoir accueilli pour ce stage.

Je voudrais également remercier, Monsieur **BUGUET Pascal**, mon tuteur de stage, pour son soutien et sa disponibilité, ainsi que l'ensemble du personnel enseignant pour nous avoir dispensés d'une formation de grande qualité et m'avoir permis d'acquérir voire d'approfondir mes connaissances.

Enfin, un grand merci à Marc qui m'apporte tous les jours l'équilibre sans lequel je ne pourrais pas continuer à avancer...

1.4 Présentation de l'entreprise

Sogeti est une entreprise de services du numérique (ESN), filiale à 100 % de Capgemini employant 20 000 personnes, dont la moitié en France et l'autre moitié en Inde.

La marque Sogeti - acronyme de « Société pour la gestion de l'entreprise et traitement de l'information » - a été initialement le nom de la maison mère Capgemini dans les années 1970 (Serge Kampf a créé Sogeti avec trois collègues en 1967 à Grenoble). Le groupe se développant, la marque disparaît au profit de Capgemini (et des variantes de ce même nom) avant d'être réutilisée à partir du 1er janvier 2002 pour la filiale des services de proximité du groupe qui regroupe les activités d'assistance technique autrefois intégrées complètement à Capgemini, la maison mère. Sogeti s'est développé principalement via l'OPE (offre publique d'échange) en 2003 sur Transiciel, pour former l'entité Sogeti-Transiciel, devenue Sogeti en 2006.

Sogeti est l'un des leaders des services informatiques et d'ingénierie de proximité, spécialisé dans la gestion des applicatifs et des infrastructures (application and infrastructure management), le conseil en technologies (high-tech engineering) et le Testing. Sogeti aide ses clients à optimiser les performances de leurs systèmes d'information grâce à l'innovation technologique. Présente dans 15 pays avec plus de 200 implantations en Europe, aux États-Unis et en Inde, la société réunit plus de 20 000 professionnels. Sogeti est une filiale à 100 % de Cap Gemini S.A., coté à la Bourse de Paris.

Filiale du groupe Capgemini, Sogeti assiste les entreprises, PME et organismes publics au niveau local contrairement à sa maison mère, tant dans la création et l'optimisation de leurs systèmes d'information que la mise en œuvre de leurs projets industriels de haute technologie. En clair, Capgemini se focalise sur les prestations sur les marchés internationaux, tandis que Sogeti se concentrait sur la prestation de qualité sur le marché national ou local.¹

1.5 Le cadre général du projet

1.5.1 Les interlocuteurs et les acteurs du projet

Réalisatrice de la conception et du développement de l'application: Silvia Sambor Panor
Chef de projet, Scrum Master, et mon tuteur de stage : Flavien Gateuil.
Il est responsable de l'organisation de mon travail et valide mes choix.
Product Owner: Sogeti High Tech

1.5.2 Gestion du projet en mode SCRUM

Les méthodes de développement dites « **méthodes agiles** » (en anglais *Agile Modeling*, noté AG) visent à réduire le cycle de vie du logiciel (donc accélérer son développement) en développant une version minimale, puis en intégrant les fonctionnalités par un processus itératif basé sur une écoute client et des tests tout au long du cycle de développement.

En 2001, 17 personnes mirent ainsi au point le Manifeste Agile dont la traduction est la suivante :

- individus et interactions plutôt que processus et outils
- développement logiciel plutôt que documentation exhaustive
- collaboration avec le client plutôt que négociation contractuelle
- ouverture au changement plutôt que suivi d'un plan rigide

Pour notre projet nous avons choisi de travailler en méthode Scrum:

Scrum est un modèle de développement inventé par Jeff Sutherland en 1993, lorsqu'il travaillait pour la société Easel Corporation, qui s'est associé avec Ken Schwaber pour formaliser Scrum.²

Le terme *Scrum* est emprunté au rugby et signifie mêlée. Ce processus agile s'articule en effet autour d'une équipe soudée, qui cherche à atteindre un but, comme c'est le cas en rugby pour avancer avec le ballon pendant une mêlée.

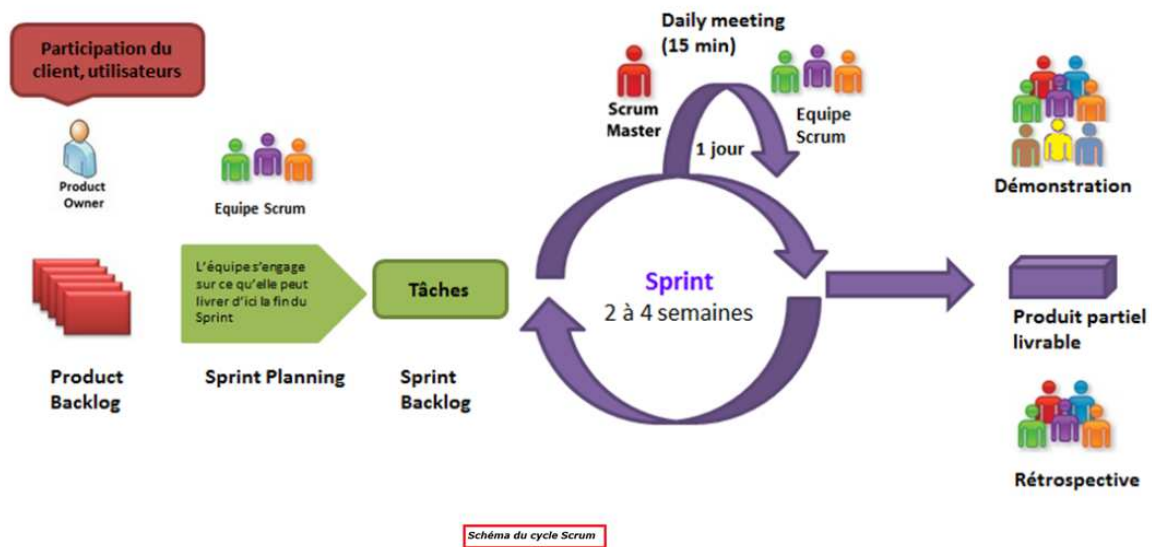
Le principe de base de Scrum est de focaliser l'équipe de façon itérative sur un ensemble de fonctionnalités à réaliser, dans des itérations de 30 jours, appelées *Sprints*.

Chaque Sprint possède un but à atteindre, défini par le directeur de produit (*Product owner*), à partir duquel sont choisies les fonctionnalités à implémenter dans ce Sprint.

Un Sprint aboutit toujours sur la livraison d'un produit partiel fonctionnel.

Pendant ce temps, le *scrummaster* a la charge de réduire au maximum les perturbations extérieures et de résoudre les problèmes non techniques de l'équipe.

Un principe fort en Scrum est la participation active du client pour définir les priorités dans les fonctionnalités du logiciel, et choisir lesquelles seront réalisées dans chaque Sprint. Il peut à tout moment ajouter ou modifier la liste des fonctionnalités à réaliser, mais jamais ce qui est en cours de réalisation pendant un Sprint.³



Mon tuteur de stage, Flavien GATEUIL, est le Product Owner et j'intervenais en tant que team member.

Mon projet s'est déroulé sur 3 sprints :

- 1 sprint de 3 semaine de 26/10/2016 au 09/ 11/ 2016,
- 1 sprint de 3 semaine de 10/11/2016 au 01/ 12/ 2016,
- 1 sprint de 2 semaine de 02/12/2016 au 16/ 12/ 2016.

CHAPITRE 2 - Cahier des charges

2.1 L'objectif principal du projet

Pour diverses raisons, il est de plus en plus fréquent de travailler sur des projets AGILE avec plusieurs équipes déportées. Cependant, la méthodologie SCRUM requiert la présence de toute l'équipe à un seul et même endroit notamment pendant les Daily Scrum Meetings.

Afin de réduire virtuellement cette distance géographique, le sujet du stage constitue à développer un board Agile type Scrum interactif et collaboratif qui permettra à plusieurs équipes géographiquement distantes d'intervenir sur un board commun : modification des tâches, changement de statut, etc. Le board sera projeté sur un mur et une caméra (type webcam, kinect, etc.) détectera les mouvements et actions effectués sur le board : la main jouera le rôle de la souris (clic, déplacement, etc.). Ainsi, toute modification sur un board sera automatiquement répercutée sur les boards des autres équipes SCRUM.

Il devra également être possible de « plugger » le système à un autre outil Agile existant sur le marché (Taiga, JIRA, etc.) via un système de plugin.

Le sujet va être découpé en deux missions distinctes :

1. Application web pure de gestion du board.
2. Système de réalité virtuelle pour la détection des mouvements et interaction avec l'application web.

Après ces deux mois de stage j'ai pu traiter la première partie de ce projet, l'Application web pure de gestion du board, que je vais présenter dans ce rapport.

2.2 Cahier des charges

Le cahier des charges fonctionnel est le document par lequel le demandeur (client) exprime ses besoins en termes de fonctions de service et de contraintes. En méthode Agile le cahier des charges représente les **users stories**.

Une **User story** est la description d'une fonctionnalité du point de vue utilisateur.

Elle prend le formalisme "En tant que... Je veux... afin de...". Une user story peut être divisée en tâches si elle est complexe.

2.2.1 Définition du contenu du Backlog produit : les USER STORIES

Le **Backlog** est l'ensemble des fonctionnalités du produit que l'on veut développer.

Nom du projet : **Board Agile Interactif et Collaboratif**

Méthodologie de gestion de projet : SCRUM

Mon tuteur de stage dirige les réunions.

L'outil Taïga 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'un brainstorming avec le directeur de l'agence et les chefs de projets dont mon tuteur de stage qui a incarné à ce titre le rôle du Product Owner d'un point de vue de la méthode Scrum.

Grace à cette application, les utilisateurs pourront :

- créer, modifier ou supprimer des projets
- consulter l'état des projets en fonction de statut des users stories
- changer l'état d'avancement des users stories dans les sprints

Ces différents points m'ont permis de rédiger le backlog produit comprenant une liste de users stories fonctionnelles.

Board Agile **BACKLOG** pour l'application web:

USER STORY
(1) - En tant qu'utilisateur, je veux me connecter afin d'interagir avec l'application.
(2) - En tant qu'utilisateur, je veux pouvoir créer un compte afin de pouvoir me connecter à l'application.
(3) - En tant qu'utilisateur, je veux pouvoir accéder à la page d'inscription afin de m'inscrire.
(4) - En tant qu'utilisateur, je veux pouvoir accéder à une interface avec tous mes projets afin de pouvoir en consulter un.
(5) - En tant qu'utilisateur, je veux pouvoir créer des "user stories" dans backlog afin de décrire les besoins du projet.
(6) - En tant qu'utilisateur, je veux pouvoir créer un projet, afin de pouvoir le consulter.
(7) - En tant qu'utilisateur, je veux pouvoir créer des "sprint" afin d'effectuer la planification des itérations du projet.
(8) - En tant qu'utilisateur, je veux pouvoir faire des modifications dans le projet, afin de pouvoir faire des mises à jour.
(9) - En tant qu'utilisateur, je veux pouvoir supprimer un projet afin d'effacer le projet de l'application.
(10) - En tant qu'utilisateur, je veux pouvoir choisir le rôle de l'utilisateur, afin de pouvoir m'inscrire.
(11) - En tant qu'utilisateur, je veux pouvoir faire des modifications dans le user profile, afin de pouvoir faire des mises à jour.
(12) - En tant qu'utilisateur, je veux pouvoir accéder à tous les sprints afin de pouvoir en consulter un.
(13) - En tant qu'utilisateur, je veux pouvoir déplacer une user story, afin de modifier son statut.

- (14) - En tant qu'utilisateur, je veux pouvoir consulter un projet, afin de voir les users stories qui me sont affectées.
- (15) - En tant qu'utilisateur, je veux pouvoir accéder à une interface, afin de réinitialiser le mot de passe.
- (16) - En tant qu'utilisateur, je veux pouvoir supprimer un user story afin d'effacer le projet de l'application.
- (17) - En tant qu'utilisateur, je veux pouvoir déplacer une user story dans un nouveau Sprint, afin de modifier son statut.
- (18) - En tant qu'utilisateur, je veux pouvoir déplacer une user story dans le Backlog, afin de modifier son statut.
- (19) - En tant qu'utilisateur, je veux pouvoir ajouter un avatar sur le user story, afin d'assigner le user story à un développeur.

Backlog extrait de Taïga :

The screenshot displays the Taiga project management interface. The main section is the 'TEAMBOARD BACKLOG' which shows a progress bar at 57% and a list of user stories. The right sidebar shows the 'SPRINTS' section for 'Sprint #1' with a list of tasks and their points.

TEAMBOARD BACKLOG

57% 21 project points 153 defined points 12 closed points 0 points / sprint

SHOW FILTERS SHOW TAGS + ADD A NEW USER STORY

Votes	User Stories	Status	Points
Project Scope [Doomline]			
▲ 0	#1 Liste des projets	New	2
▲ 0	#9 Création d'un projet	New	3
▲ 0	#11 Modification d'un projet	New	3
▲ 0	#12 Suppression d'un projet	New	2
▲ 0	#13 Thématiser l'application	New	2
▲ 0	#16 Type de projet	New	3
▲ 0	#7 Créer des "user story"	New	3
▲ 0	#8 Ajouter des "critères d'acceptations"	New	1
▲ 0	#10 Créer des "sprint"	New	3
▲ 0	#14 Créer des tâches	New	3
▲ 0	#18 Système d'estimation (les story point)	New	1
▲ 0	#20 Priorités des tâches	New	2
▲ 0	#15 Affectation de tâches	New	2
▲ 0	#23 Supprimer des "user story"	New	2
▲ 0	#22 Modifier des "user story"	New	3
▲ 0	#26 Système de filtre	New	3
▲ 0	#19 Manipuler le board avec la main	New	10
▲ 0	#17 Fonctionnalité à un mouvement	New	8
▲ 0	#21 Pincer le post-it (user story)	New	5
▲ 0	#27 Alerte par email	New	2
▲ 0	#28 Alerte lors des réunions (time boxing)	New	2
▲ 0	#43 Humeur des tâches	New	3

SPRINTS

▼ Sprint #1 12 closed 23 total

26/10/2016 - 09/ 11/ 2016

#103 Restructurer projet front-end	2
#72 Layout de l'application	1
#71 Page d'accueil de l'application	1
#95 Mise en place de Jersey	2
#87 Structure des fichiers SQL	3
#84 Framework de persistance	5
#3 Création d'un nouvel utilisateur	3
#92 Système de gestion des tokens JWT	3
#2 Système d'authentification	2
#73 Système d'autorisations globales	1

SPRINT TASKBOARD

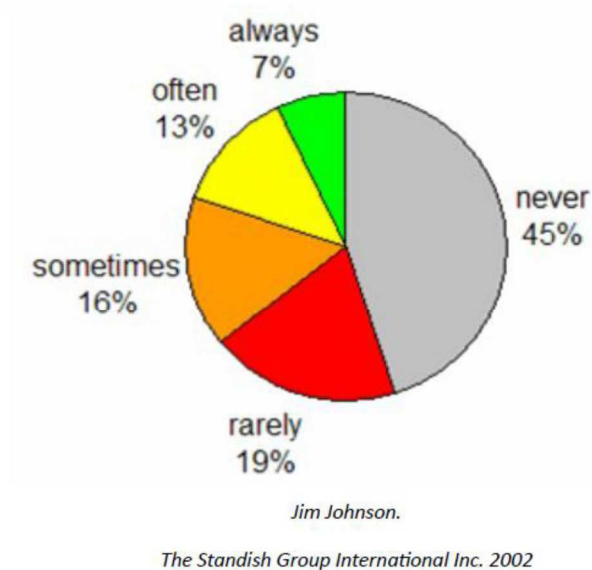
Show closed sprints

2.2.2 Priorisation du 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.



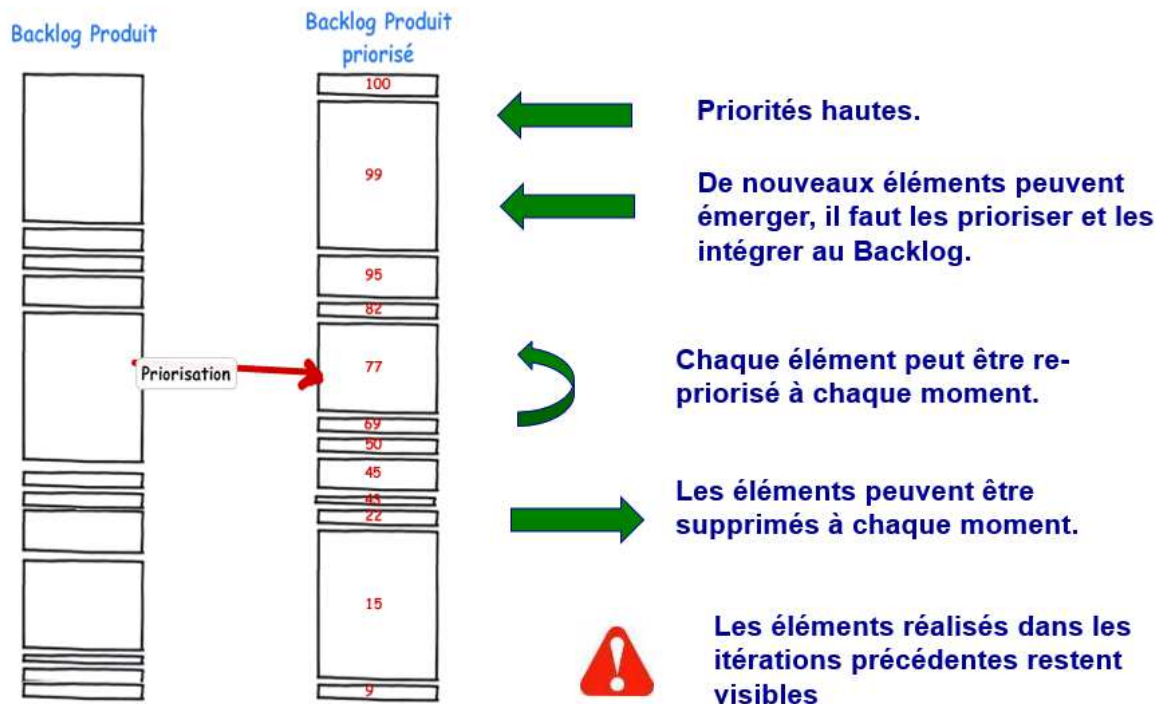
- ✓ L'objectif de la gestion de projet informatique en mode Scrum est donc de livrer en priorité des fonctionnalités opérationnelles utiles à l'utilisateur.
- ✓ Même si le projet devait être interrompu faute de budget, le client est donc détenteur d'une application éventuellement partielle mais opérationnelle 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, au cours d'une séance de planning poker, et décomposées en fonctionnalités plus fines, en tâches les plus simples possibles à implémenter.

Planning poker est la séance d'estimation menée par l'équipe de développement qui évaluent l'ensemble de l'effort nécessaire pour traiter les user stories du *backlog*.

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.

Schéma Priorisation du Backlog



2.2.3 Définition du Sprint

Board Agile - 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 Scrum Board les projets créés dans l'espace utilisateur, la condition préalable est que la base de données qui va stocker les données soit créée. C'est donc une des premières fonctions à implémenter incluse dans le sprint 1.

Backlog produit de Board Agile priorisé

User stories	Priorité	Story points
<i>Conception de l'application</i>	Haute	6
<i>Création de la BD</i>	Haute	6

Story points est un outil d'estimation de l'effort (en jour/homme) nécessaire pour développer des fonctionnalités.

Ce backlog du sprint 1 comprend donc:

- ✓ des stories portant sur la définition et la préparation du projet lui-même :
 - définir la conception de l'application Web à développer
- ✓ 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 et Conception

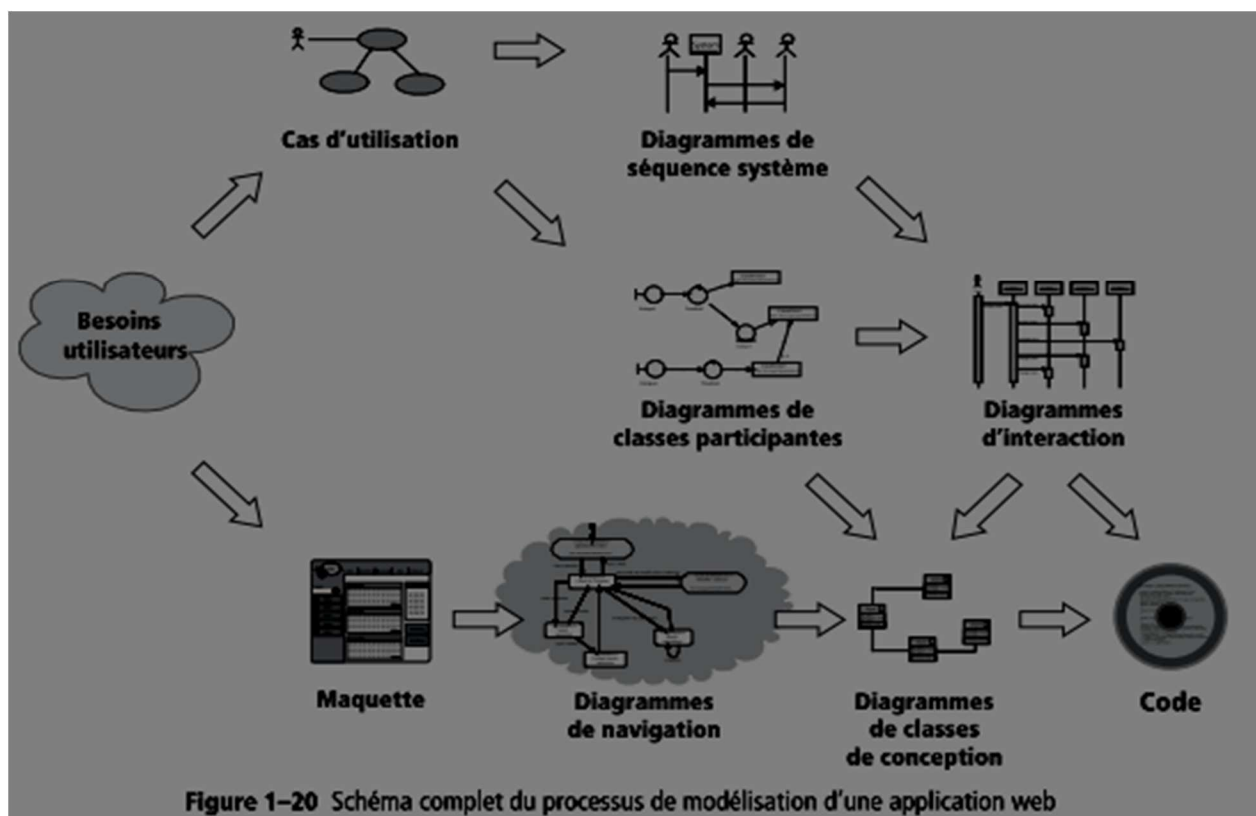
3.1 La démarche d'analyse avec UML

Après la rédaction du backlog (user stories) produit et une analyse du besoin des utilisateurs du projet en question, je suis passée à la partie modélisation et représentation graphique (UML) de notre futur système en suivant la démarche préconisée par Pascal ROQUES pour la modélisation d'une application web, que j'ai enrichie avec le cours de M. Pascal Buguet, à laquelle j'ai fait quelques adaptations en fonction des besoins de notre projet.

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.



Les étapes de l'analyse :

Point de départ	Point d'arrivée	L'étape suivante
	Besoin des utilisateurs : user stories du backlog produit	Diagramme de cas d'utilisation (DCU)
Besoin des utilisateurs : user stories du backlog produit	DCU	Fiche textuelle d'un CU, Maquettes, DSS
DCU	Fiche textuelle d'un CU	DSS
DCU	Maquettes	DNAV
DCU, Fiche textuelle d'un CU	Diagramme de séquence système	DSD, DAC
Maquettes (IHM)	Diagramme de navigation	DCL
DCU, DSS, Scénarios	Diagramme d'activité	Codage de la dynamique des interfaces (IHM)

3.2 Le diagramme de cas d'utilisation

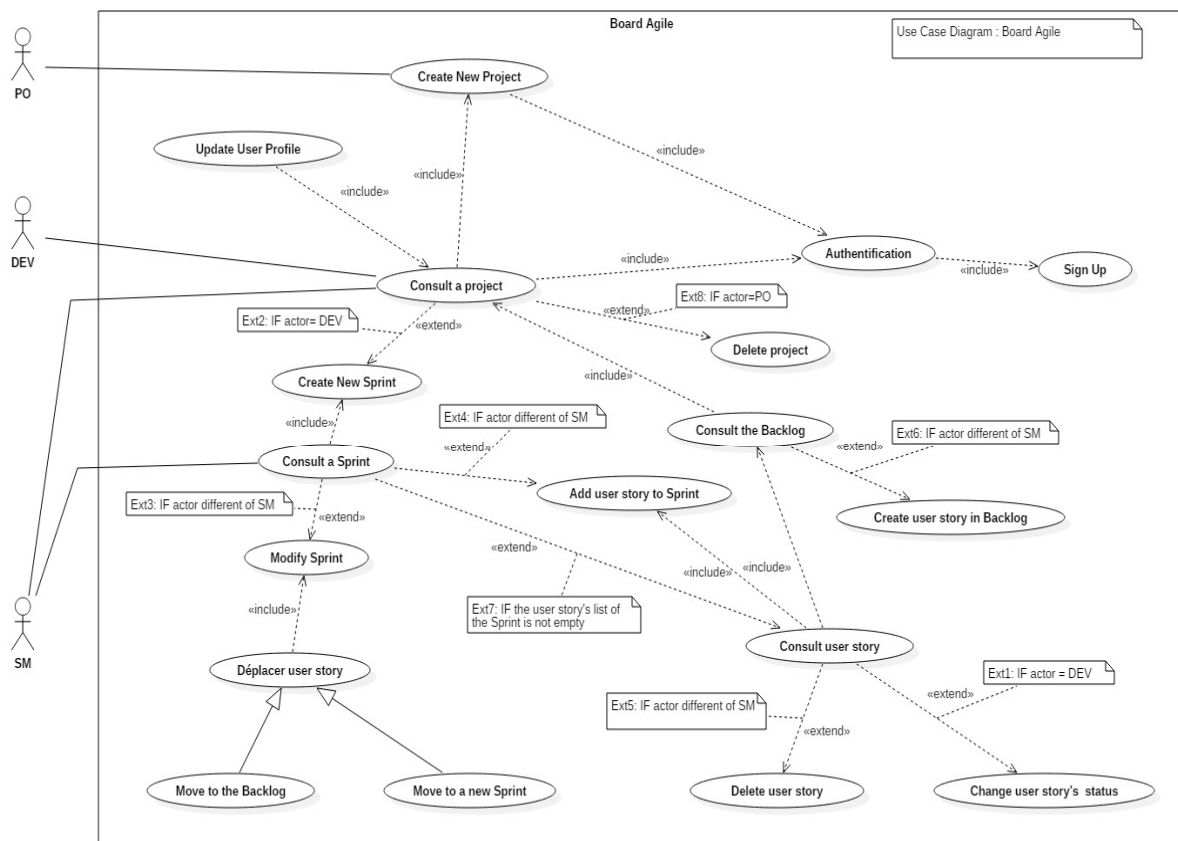
Un diagramme de cas d'utilisation (DCU) est un ensemble de cas d'utilisation (CU). Représente les fonctionnalités (CU) nécessaires aux utilisateurs.

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

Un cas d'utilisation (« use case ») représente un ensemble de séquences d'actions qui sont réalisées par le système et qui produisent un résultat observable intéressant pour un acteur particulier.

Application :

Dans le projet Board Agile, les acteurs principaux sont : le Product Owner (PO), Scrum Development Team (DEV) et le Scrum Master (SM), qui déclenchent les cas d'utilisation de l'application.



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

La fiche de description textuelle d'un cas d'utilisation n'est pas normalisée par UML et par conséquent j'ai élaboré des descriptions textuelles d'une façon qui soit le plus adaptée à notre projet.

Etapes	Description
Identification du CU	<p>Titre : Créer un projet</p> <p>Résumé : sauvegarder dans la BD des données</p> <p>Acteur : utilisateur (PO, SM, DEV)</p> <p>Auteur : S. S. Panor</p>
Pré-conditions	<p>La connexion à la base de données est disponible</p> <p>L'utilisateur doit être connecté.</p>
Scénario nominal	<p>1. Créer un produit</p> <p>1.1 L'utilisateur choisit de créer un produit</p> <p>1.2 Le système affiche un formulaire de saisie des données</p> <p>1.3 L'utilisateur saisit les informations nécessaires et valide le formulaire</p> <p>1.4 Le système vérifie la présence des données obligatoires</p> <p>1.5 Le système enregistre la saisie validée dans la BD</p>
Scénarii alternatifs d'erreurs	<p>1.4 Le système vérifie la présence des données obligatoires : Un des champs obligatoire au minimum n'est pas saisi ou ne correspond pas au format souhaité</p> <p>1.4.1 Un message est affiché « merci de saisir tous les champs obligatoires /merci de vérifier le format de votre saisie.</p> <p>1.4.2 Le scénario recommence à partir du cas d'utilisation 1.2</p>

Le scénario nominal est celui qui satisfait les objectifs des acteurs par le chemin le plus direct.

3.3 Les maquettes

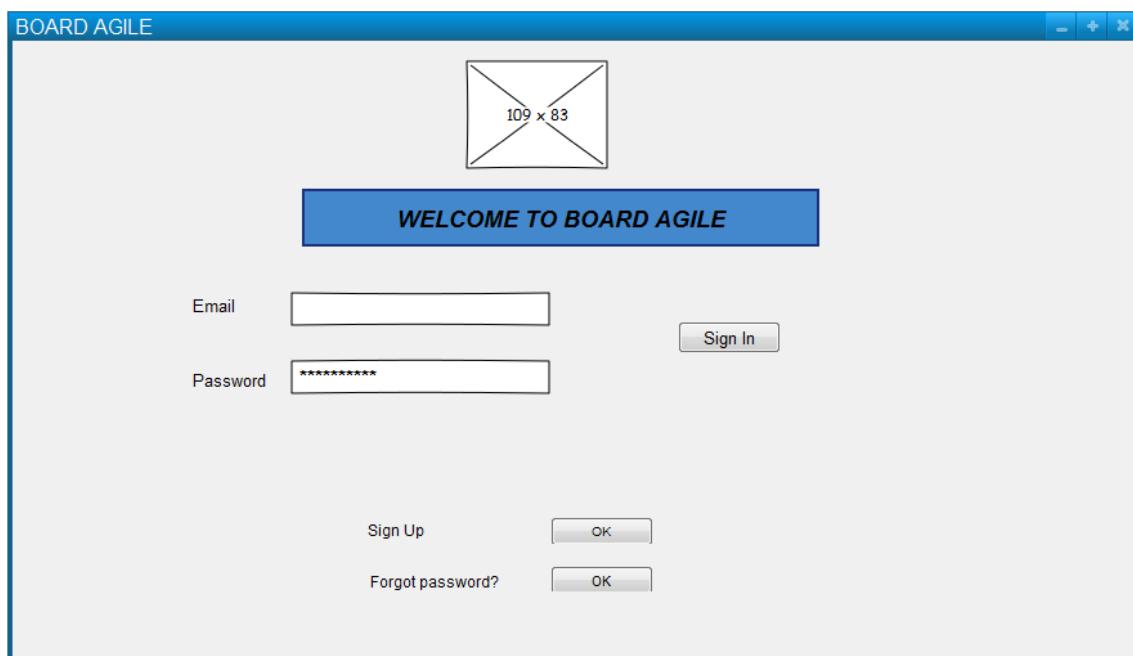
Une maquette d'IHM (Interface Homme-Machine) est un produit jetable donnant aux utilisateurs une vue concrète mais non définitive de la future interface de l'application.³

Les maquettes sont réalisées avec l'outil de maquettage Pencil Evolus.

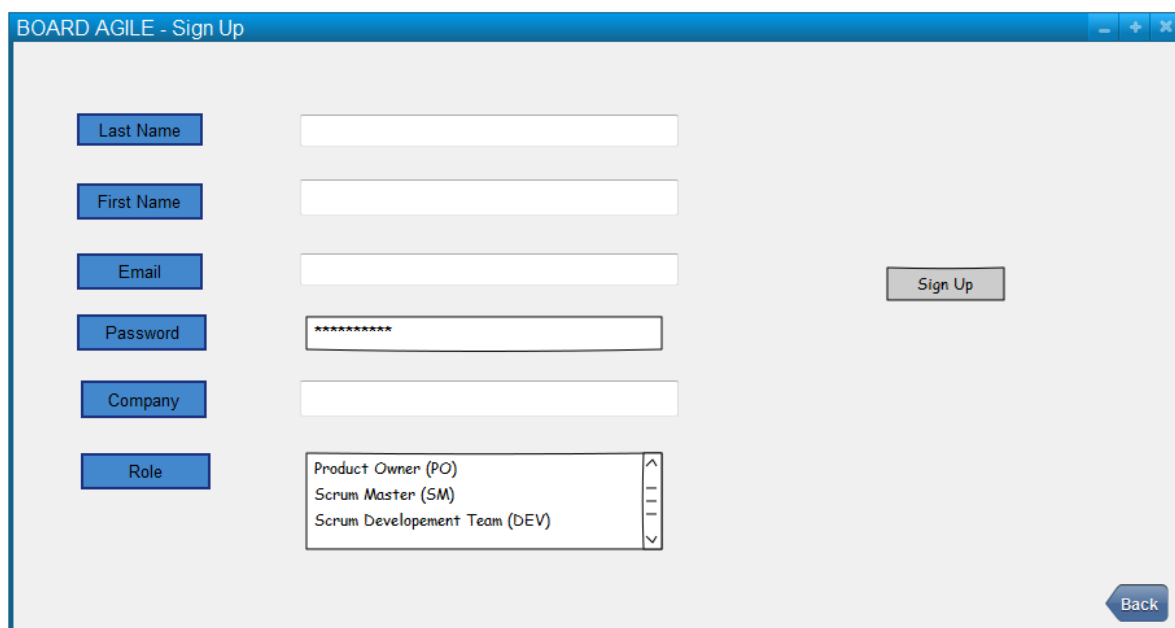
La liste des cas d'utilisation m'a permis de réaliser ces maquettes.

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).

3.3.1 Maquette 1 : Page d'accueil



3.3.2 Maquette 2 : Page d'inscription



BOARD AGILE - Sign Up

Last Name

First Name

Email

Password

Company

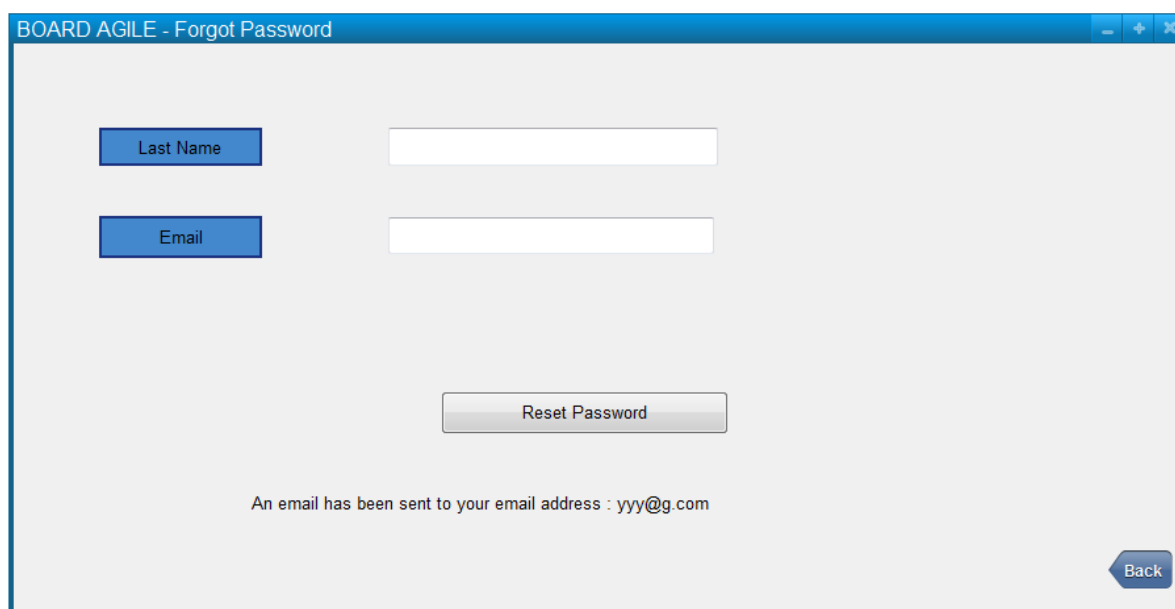
Role

Product Owner (PO)
Scrum Master (SM)
Scrum Development Team (DEV)

Sign Up

Back

3.3.3 Maquette 3 : Page Mot de passe oublié



BOARD AGILE - Forgot Password

Last Name

Email

Reset Password

An email has been sent to your email address : yyy@g.com

Back

3.3.4 Maquette 4 : Page réinitialiser mot de passe

BOARD AGILE - Reset Password

New Password

Confirm New Password

New Password

Your password has been successfully updated.

Back

3.3.5 Maquette 5 : Page Product Owner (PO)

BOARD AGILE - PO

PO

Bonjour Z

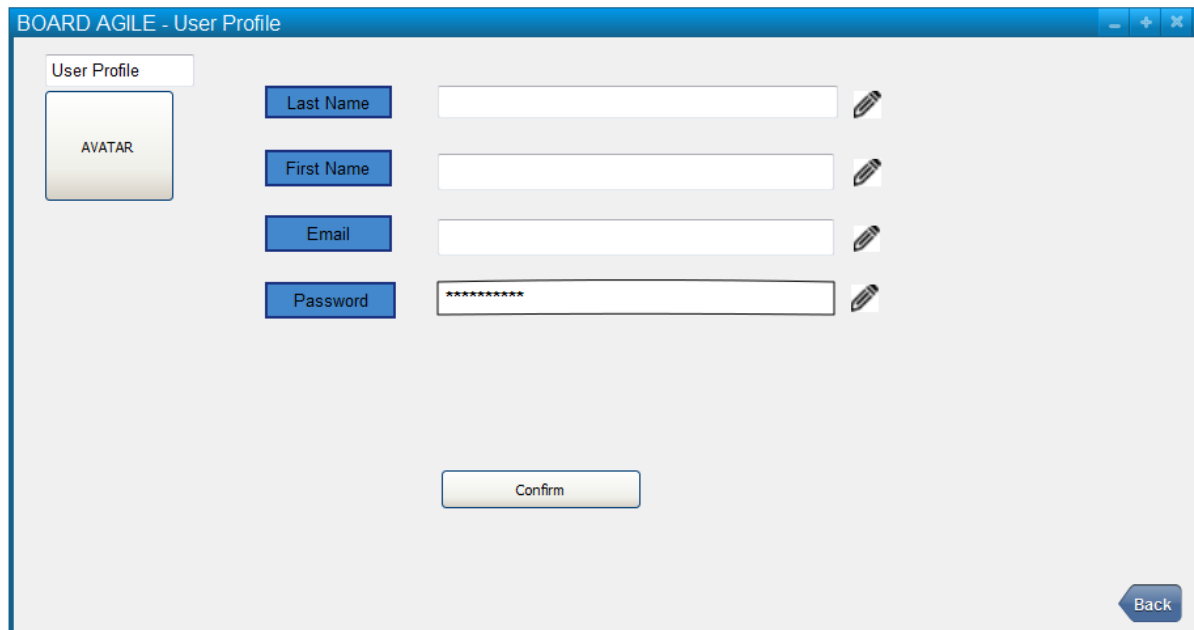
User Profile

Create New Project

Project 1	Board Agile Interactif et collaboratif	Sprint 1
Project 2	Aaaaa	Backlog
Project 3	Bbbbb	Backlog

Back

3.3.6 Maquette 6 : Page profil utilisateur



BOARD AGILE - User Profile

User Profile

AVATAR

Last Name

First Name

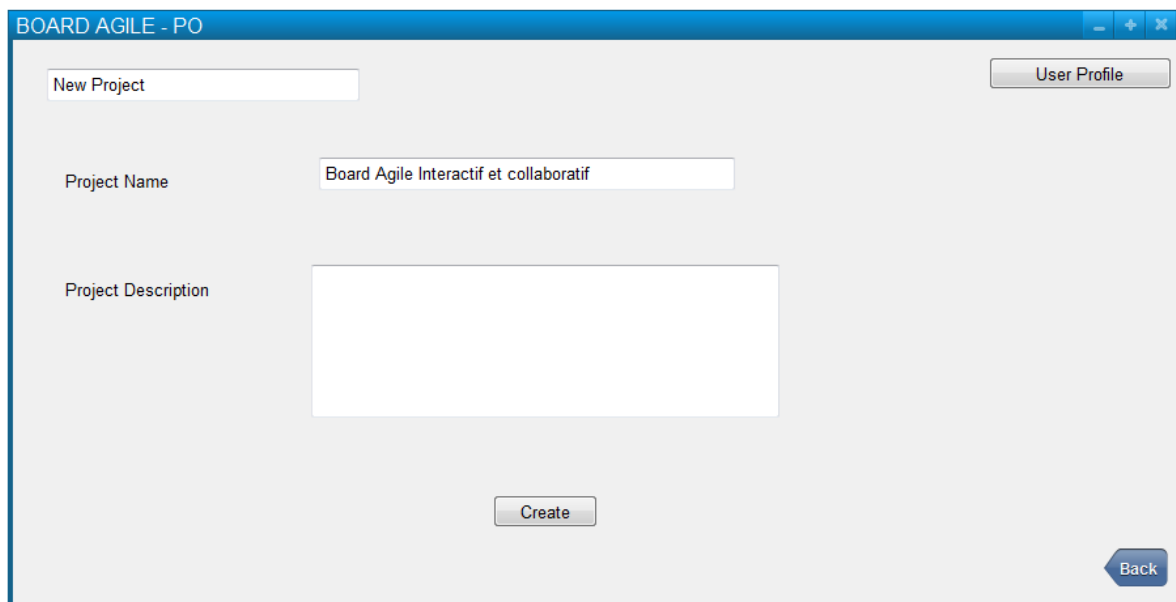
Email

Password

Confirm

Back

3.3.7 Maquette 7 : Page Product Owner (PO) – créer nouveau Projet



BOARD AGILE - PO

New Project

User Profile

Project Name

Project Description

Create

Back

3.3.8 Maquette 8 : Page Product Owner (PO) – consultation Projet

3.3.9 Maquette 9 : Page Product Owner (PO) – consultation Sprint

3.3.10 Maquette 10 : Page Equipe développement (DEV)

BOARD AGILE - DEV

DEV Bonjour Y User Profile

Sprint	Project	TO DO	ON GOING	BLOCKED	IN TEST	DONE
Sprint 1	Project 1	3	1			1
Sprint 3	Project 2	3	2			2
Sprint 2	Project 3	2		1	1	

Back

3.3.11 Maquette 11: Page profil utilisateur

BOARD AGILE - User Profile

User Profile

AVATAR

Last Name

First Name

Email

Password

Confirm

Back

3.3.12 Maquette 12 : Page Equipe développement – consultation Sprint

BOARD AGILE - DEV

DEV Sprint 1 Board Agile Interactif et Collaboratif User Profile

TO DO	ON GOING	BLOCKED	IN TEST	DONE
Red Card	Orange Card			Green Card
Red Card	Orange Card			
Red Card				

Back

3.3.13 Maquette 13 : Page Equipe développement (DEV) – consultation projet

BOARD AGILE - DEV Project 1

Board Agile User Profile

Backlog

US 1

US 2

Sprint

Sprint 1

US 1

Sprint 2

Create New Sprint

Back

3.3.14 Maquette 14 : Page Equipe développement (DEV) – créer nouveau Sprint

BOARD AGILE - Dev Create new Sprint

Board Agile

User Profile

New Sprint

Sprint Number

Started date

End Date

Create Sprint

Back

3.3.15 Maquette 15 : Page Scrum Master (SM)

BOARD AGILE - SM

SM

Bonjour Q

User Profile

Project 1

Board Agile Interactif et collaboratif

Project 2

Aaaaa

Project 3

Bbbbb

Project 4

Project 5

Sprint 1

Backlog

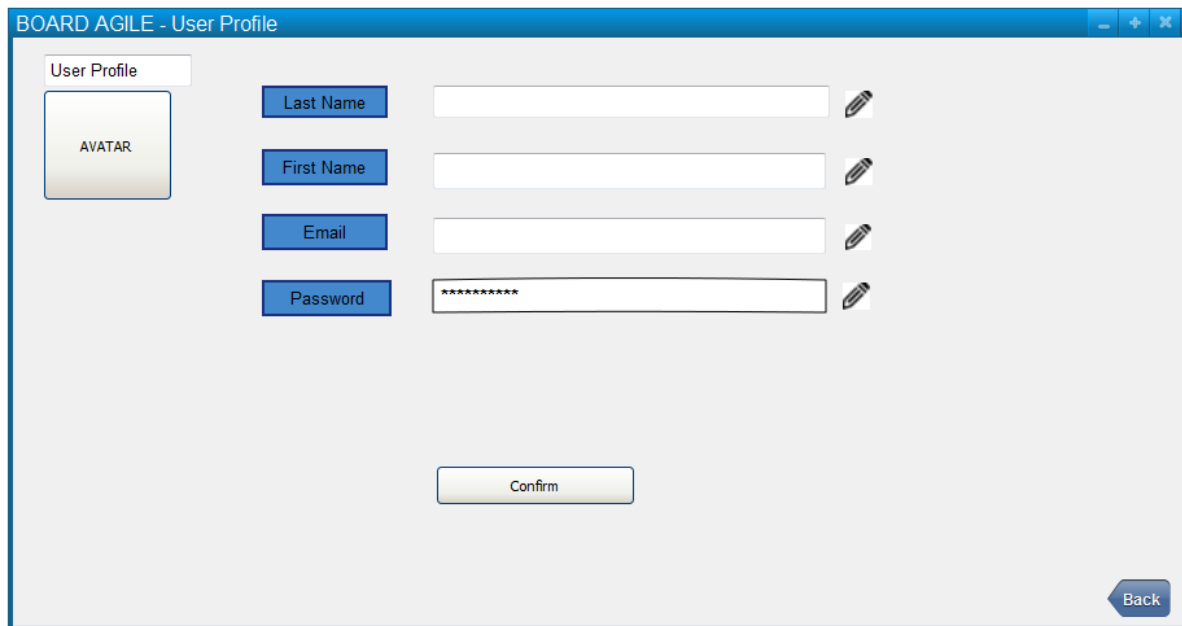
Backlog

Backlog

Backlog

Back

3.3.16 Maquette 16 : Page profil utilisateur



BOARD AGILE - User Profile

User Profile

AVATAR

Last Name

First Name

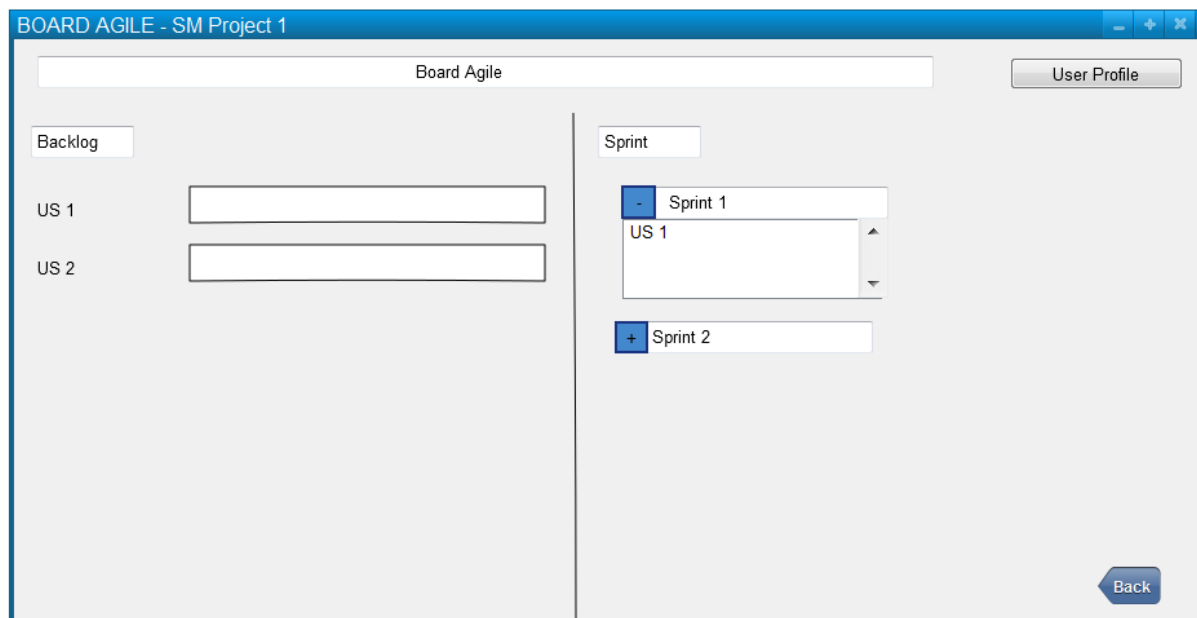
Email

Password

Confirm

Back

3.3.17 Maquette 17 : Page Scrum Master (SM) – consultation Projet



BOARD AGILE - SM Project 1

Board Agile

User Profile

Backlog

US 1

US 2

Sprint

- Sprint 1

US 1

+ Sprint 2

Back

3.3.18 Maquette 18 : Page Scrum Master (SM) – consultation Sprint

BOARD AGILE - SM Sprint1

Sprint 1 Board Agile Interactif et Collaboratif User Profile

TO DO	ON GOING	BLOCKED	IN TEST	DONE
33 x 28	33 x 28			33 x 28
33 x 28	32 x 31			

Back

3.4 Le diagramme de navigation

Le Diagramme de Navigation sert à représenter le cheminement, la cinématique de l'application ou couche Contrôle (Control), entre les différents écrans (IHM) de l'application ou couche Dialogue (Boundary).

Une **navigation** a un **début** et éventuellement une **fin**.

Ils peuvent être représentés par des Diagrammes d'États-Transitions.

Un **état** correspond à une maquette, à une classe dialogue : une page, une frame ou élément de page, une boîte de dialogue, une exception ou bien un connecteur (un sous-système).

Les **transitions**, passage d'un écran à un autre, correspondent aux actions générées par les interactions – des événements - de l'utilisateur via des boutons, des liens etc.

Les diagrammes de navigation sont dérivés des maquettes (copies d'écrans). un écran = un état.

Une maquette correspond le plus souvent à un Cas d'Utilisation.

Les écrans – états – sont représentés par des rectangles aux coins arrondis.

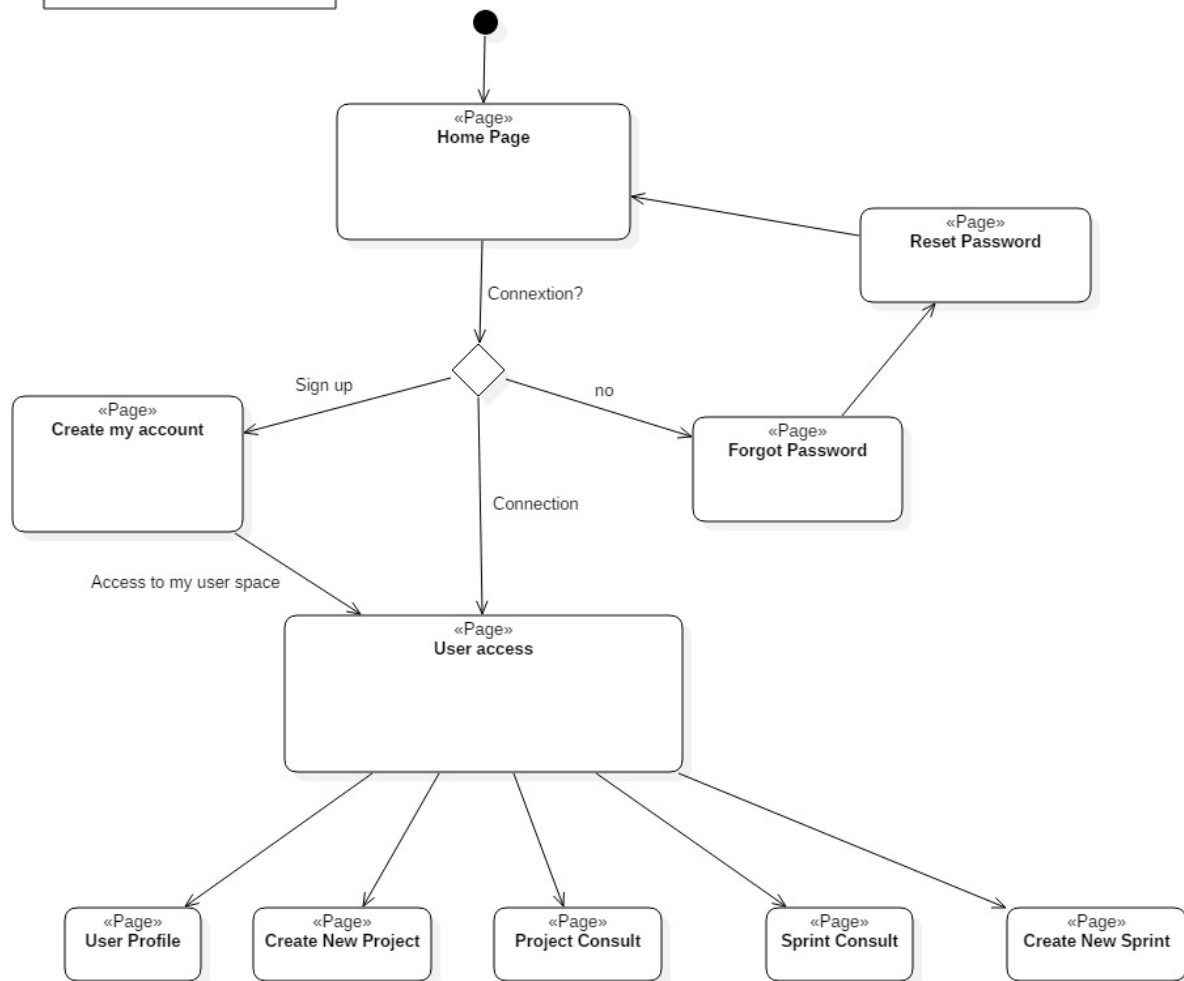


un rond plein représente un pseudo-état initial.



un rond plein encerclé représente un pseudo-état final.

Navigation Diagramm : Board Agile



3.5 Le Diagramme de Séquence Système

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 (les objets) à l'horizontal.

Un diagramme de séquence correspond à un seul scénario.

Les DSEQ sont utilisés différemment, de façon plus ou moins détaillée, au cours du cycle de vie d'un projet.

Le DSEQ utilise quatre concepts clés : l'**acteur**, la **classe** ou l'**objet**, l'**activation**, le **message**.

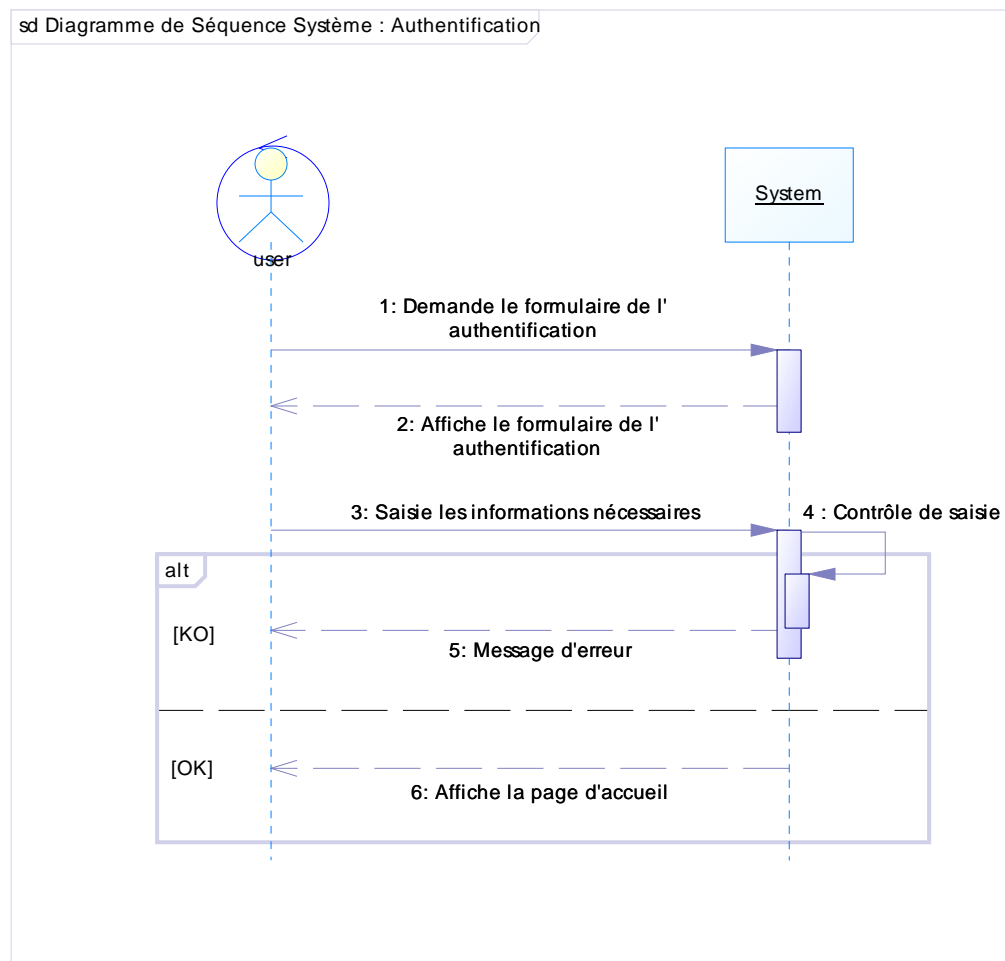
Dans la phase de conception de l'application, ce diagramme est détaillé et les messages correspondent aux messages des langages informatiques.

Au début – en analyse - ils sont utilisés pour spécifier les cas d'utilisation (Diagrammes de Séquence Système où deux objets – utilisateur et système – sont représentés).

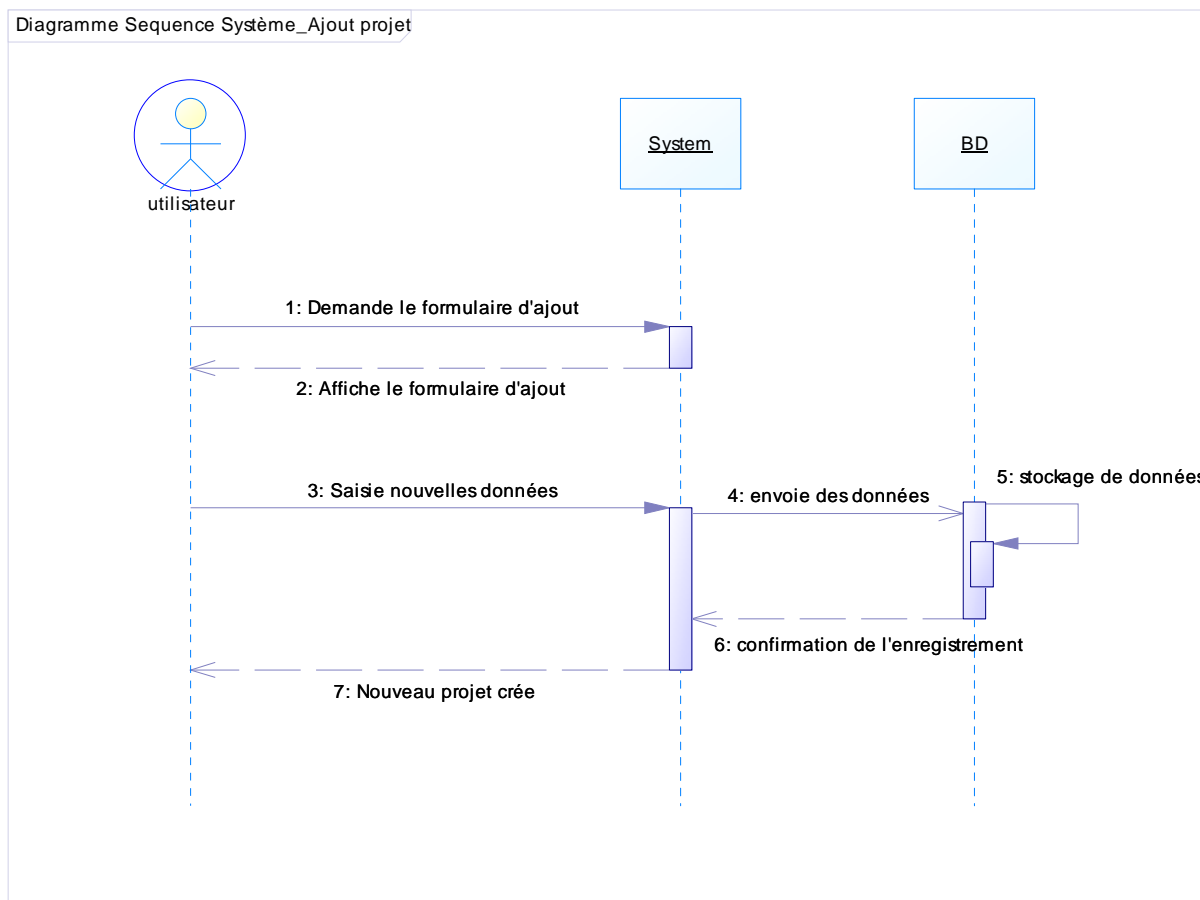
Par la suite – soit en analyse, soit en conception, ils seront plus détaillés, et les messages correspondront aux messages des langages informatiques (Diagrammes de Séquence détaillés où n objets– utilisateur, dialogues, contrôleurs et modèles - sont représentés).

Les acteurs et les classes sont représentés par des rectangles ou par des icônes et une ligne discontinue verticale appelée ligne de vie (lifeline). La ligne de vie représente la durée de la présence de l'objet – l'instance de la classe – en mémoire.

3.5.1 Diagramme de séquence système : Authentification



3.5.2 Diagramme de séquence système : Ajout projet



3.6 Le Diagramme d'activité

Le diagramme d'activité représente une partie de la dynamique du système (les processus, les traitements) ou d'un objet (les méthodes).

Un diagramme d'activité représente graphiquement le déroulé d'un cas d'utilisation du point de vue de l'utilisateur, d'une opération d'une classe ou un DET (Diagramme d'Etat-Transition).

Un diagramme d'activité possède un pseudo état initial (rond plein).

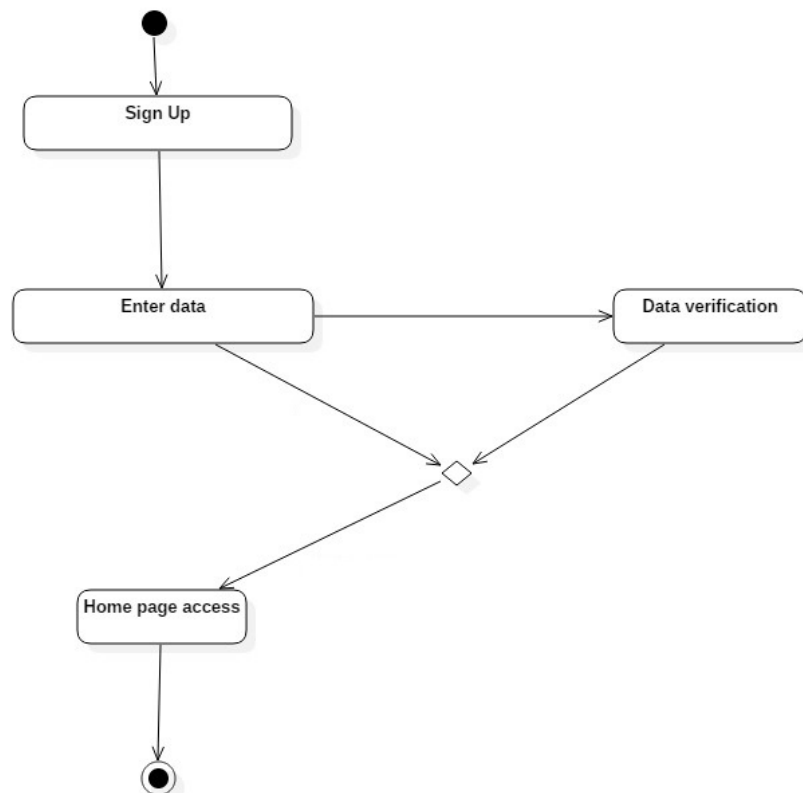
Un diagramme d'activité possède un ou plusieurs pseudo états finals (rond plein entouré d'un cercle).

Les activités sont représentées par des rectangles aux coins arrondis.

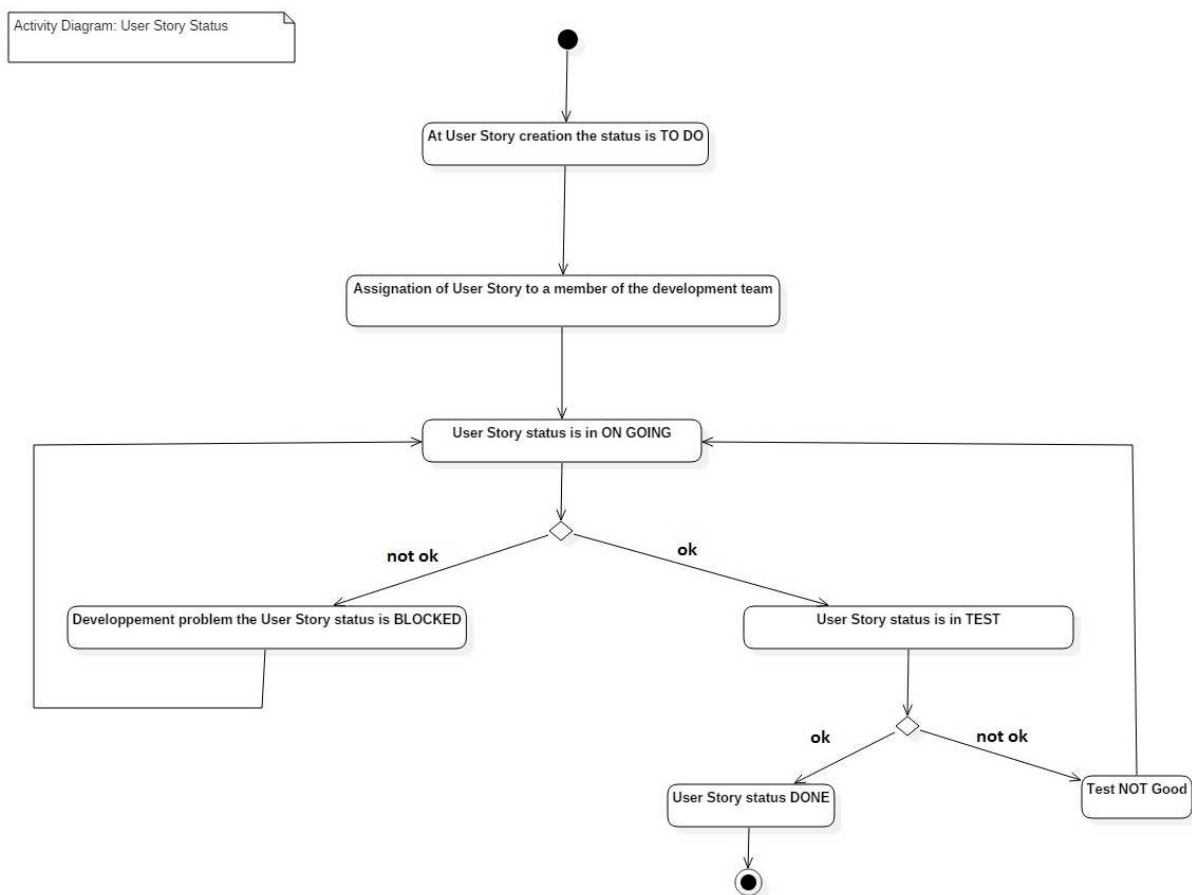
Les transitions sont représentées par des flèches.

3.6.1 Diagramme d'activité : Inscription

Activity Diagram : Sign Up



3.6.2 Diagramme d'activité : Statut User Story



CHAPITRE 4 - Conception de la Base de Données

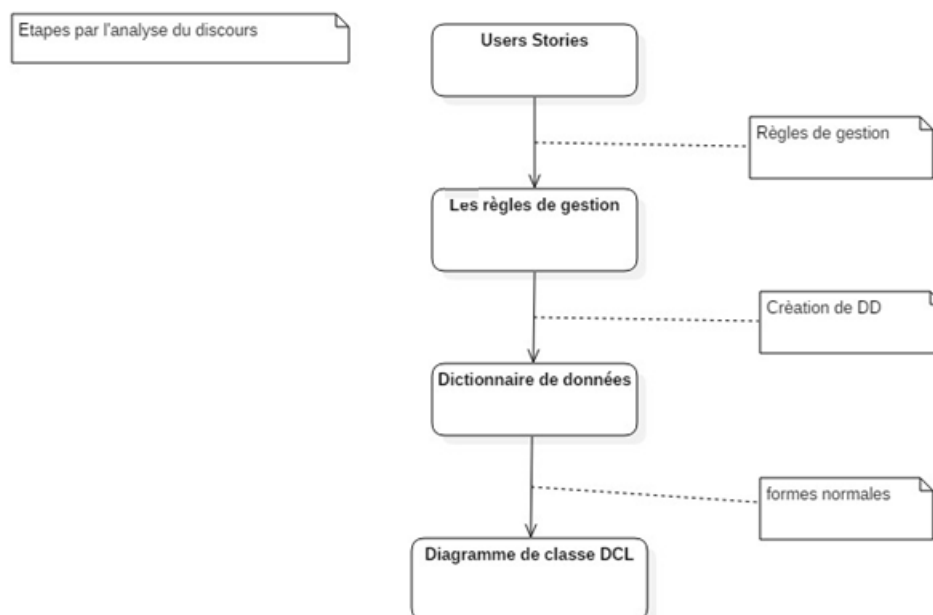
4.1 La démarche utilisée

La modélisation de la base de données de notre futur application web, nous avons suivi une démarche par l'analyse du discours (ou règles de gestion).

La démarche présentée ici es un processus visant à créer le DCL (Diagramme de classes) du domaine et donc orientée vers la création de la Base de Données.

Dans un premier temps nous avons construit le Diagramme de classes de Domaine (DCL), en passant par les étapes suivantes :

- ✓ Les règles de gestion (les utilisateurs décrivent leurs besoins et expliquent la sémantique des données),
- ✓ L'élaboration du dictionnaire des données (DD),
- ✓ L'élaboration du DCL (création des classes puis des associations puis ajout des multiplicités).



4.1.1 Les règles de gestion

Avant de se lancer dans la création de mes classes et associations, j'ai recueilli les besoins des futurs utilisateurs de l'application. Et à partir de ces besoins, j'ai établi les règles de gestion des données à conserver.

- Pour chaque utilisateur on doit connaître (le nom, le prénom, l'email, mot de passe)
- Chaque utilisateur possède un rôle
- Chaque utilisateur possède un espace utilisateur
- Un projet peut appartenir à un ou plusieurs utilisateurs
- Un projet peut avoir une ou plusieurs user stories
- Une user story possède un ou plusieurs statut
- Un sprint possède une ou plusieurs user stories
- Une user story possède un sprint
- Un projet possède un ensemble de caractéristiques (nom, date de début, date de fin)
- Chaque user story possède un titre, description, priorité, date de création)
- L'utilisateur pourra consulter, créer, modifier ou supprimer des projets
- Un utilisateur pourra changer le statut de user story
- Un sprint possède un nombre, une date de début et une date de fin
- Chaque user story possède un statut.

4.1.2 Dictionnaire de données

Le dictionnaire des données est un document qui regroupe toutes les données qu'on aura à conserver dans notre base (et qui figureront dans le DCL). Pour chaque donnée, il indique :

- ✓ **Le code** : Il s'agit d'un libellé désignant une donnée (par exemple « name_project » pour le nom du projet)
- ✓ **La désignation** : Il s'agit d'une mention décrivant ce à quoi la donnée correspond (par exemple « nom du projet »)
- ✓ **Le type de donnée** :
 - Caractère : lorsque la donnée est uniquement composée de caractères alphabétiques
 - Entier ou Réel : lorsque la donnée est composée uniquement de nombres (entiers ou réels)
 - Date : lorsque la donnée est une date (au format AAAA-MM-JJ)
 - Booléen : Vrai ou Faux
- ✓ **La taille** : Elle s'exprime en nombre de caractères ou de chiffres. Dans le cas d'une date au format AAAA-MM-JJ, on compte également le nombre de caractères, soit 10 caractères. Pour ce qui est du type booléen, ce n'est pas nécessaire de préciser la taille (ceci dépend de l'implémentation du SGBDR).³

Après l'étude des règles de gestion, j'ai pu établir le dictionnaire des données suivant :

Nom	Code	Désignation	Type	Taille
Users	id_user	Identifiant d'un utilisateur	Entier	(11)
	first_name_user	Prénom d'un utilisateur	Caractère	(50)
	last_name_user	Nom d'un utilisateur	Caractère	(50)
	email	L'email d'un utilisateur	Caractère	(255)
	password	Mot de passe d'un utilisateur	Caractère	(255)
	company	La société d'un utilisateur	Caractère	(30)
	created_date_user	La date de création d'un utilisateur	Date	
Projects	id_project	Identifiant d'un projet	Entier	(11)
	name_project	Nom d'un projet	Caractère	(255)
	start_date_project	La date de début d'un projet	Date	
	end_date_project`	La date de fin d'un projet	Date	
Roles	id_role	Identifiant d'un rôle	Entier	(11)
	name_role	Le nom d'un rôle	Caractère	(20)
Sprints	id_sprint	Identifiant d'un sprint	Entier	(11)
	number_sprint	Le numéro d'un sprint	Entier	(11)
	start_date_sprint	La date de début d'un sprint	Date	
	end_date_sprint	La date de fin d'un sprint	Date	
Statuses	id_status	Identifiant d'un status	Entier	(11)
	name_status	Le nom d'un statut	Caractère	(20)
User Stories	id_user_story	Identifiant d'une user story	Entier	(11)
	title	Le titre d'une user story	Caractère	(70)
	description	La description d'une user story	Text	
	priority	La priorité d'une user story	Entier	(11)
	creation_date_user_story	La date de la création d'une user story	Date	

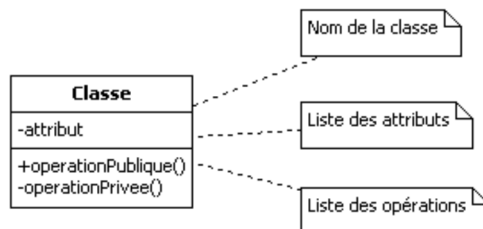
4.2 Les diagrammes de conception de la base de données

A partir d'analyse de règles de gestion j'ai réalisé un diagramme de classes de données (DCL) avec StarUML et à l'aide des fonctionnalités que nous offre l'outil de conception Power AMC j'ai réalisé le modèle conceptuel de données MCD), j'ai généré le modèle logique de données (MLD) à partir du MCD et le modèle physique de données (MPD) à partir du MLD.

4.2.1 Le diagramme de classes (DCL)

En conception, le diagramme de classes représente la structure statique du code, par le biais des attributs (propriétés) et des relations entre classes (associations), mais il contient également les opérations (aussi appelées méthodes) qui décrivent les responsabilités dynamiques des classes logicielles.³

C'est un diagramme normalisé de l'UML.



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.

Une **association** (ou relation) définit un lien sémantique entre plusieurs entités.

Chaque extrémité d'une association porte une indication de multiplicité.

Les **multiplicités** (dans UML) expriment le nombre minimum et maximum d'objets d'une classe qui peuvent être reliés à des objets d'une autre classe.⁴

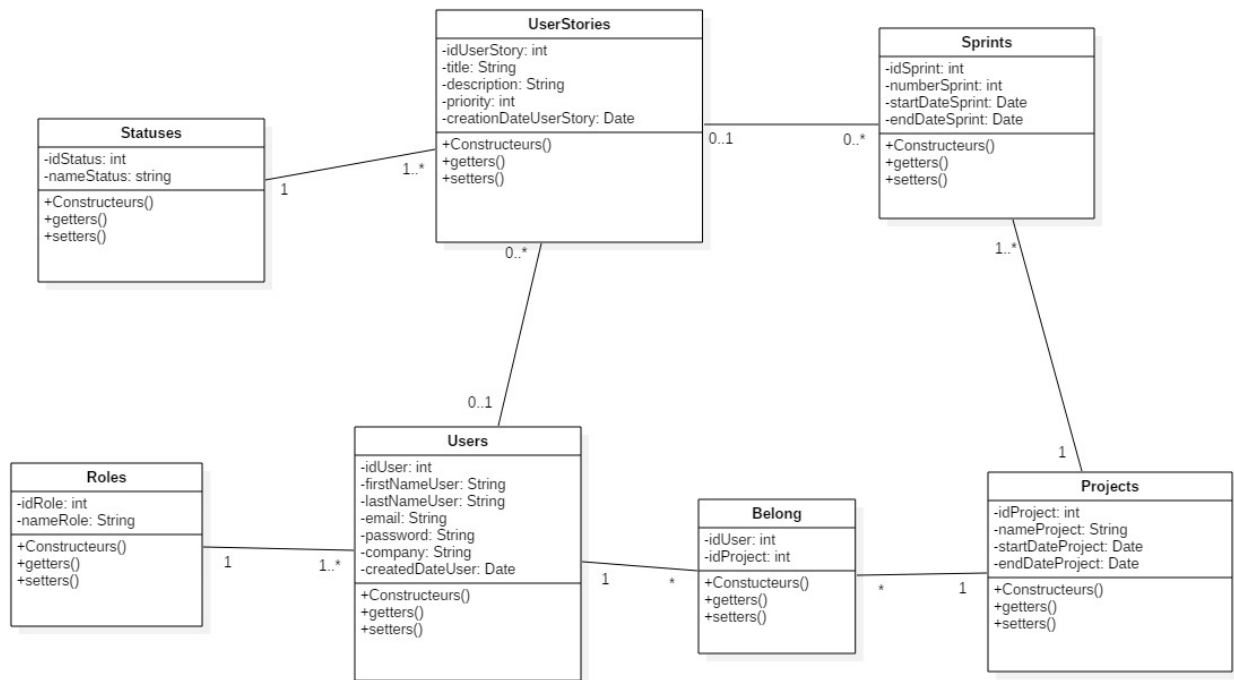
Tableau 1.1 Terminologie

Entité-association (Merise)	UML
Entité	Classe
Association (Relation)	Association (Relation)
Occurrence	Objet
Cardinalité	Multiplicité
Modèle conceptuel de données (Merise)	Diagramme de classes

Tableau 1.2 Cardinalités/multiplicités

Cardinalités (Merise)	Multiplicités d'UML
0, 1	0..1
1, 1	1
0, N	0..* ou *
1, N	1..*
N, N	N..N

Class Diagram : Board Agile

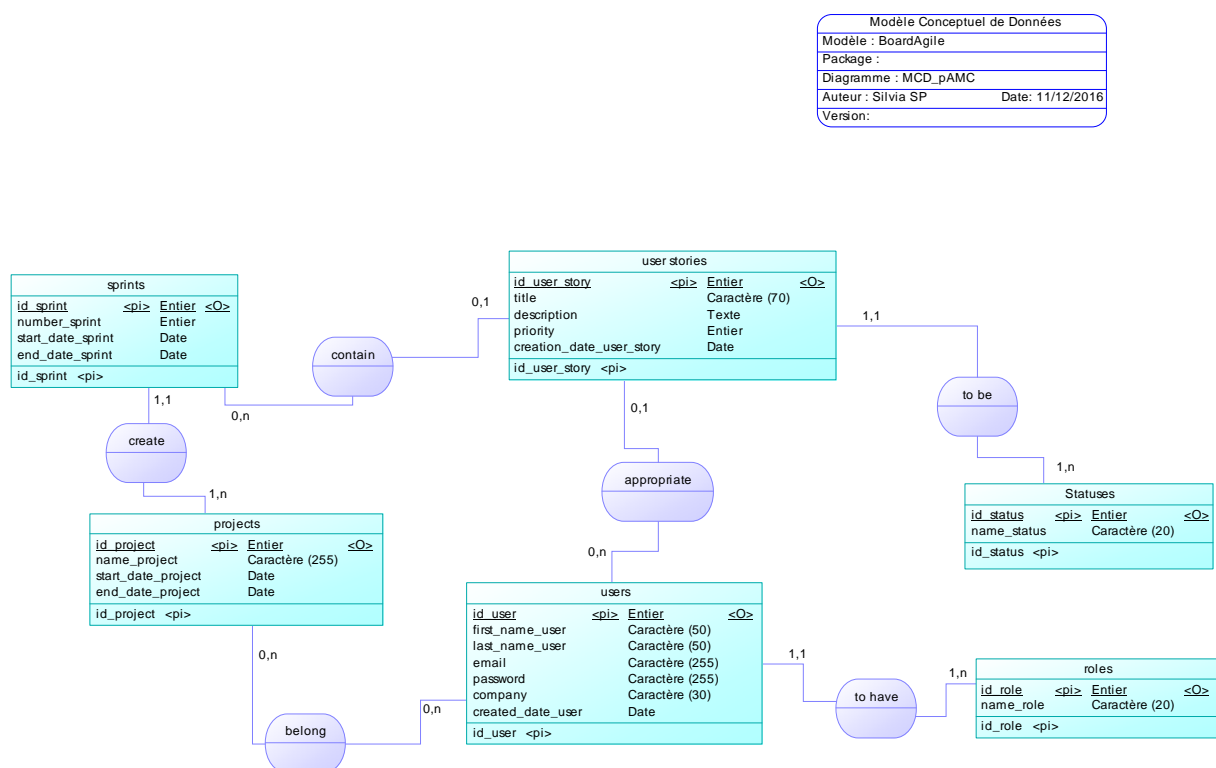


4.2.2 Modèle conceptuel de données (MCD)

Le modèle conceptuel des données (MCD) a pour but de représenter de façon structurée les données qui seront utilisées par le système d'information. Le modèle conceptuel des données décrit la sémantique c'est à dire le sens attaché à ces données et à leurs rapports et non à l'utilisation qui peut en être faite.

Une **entité** représente un objet (acteur, document, concept, ...), ou plus exactement un ensemble d'objets ayant les mêmes caractéristiques.

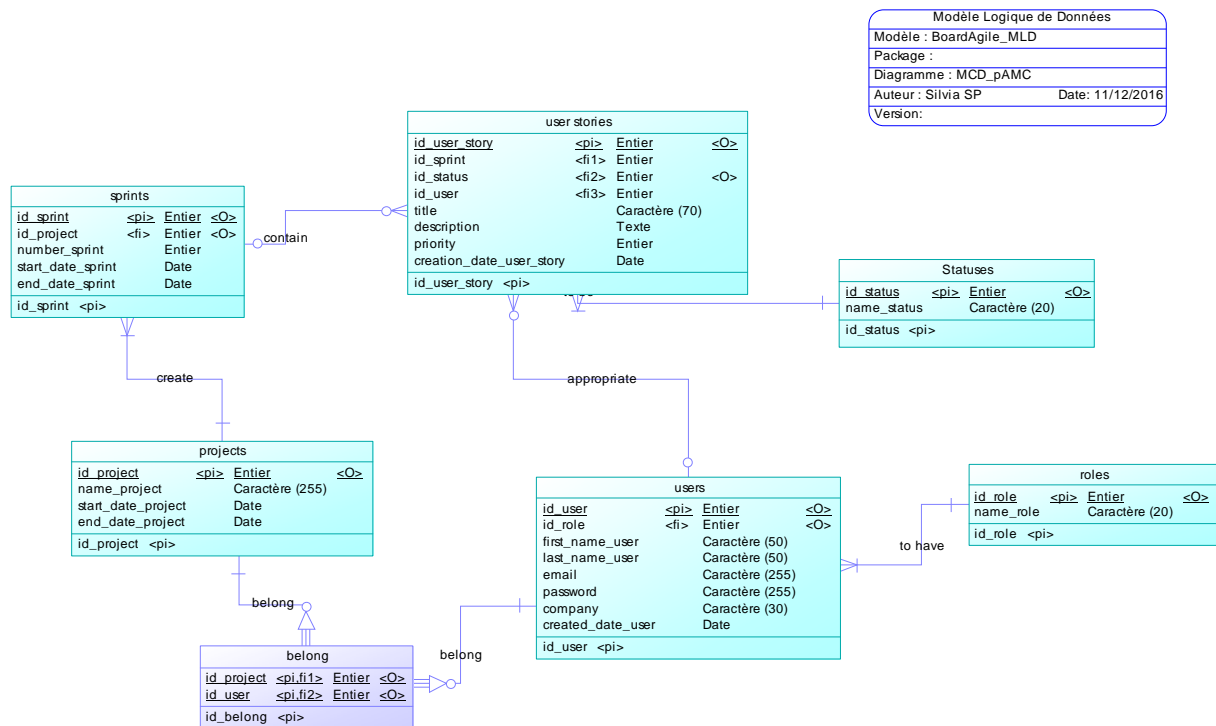
Les **cardinalités** (dans Merise) ce sont des expressions qui permettent d'indiquer combien de fois au minimum et au maximum le lien entre 2 entités peut se produire. Les cardinalités traduisent des règles de gestion.



4.2.3 Modèle logique de données (MLD)

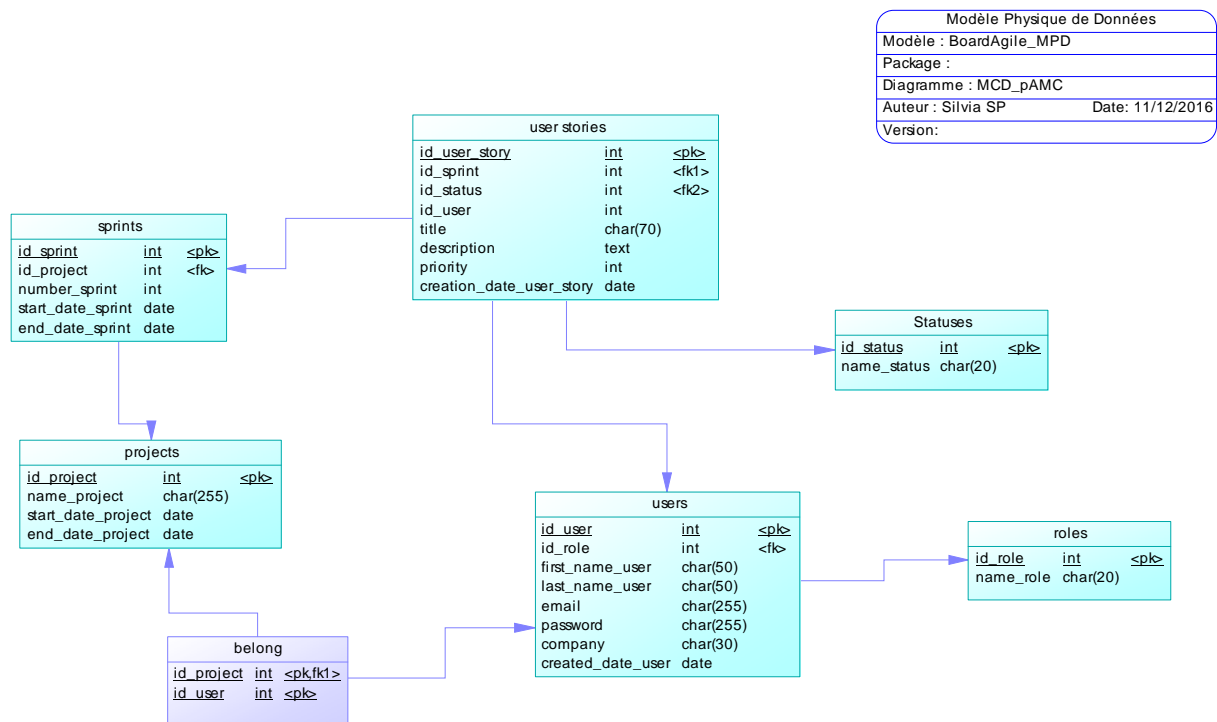
Le modèle logique de données (MLD) est composé uniquement de ce que l'on appelle des relations. Ces relations sont à la fois issues des classes du DCL mais aussi d'associations.

Une relation possède un nom qui correspond en général à celui de la classe ou de l'association qui lui correspond. Elle possède aussi une ou plusieurs clés primaires.



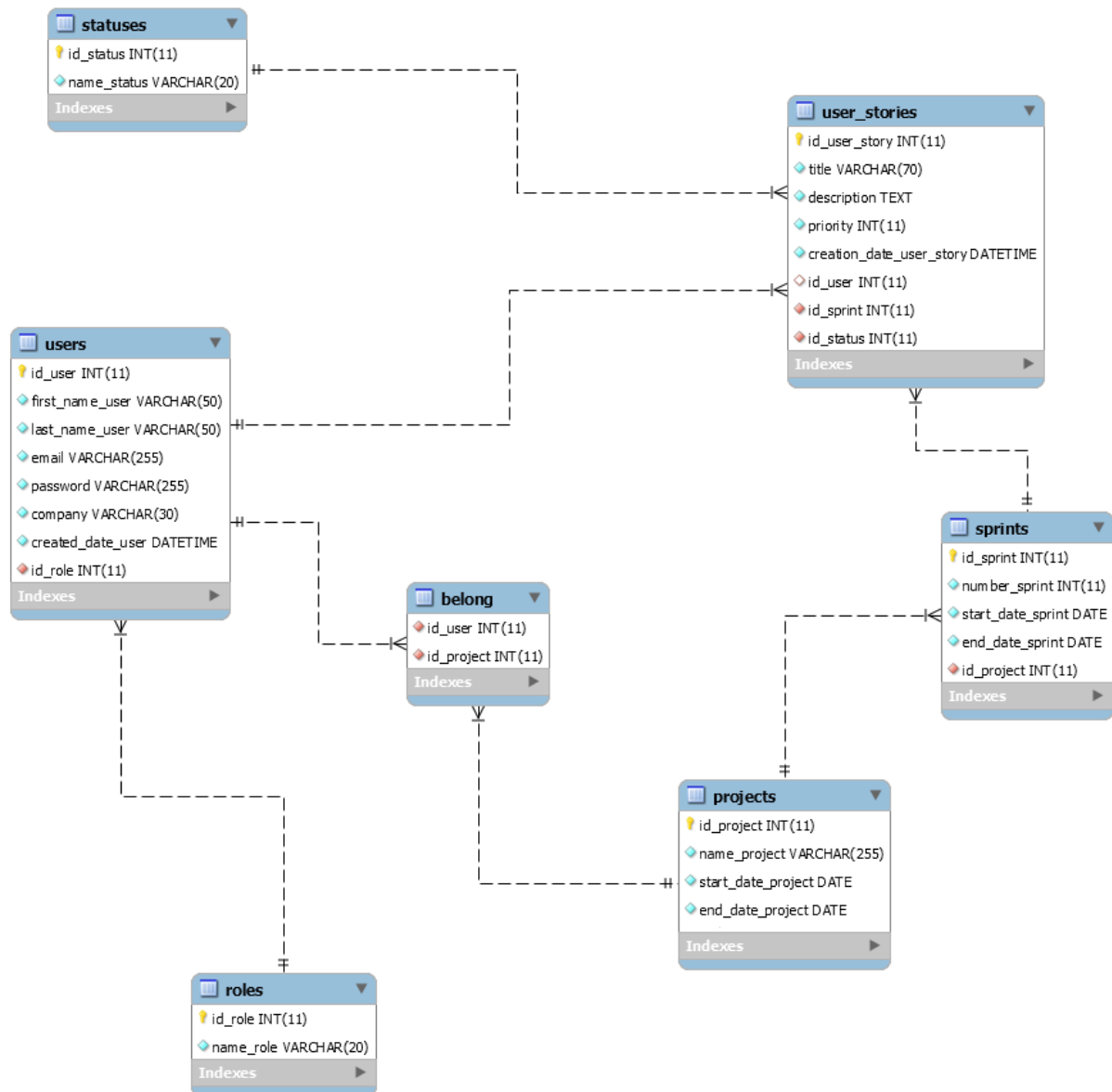
4.2.4 Modèle physique de données (MPD)

Un modèle physique de données découlant d'un modèle logique qui spécifie les détails d'implantation du modèle logique dans un SGBDR, il est formulé grâce à un script de définition de données.



4.3 Le schéma de la Base de Données

J'ai réalisé le Schéma de la Base de Données avec le logiciel MySQL Workbench, en faisant un reverse Engineer Database une fois ma base et les tables créées dans le *phpMyAmdin*.



4.6 Le code SQL (LDD) de création de la Base de Données

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

SQL permet de demander des opérations d'algèbre relationnelle telles que : l'intersection, la sélection et la jointure.

Le sous-langage LDD permet de créer (CREATE), modifier (ALTER) et supprimer (DROP) les objets de la base de données.

Le LDD pour manipuler la structure des données d'une BD.

Le LMD pour manipuler les données d'une BD (via CRUD).

Voici le script de création de la base de données créée avec *phpMyAmdin* (via XAMPP) BOARD AGILE sur laquelle je travaille en développement :

Création de la base de données :

```
-----
--
-- Base de données : `board_agile`
--
CREATE DATABASE IF NOT EXISTS `board_agile`
DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci;
USE `board_agile`;

-----

--
-- Structure de la table `belong`
--

DROP TABLE IF EXISTS `belong`;
CREATE TABLE IF NOT EXISTS `belong` (
  `id_user` int(11) NOT NULL,
  `id_project` int(11) NOT NULL,
  KEY `FK_belong_id_user_idx` (`id_user`),
  KEY `FK_belong_id_project_idx` (`id_project`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-----

--
-- Structure de la table `projects`
--

DROP TABLE IF EXISTS `projects`;
CREATE TABLE IF NOT EXISTS `projects` (
  `id_project` int(11) NOT NULL AUTO_INCREMENT,
  `name_project` varchar(255) CHARACTER SET latin1 NOT NULL,
  `start_date_project` date NOT NULL,
  `end_date_project` date NOT NULL,
  PRIMARY KEY (`id_project`)
```

```

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-----

--
-- Structure de la table `roles`
--

DROP TABLE IF EXISTS `roles`;
CREATE TABLE IF NOT EXISTS `roles` (
  `id_role` int(11) NOT NULL AUTO_INCREMENT,
  `name_role` varchar(20) CHARACTER SET latin1 NOT NULL,
  PRIMARY KEY (`id_role`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-----

--
-- Structure de la table `sprints`
--

DROP TABLE IF EXISTS `sprints`;
CREATE TABLE IF NOT EXISTS `sprints` (
  `id_sprint` int(11) NOT NULL AUTO_INCREMENT,
  `number_sprint` int(11) NOT NULL,
  `start_date_sprint` date NOT NULL,
  `end_date_sprint` date NOT NULL,
  `id_project` int(11) NOT NULL,
  PRIMARY KEY (`id_sprint`),
  UNIQUE KEY `FK_sprints_id_project` (`id_project`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-----

--
-- Structure de la table `statuses`
--

DROP TABLE IF EXISTS `statuses`;
CREATE TABLE IF NOT EXISTS `statuses` (
  `id_status` int(11) NOT NULL AUTO_INCREMENT,
  `name_status` varchar(20) CHARACTER SET latin1 NOT NULL,
  PRIMARY KEY (`id_status`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-----

--
-- Structure de la table `users`
--

DROP TABLE IF EXISTS `users`;
CREATE TABLE IF NOT EXISTS `users` (
  `id_user` int(11) NOT NULL AUTO_INCREMENT,
  `first_name_user` varchar(50) CHARACTER SET latin1 NOT NULL,
  `last_name_user` varchar(50) CHARACTER SET latin1 NOT NULL,
  `email` varchar(255) CHARACTER SET latin1 NOT NULL,
  `password` varchar(255) CHARACTER SET latin1 NOT NULL,
  `company` varchar(30) CHARACTER SET latin1 NOT NULL,
  `created_date_user` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `id_role` int(11) NOT NULL,

```

```

PRIMARY KEY (`id_user`),
UNIQUE KEY `email` (`email`),
KEY `id_role` (`id_role`) USING BTREE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-----

--
-- Structure de la table `user_stories`
--

DROP TABLE IF EXISTS `user_stories`;
CREATE TABLE IF NOT EXISTS `user_stories` (
  `id_user_story` int(11) NOT NULL AUTO_INCREMENT,
  `title` varchar(70) CHARACTER SET latin1 NOT NULL,
  `description` text CHARACTER SET latin1 NOT NULL,
  `priority` int(11) NOT NULL,
  `creation_date_user_story` datetime NOT NULL,
  `id_user` int(11) DEFAULT NULL,
  `id_sprint` int(11) NOT NULL,
  `id_status` int(11) NOT NULL,
  PRIMARY KEY (`id_user_story`),
  UNIQUE KEY `title` (`title`),
  KEY `FK_user_stories_id_sprint` (`id_sprint`),
  KEY `FK_user_stories_id_status` (`id_status`),
  KEY `FK_user_stories_id_user` (`id_user`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Contraintes pour les tables exportées
--

--
-- Contraintes pour la table `belong`
--
ALTER TABLE `belong`
  ADD CONSTRAINT `FK_belong_id_project` FOREIGN KEY (`id_project`)
REFERENCES `projects` (`id_project`) ON DELETE NO ACTION ON UPDATE NO AC-
TION,
  ADD CONSTRAINT `FK_belong_id_user` FOREIGN KEY (`id_user`) REFERENCES
`users` (`id_user`) ON DELETE NO ACTION ON UPDATE NO ACTION;

--
-- Contraintes pour la table `sprints`
--
ALTER TABLE `sprints`
  ADD CONSTRAINT `FK_sprints_id_project` FOREIGN KEY (`id_project`) REFER-
ENCES `projects` (`id_project`) ON DELETE NO ACTION ON UPDATE NO ACTION;

--
-- Contraintes pour la table `users`
--
ALTER TABLE `users`
  ADD CONSTRAINT `FK_users_id_role` FOREIGN KEY (`id_role`) REFERENCES
`roles` (`id_role`);

--
-- Contraintes pour la table `user_stories`
--
ALTER TABLE `user_stories`

```

```
ADD CONSTRAINT `FK_user_stories_id_sprint` FOREIGN KEY (`id_sprint`) REF-  
ERENCES `sprints` (`id_sprint`),  
ADD CONSTRAINT `FK_user_stories_id_status` FOREIGN KEY (`id_status`) REF-  
ERENCES `statuses` (`id_status`) ON DELETE NO ACTION ON UPDATE NO ACTION,  
ADD CONSTRAINT `FK_user_stories_id_user` FOREIGN KEY (`id_user`) REFER-  
ENCES `users` (`id_user`);
```

CHAPITRE 5 - Conception de l'application

5.1 Le diagramme de séquence détaillé

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 Board Agile :

- ✓ les utilisateurs qui interagissent avec le système au moyen des Boundaries,
- ✓ les Boundaries qui envoient des instructions au contrôleur,
- ✓ le Contrôleur qui fait l'intermédiaire entre les Boundaries et les entités et qui orchestre l'exécution des commandes,
- ✓ les modèles (Entities et les DAOs) qui représentent les données du système.

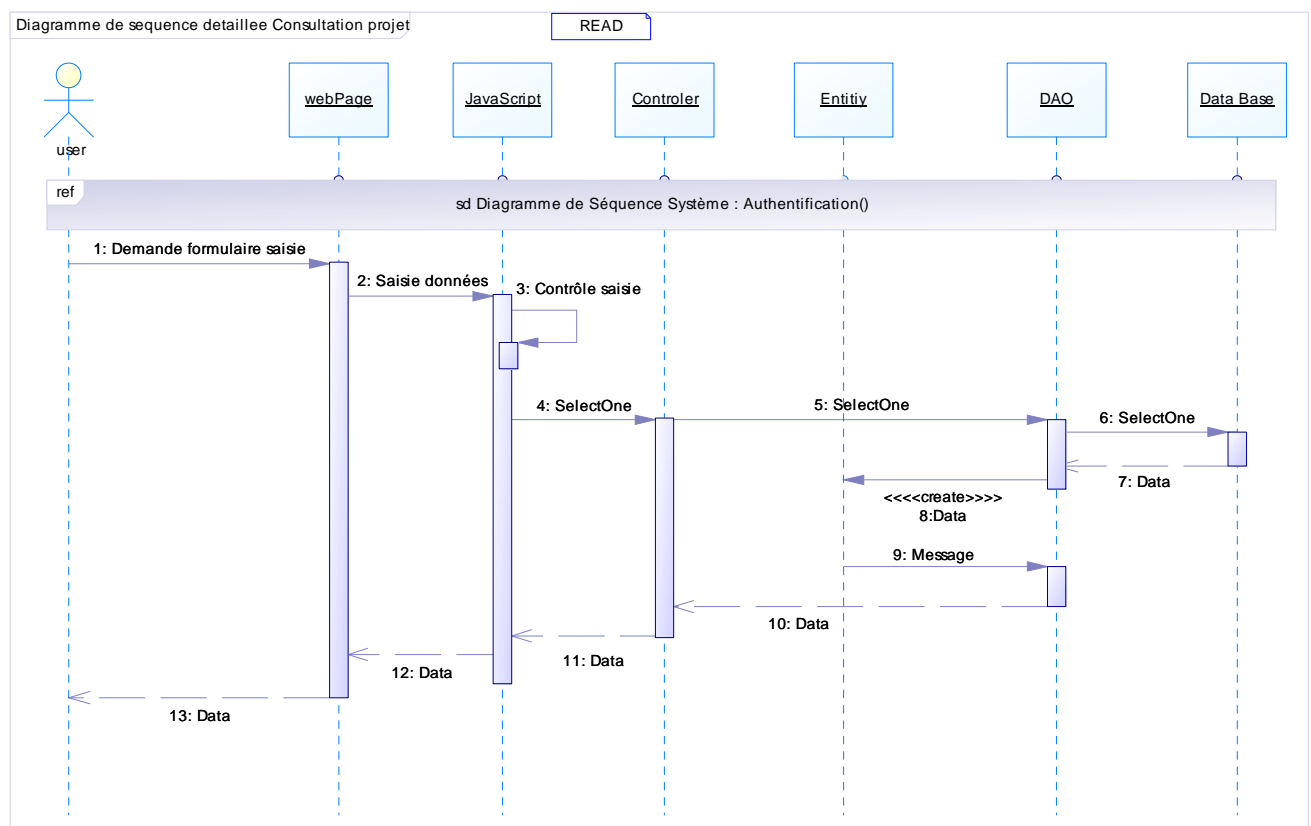
Dans l'application Board Agile :

Boundaries : Pages Web

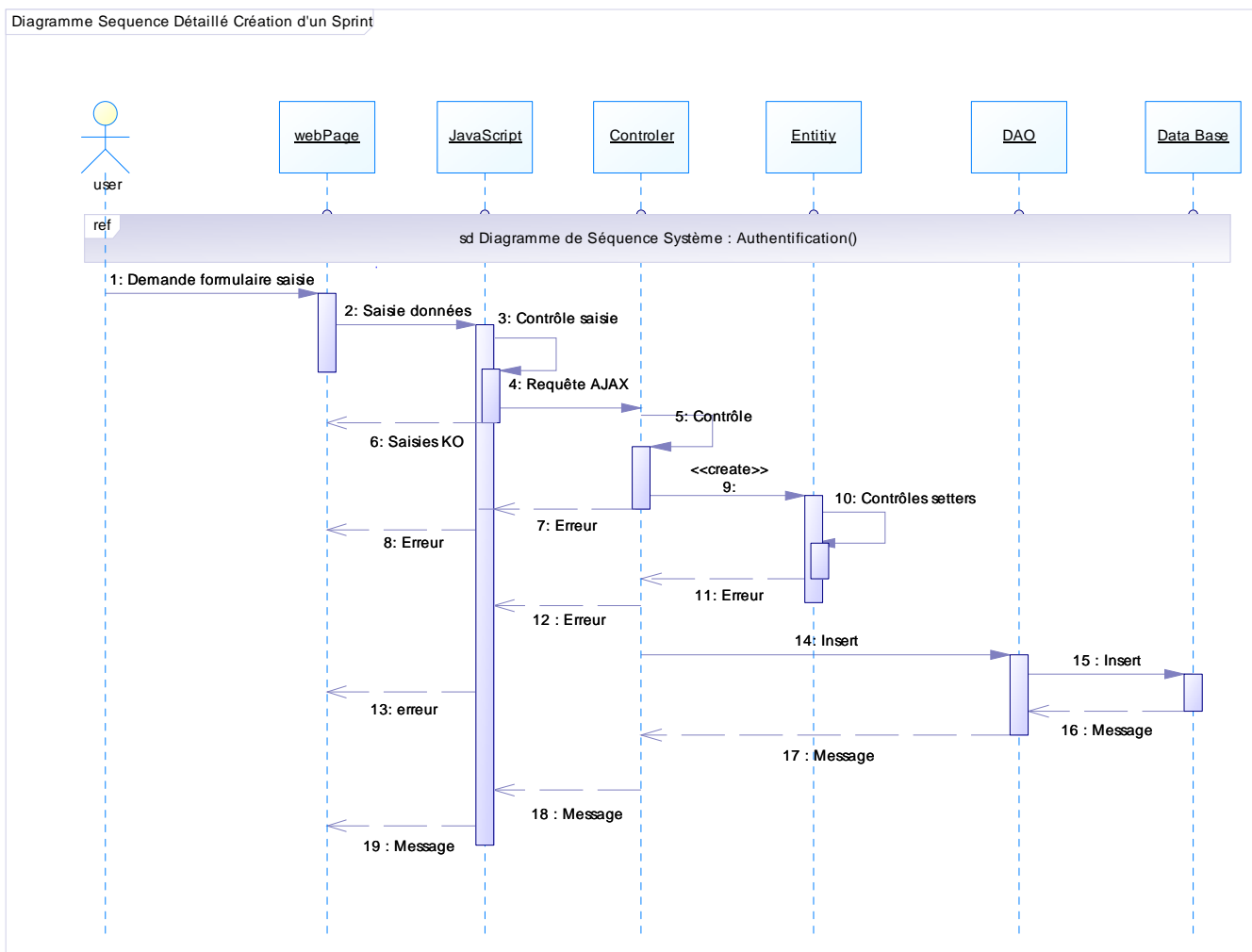
Contrôleur : Servlet

Entities : classes POJO en Java

5.1.1 Diagramme de séquence détaillé : Consultation d'un projet



5.1.2 Diagramme de séquence détaillé : Création d'un sprint



CHAPITRE 6 – Développement

6.1 Technologies utilisées

Les langages utilisés sont HTML 5, CSS 3 (Bootstrap, Bootsfaces), JavaScript.

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.

Le langage utilisé est Java EE (Java Enterprise Edition).

Le serveur d'application Apache -Tomcat.

Le SGBDR utilisé est MySQL pour sa robustesse, sa sécurisation des données et sa simplicité, via XAMPP Apache.

6.2 Introduction au langage Java EE

Java Enterprise Edition, ou **Java EE** (anciennement **J2EE**), est une spécification pour la technique Java d'Oracle plus particulièrement destinée aux applications d'entreprise.

Java EE permet de développer des applications web en Java. On parle aussi de clients légers.

6.2.1 Environnement requis pour le développement

Avant de commencer à développer, nous nous sommes assuré que nous disposons sur notre machine de développement de l'environnement requis et de la bonne configuration de tous les outils nécessaires.

✓ Logiciels tiers

Pour utiliser Java, j'ai installé JDK (Java Development Kit) sur ma machine, qui permet de créer des applications Java. Il contient des outils de compilation et une version du JRE.

Le JRE (Java Runtime Environment) permet d'exécuter des applications Java.

J'ai installé également **Tomcat** un conteneur web qui permet d'exécuter les applications Java Web (intégré dans le serveur d'application Apache –Tomcat).

Apache représente de son côté le serveur HTTP qui reçoit les requêtes du navigateur Web (requêtes HTTP).

✓ Environnement de développement intégré (EDI)

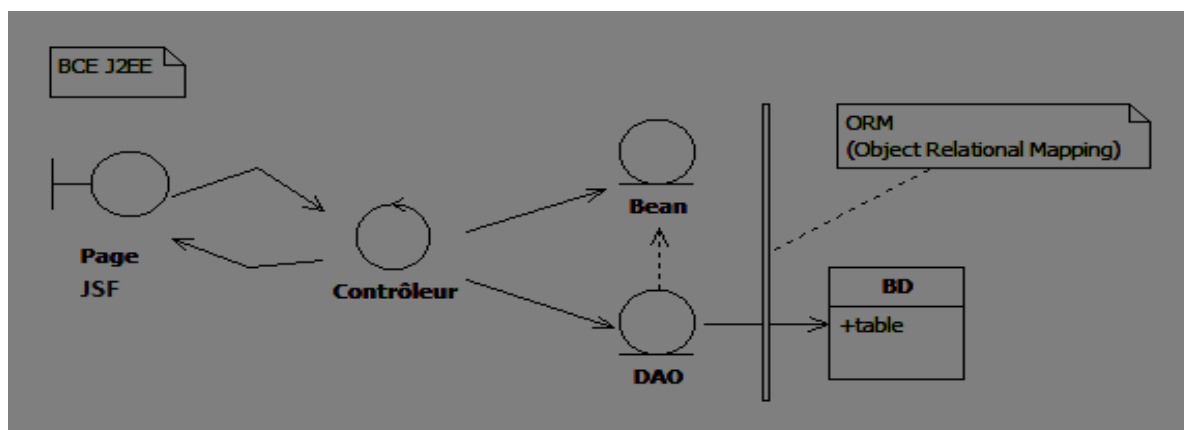
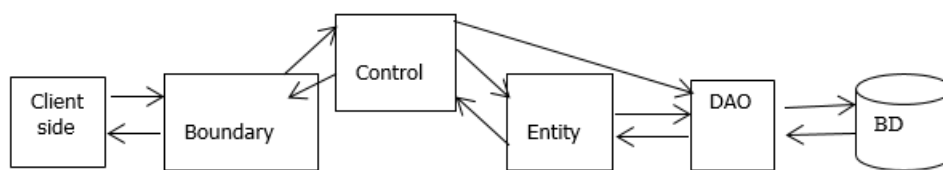
Tous au long de la partie développement, nous avons utilisé Eclipse qui est un IDE, *Integrated Development Environment* (EDI environnement de développement intégré en français), c'est-à-dire un logiciel qui simplifie la programmation en proposant un certain nombre de raccourcis et d'aide à la programmation. Il est développé par IBM, est gratuit et disponible pour la plupart des systèmes d'exploitation.

Au fur et à mesure de la programmation, Eclipse compile automatiquement le code écrit, en soulignant en rouge ou jaune les problèmes qu'il détecte. Il souligne en rouge les parties du programme qui ne compilent pas, et en jaune les parties qui compilent mais peuvent éventuellement poser problème (on dit qu'Eclipse lève un avertissement, ou *warning* en anglais). Pendant l'écriture du code, cela peut sembler un peu déroutant au début, puisque tant que la ligne de code n'est pas terminée (en gros jusqu'au point-virgule), Eclipse indique une erreur dans le code.⁵

6.2.2 Architecture

6.2.2.1 Le modèle design pattern ECB (Entity-Boundary-Control)

Ce modèle décrit une architecture de composants à trois niveaux :



Boundary : c'est la partie *présentation*. Ce sont les codes HTML, JSF dans notre cas.

Control : c'est le niveau chargé du *traitement des requêtes*. Ce sont les servlets pour le web.

Entity : c'est la *logique métier* avec la manipulation des données de l'application. Ce sont les JavaBeans ou des classes et packages JAVA.

La couche Entity est en "connexion" avec le DAO (Data Access Object), qui lui est dépendant de la source de données (BD MySQL) mais indépendant de l'application.

Les composants métier (POJO) doivent être utilisables par tout type d'application (WEB, C/S lourd, ...).

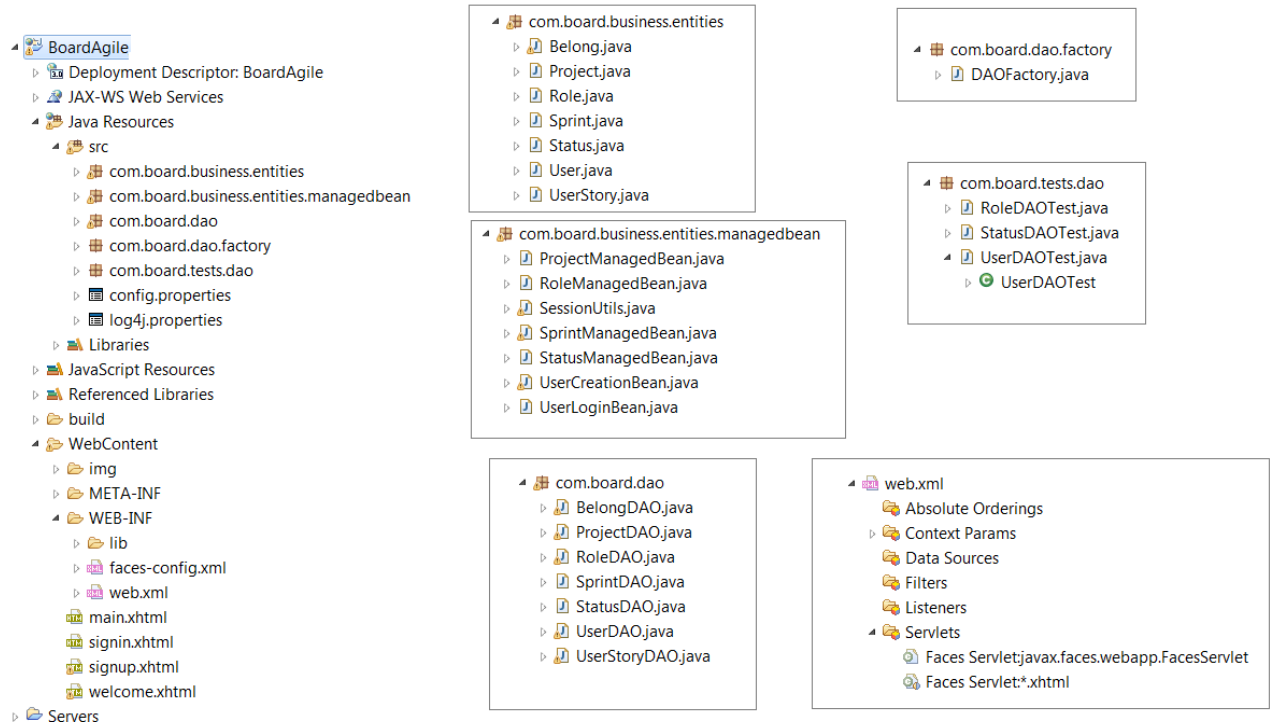
*L'objectif du pattern ECB est de favoriser la décomposition d'une application en composants réutilisables relativement indépendants entre eux.*⁶

6.2.3 Création et configuration de l'application

6.2.3.1 Arborescence de notre application

La partie Back-end: Java Resources

La partie Front-end: WebContent



Connexion BD :

```

config.properties
  agileboard.connection.url=jdbc:mysql://localhost:33...
  agileboard.connection.driver=com.mysql.jdbc.Driver
  agileboard.connection.username=root
  agileboard.connection.password=
  
```

6.3 Interfaces web

6.3.1 Code HTML

6.3.1.1 Page Accueil (Welcome)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:b="http://bootsfaces.net/ui">
  <h:head>
    <title>Board Agile</title>
  </h:head>

  <!-- Add 50px padding top to the body for the navbar -->
  <h:body style="padding-top: 50px; padding-bottom: 20px;">
    <b:navBar brand="BoardAgile" brandHref="#" inverse="true" fixed="top">
      <b:navBarLinks>
        <b:navLink value="Sign In" outcome="signin"/>
        <b:navLink value="Sign Up" outcome="signup"/>
      </b:navBarLinks>
    </b:navBar>

    <!-- Add jumbotron grey -->
    <b:jumbotron>
      <b:container>
        <h1>Bienvenue dans Board Agile!</h1>
        <p>Ce site vous permettra quel que soit votre location de partager et
        suivre ensemble l'avancement de vos différents projets de manière in-
        teractive.</p>
      </b:container>

      <!-- Add blue button -->
      <b:button look="primary" value="Avoir plus d'infos »" action="signin" />
    </b:jumbotron>

    <!-- Add container with rows -->
    <b:container>
      <b:row>
        <b:column col-md="4">
          <h2>Collaboratif</h2>
          <p>Votre équipe pourra a travers les daily meetings progresser
          de façon sûre sur vos projets</p>
          <p><b:button outcome="#" value="View details »"/></p>
        </b:column>
        <b:column col-md="4">
          <h2>Interactif</h2>
```

via la kinect

```
<p>Finis le temps de la souris! Chacun pourra utiliser sa main
via la kinect</p>
<p><b:button outcome="#" value="View details »"/></p>
</b:column>
<b:column col-md="4">
  <h2>Multisites</h2>
  <p>Plus besoin de se reunir en un seul lieu. Grace a cette ap-
  plication, vous pourrez vous connecter et interrager dans les
  daily meetings peu importe l'endroit ou vous vous trouvez.</p>
  <p><b:button outcome="#" value="View details »"/></p>
</b:column>
</b:row>
<hr/>

<!-- Add footer -->
<footer>
<p>© Sogeti High Tech Company - 2016</p>
</footer>
</b:container>

</h:body>
</html>
```

6.3.1.2 Page Sign In

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
  xmlns:b="http://bootsfaces.net/ui"
  xmlns:f="http://java.sun.com/jsf/core"
  xmlns:h="http://java.sun.com/jsf/html">
  <h:head>
    <title>BoardAgile - Login</title>

    <!-- CSS code -->

    <style type="text/css">
      .form-signin {
        margin: 0 auto;
        max-width: 330px;
        padding: 15px;
      }

      h2.form-signin-heading, li, #remember, #forgot
      {
        color:#ffffff;
      }

      #forgot{
        text-align:center;
      }
    </style>

    </h:head>
    <h:body style="padding-top: 60px; background-image: url('img/bagile_bkg.jpg');
    background-size: cover;">
      <b:navBar brand="BoardAgile" brandHref="#" inverse="true" fixed="top">
        <b:navBarLinks>
          <b:navLink value="Back" outcome="welcome"/>
          <b:navLink value="Sign Up" outcome="signup"/>
        </b:navBarLinks>
      </b:navBar>
    </h:body>
  </html>
```

```

        </b:navbarLinks>
    </b:navBar>

    <b:container>
        <h:form styleClass="form-signin" prependId="false">
            <h2 class="form-signin-heading">Please Sign In</h2>
            <b:inputText value="#{userLoginBean.email}" placeholder="Email">
                <f:facet name="prepend">
                    <b:icon name="user" />
                </f:facet>
            </b:inputText>

            <b:inputSecret value="#{userLoginBean.password}" place-
holder="Password ">
                <f:facet name="prepend">
                    <b:iconAwesome name="key" />
                </f:facet>
            </b:inputSecret>

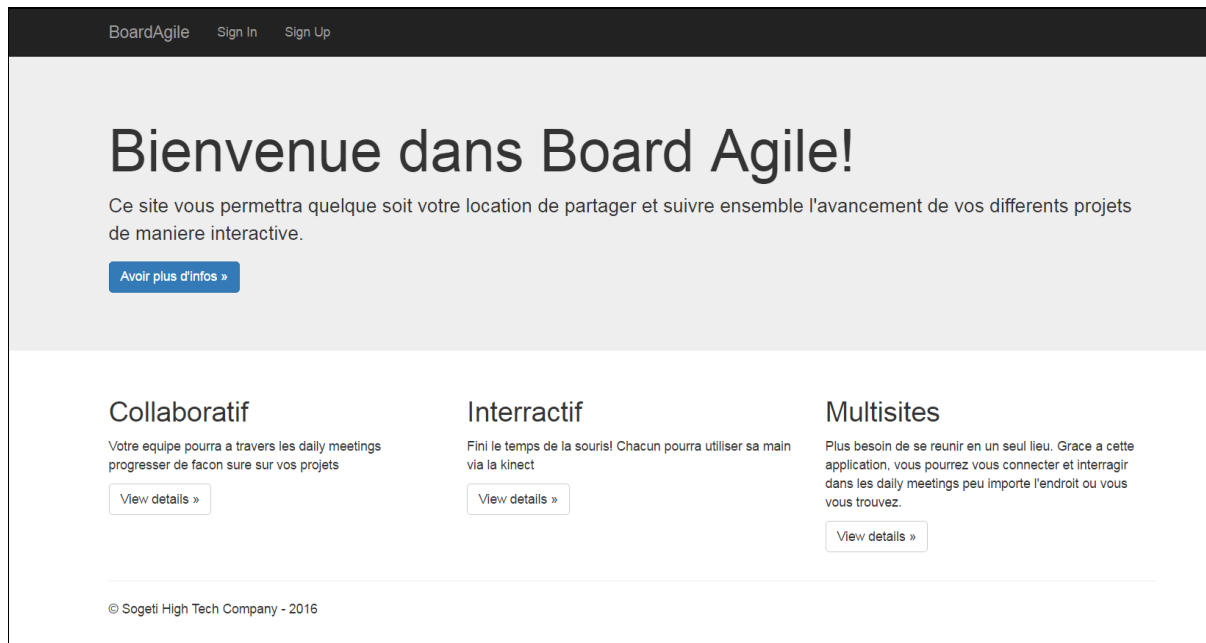
            <h:messages globalOnly="true" infoClass="info" />

            <b:commandButton value="Sign In" action="#{userLoginBean.vali-
dateUsernamePassword}" look="primary btn-block" icon="log-in" size="lg" />
            <h:link value="forgot password" outcome="signin" id="forgot"
/>
        </h:form>
    </b:container>
</h:body>
</html>

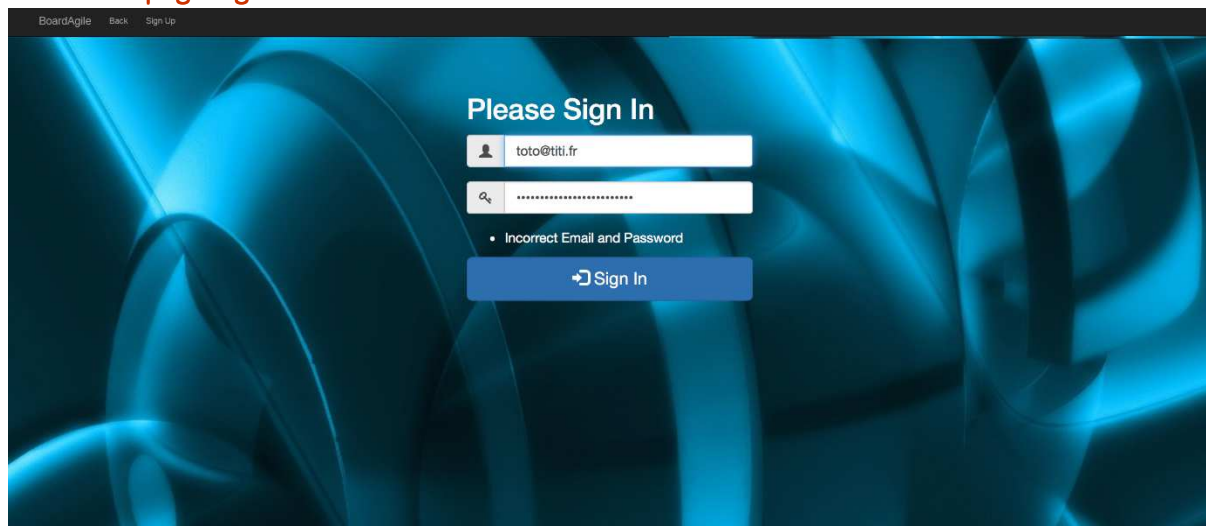
```


6.3.2 Aperçu dans le navigateur

6.3.2.1 La page Accueil (Welcome)



6.3.2.2 La page Sign In



6.4 Code pour les Entities et DAO

6.4.1 Code User DAO

```
package com.board.dao;

import java.sql.Connection;
import java.sql.Date;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.Timestamp;
import java.util.ArrayList;
import java.util.List;

import com.board.business.entities.Project;
import com.board.business.entities.User;

/**
 *
 * @author spanorsa
 *
 */

public class UserDAO {

    // Attributs
    private Connection connect;
    private final String userTabName = "board_agile.users";
    private final String idUser = "id_user";
    private final String idRole = "id_role";
    private final String firstNameUser = "first_name_user";
    private final String lastNameUser = "last_name_user";
    private final String uEmail = "email";
    private final String uPassword = "password";
    private final String uCompany = "company";
    private final String createdAtUser = "created_date_user";

    // Constructeurs
    public UserDAO(Connection cn) {
        connect = cn;
    }

    // Delete
    public int delete (int idUser){

        //Initialisation de la valeur de retour de l'execution de la
requete SQL
        int nbLigne = -1;
        try {
            String requete = "DELETE FROM "+ userTabName + " WHERE
idUser = ?";

            // preparation de la requete
PreparedStatement ps = connect.prepareStatement(requete);
System.out.println(ps);

            // valorisation
```

```

        ps.setInt(1, idUser);

        // execution de la requete
        nbLigne = ps.executeUpdate();

    } catch (SQLException e) {
        System.err.println(e);
    }
    return nbLigne;
} /// delete

//Update
public int update(User user) {
    PreparedStatement ps;
    String requete = "UPDATE " + userTabName + " SET "
        + idRole + " = ?, "
        + firstNameUser + " = ?, "
        + lastNameUser + " = ?, "
        + uEmail + " = ?, "
        + uPassword + " = ?, "
        + uCompany + " = ?, "
        + createdAtUser + " = ? "
        + "WHERE " + idUser + " = ?";

    int resultat = -1;

    try {
        ps = connect.prepareStatement(requete);

        ps.setInt(1, user.getIdRole());
        ps.setString(2, user.getFirstNameUser());
        ps.setString(3, user.getLastNameUser());
        ps.setString(4, user.getEmail());
        ps.setString(5, user.getPassword());
        ps.setString(6, user.getCompany());
        ps.setDate(7, new Date(new java.util.Date().getTime()));
        ps.setInt(8, user.getIdUser());

        resultat = ps.executeUpdate();

        ps.close();

    } catch (SQLException e) {
        System.err.println(e.getMessage());
    } finally {
        //connect.close();
    }
    return resultat;
} /// update

//Insert
public int insert(User user) {
    PreparedStatement ps;
    String requete = "INSERT INTO " + userTabName + " ("
        + idRole + ", "
        + firstNameUser + ", "
        + lastNameUser + ", "
        + uEmail + ", "

```

```

        + uPassword + ", "
        + uCompany + ", "
        + createDateUser + ") "
    + "VALUES (?, ?, ?, ?, ?, ?, ?, ?)";
int resultat = -1;

try {
    ps = connect.prepareStatement(requete);

    ps.setInt(1, user.getIdRole());
    ps.setString(2, user.getFirstNameUser());
    ps.setString(3, user.getLastNameUser());
    ps.setString(4, user.getEmail());
    ps.setString(5, user.getPassword());
    ps.setString(6, user.getCompany());
    ps.setTimestamp(7, new Timestamp(System.currentTimeMillis()));

    resultat = ps.executeUpdate();

    ps.close();

} catch (SQLException e) {
    System.err.println(e.getMessage());
}
return resultat;
} /// insert

//Select All
public List<User> selectAll() {
    List<User> list = new ArrayList<>();
    Statement st;
    String requete = "SELECT * FROM " + userTabName;
    ResultSet rsResultat;

    try {
        st = connect.createStatement();
        rsResultat = st.executeQuery(requete);

        while (rsResultat.next()) {
            User u = new User();

            u.setIdRole(rsResultat.getInt(1));
            u.setFirstNameUser(rsResultat.getString(2));
            u.setLastNameUser(rsResultat.getString(3));
            u.setEmail(rsResultat.getString(4));
            u.setPassword(rsResultat.getString(5));
            u.setCompany(rsResultat.getString(6));
            u.setCreateDateUser(rsResultat.getDate(7));
            u.setIdUser(rsResultat.getInt(8));

            list.add(u);
        }
        st.close();
        rsResultat.close();

    } catch (SQLException e) {
        System.err.println(e.getMessage());
    } finally {
        //connect.close();
    }
}

```

```

        return list;
    } /// selectAll

    // Validation of email and password
    public User validate(String email, String password) {

        PreparedStatement ps = null;
        User u = null;

        try {
            ps = connect.prepareStatement("Select * FROM " + userTabName + " where " + userEmail + " = ? and " + userPassword + " = ?");
            ps.setString(1, email);
            ps.setString(2, password);

            ResultSet rsResultat = ps.executeQuery();

            if (rsResultat.next()) {
                u = new User();
                u.setIdUser(rsResultat.getInt(1));
                u.setFirstNameUser(rsResultat.getString(2));
                u.setLastNameUser(rsResultat.getString(3));
                u.setEmail(rsResultat.getString(4));
                u.setPassword(rsResultat.getString(5));
                u.setCompany(rsResultat.getString(6));
                u.setCreatedDateUser(rsResultat.getDate(7));
                u.setIdRole(rsResultat.getInt(8));
                //result found, means valid inputs
                return u;
            }
        } catch (SQLException ex) {
            System.out.println("Login error -->" + ex.getMessage());
            return u;
        } finally {
            //connect.close();
        }
        return u;
    } /// validation
} /// class

```

6.4.2 Code Entities - User POJO (JavaBean)

```
package com.board.business.entities;

import java.util.Date;

import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Pattern;
import javax.validation.constraints.Size;

/**
 *
 * @author spanorsa
 *
 */

@ManagedBean
@RequestScoped
public class User {
    private int idUser;
    private int idRole;
    private String firstNameUser;
    private String lastNameUser;
    private String email;
    private String password;
    private String company;
    private Date createdDateUser;

    public User() {

    }

    public User(int idUser, int idRole, String firstNameUser, String
lastNameUser, String email, String password,
String company, Date createdDateUser) {
        this.idUser = idUser;
        this.idRole = idRole;
        this.firstNameUser = firstNameUser;
        this.lastNameUser = lastNameUser;
        this.email = email;
        this.password = password;
        this.company = company;
        this.createdDateUser = createdDateUser;
    }

    public int getIdUser() {
        return idUser;
    }
}
```

```
public void setIdUser(int idUser) {
    this.idUser = idUser;
}

public int getIdRole() {
    return idRole;
}

public void setIdRole(int idRole) {
    this.idRole = idRole;
}

public String getFirstNameUser() {
    return firstNameUser;
}

public void setFirstNameUser(String firstNameUser) {
    this.firstNameUser = firstNameUser;
}

public String getLastNameUser() {
    return lastNameUser;
}

public void setLastNameUser(String lastNameUser) {
    this.lastNameUser = lastNameUser;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

public String getPassword() {
    return password;
}

public void setPassword(String password) {
    this.password = password;
}

public String getCompany() {
    return company;
}

public void setCompany(String company) {
```

```

        this.company = company;
    }

    public Date getCreatedDateUser() {
        return createdDateUser;
    }

    public void setCreatedDateUser(Date createdDateUser) {
        this.createdDateUser = createdDateUser;
    }
}

```

6.4.3 Code Entities - Class UserLoginBean

```

package com.board.business.entities.managedbean;

import java.util.List;

import javax.faces.application.FacesMessage;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
import javax.faces.context.FacesContext;
import javax.servlet.http.HttpSession;

import com.board.business.entities.Project;
import com.board.business.entities.User;
import com.board.dao.factory.DAOFactory;

@ManagedBean
@SessionScoped
public class UserLoginBean {

    private User user;
    private List<Project> projects;
    private String email;
    private String password;

    //validate login
    public String validateUsernamePassword() {
        if(email == null || email.isEmpty()){
            FacesContext.getCurrentInstance().addMessage(
                null,
                new FacesMessage(FacesMessage.SEVERITY_WARN,
                    "Please enter an email",
                    "Empty email"));
            return "signin";
        }
        else if(password == null || password.isEmpty()){
            FacesContext.getCurrentInstance().addMessage(
                null,
                new FacesMessage(FacesMessage.SEVERITY_WARN,
                    "Please enter a password",
                    "Empty password"));
            return "signin";
        }
        user= DAOFactory.getUserDAO().validate(email, password);
    }
}

```



```

        if (user != null) {
            HttpSession session = SessionUtils.getSession();
            session.setAttribute("user", user.getIdUser());
            return "main";
        } else {
            FacesContext.getCurrentInstance().addMessage(
                null,
                new FacesMessage(FacesMessage.SEVERITY_WARN,
                    "Incorrect Email and Password",
                    "Please enter correct Email and
Password"));
            return "signin";
        }
    }

    //logout event, invalidate session
    public String logout() {
        HttpSession session = SessionUtils.getSession();
        session.invalidate();
        return "signin";
    }

    public User getUser() {
        return user;
    }

    public List<Project> getProjects() {
        projects = DAOFactory.getBelongDAO().selectPro-
jectsByUser(getUser().getIdUser());
        return projects;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getRole() {
        return DAOFactory.getRoleDAO().selectById(getUser().getI-
dRole()).getNameRole();
    }
}

```

6.4.4 Code Factory de DAOs

```
package com.board.dao.factory;

import java.io.IOException;
import java.io.InputStream;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.Properties;

import com.board.dao.BelongDAO;
import com.board.dao.ProjectDAO;
import com.board.dao.RoleDAO;
import com.board.dao.SprintDAO;
import com.board.dao.StatusDAO;
import com.board.dao.UserDAO;
import com.board.dao.UserStoryDAO;

public class DAOFactory {

    private static Connection connect;

    private static final String FICHER_PROPERTIES = "config.prop-
erties";
    private static final String PROPERTY_URL = "agileboard.con-
nection.url";
    private static final String PROPERTY_DRIVER = "agileboard.con-
nection.driver";
    private static final String PROPERTY_NOM_UTILISATEUR = "agileboard.con-
nection.username";
    private static final String PROPERTY_MOT_DE_PASSE = "agileboard.con-
nection.password";

    private static String driver;
    private static String url;
    private static String username;
    private static String password;

    private static void loadConnectionParam(){

        Properties properties = new Properties();

        ClassLoader classLoader = Thread.currentThread().getCon-
textClassLoader();
        InputStream fichierProperties = classLoader.getResource-
AsStream( FICHER_PROPERTIES );

        if ( fichierProperties == null ) {
            try {
                throw new Exception( "Le fichier properties " + FI-
CHIER_PROPERTIES + " est introuvable." );
            } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }

        try {
            properties.load( fichierProperties );
        }
    }
}
```

```

        url = properties.getProperty( PROPERTY_URL );
        driver = properties.getProperty( PROPERTY_DRIVER );
        username = properties.getProperty( PROPERTY_NOM_UTILISATEUR );
        password = properties.getProperty( PROPERTY_MOT_DE_PASSE );
    } catch ( IOException e ) {
        try {
            throw new Exception( "Impossible de charger le
fichier properties " + FICHER_PROPERTIES, e );
        } catch ( Exception e1 ) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }
    }
}

public static Connection getInstance(){
    //load connection properties
    loadConnectionParam();

    if(connect == null){
        try {
            Class.forName(driver);
            connect = DriverManager.getConnection(url,
username, password);
        } catch ( SQLException | ClassNotFoundException e ) {
            e.printStackTrace();
        }
    }
    return connect;
}

//ProjectDAO instance
public static ProjectDAO getProjectDAO(){
    return new ProjectDAO(getInstance());
}

//RoleDAO instance
public static RoleDAO getRoleDAO(){
    return new RoleDAO(getInstance());
}

//SprintDAO instance
public static SprintDAO getSprintDAO(){
    return new SprintDAO(getInstance());
}

//StatusDAO instance
public static StatusDAO getStatusDAO(){
    return new StatusDAO(getInstance());
}

//UserDAO instance
public static UserDAO getUserDAO(){
    return new UserDAO(getInstance());
}

//UserStoryDAO instance
public static UserStoryDAO getUserStoryDAO(){
    return new UserStoryDAO(getInstance());
}

```

```
//BelongDAO instance
public static BelongDAO getBelongDAO(){
    return new BelongDAO(getInstance());
}

}
```

6.4.5 Connexion vers la Base de données

```
agileboard.connection.url=jdbc:mysql://localhost:3306/board_agile
agileboard.connection.driver=com.mysql.jdbc.Driver
agileboard.connection.username=root
agileboard.connection.password=
```

6.5 Contrôleur

Dans une application Web à base de JSF le fichier **web.xml** fait office de **contrôleur** (la servlet).

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee" xsi:schemaLoca-
tion="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/ja-
vae/web-app_3_0.xsd" version="3.0">

    <context-param>
        <param-name>javax.faces.CONFIG_FILES</param-name>
        <param-value>/WEB-INF/faces-config.xml</param-value>
    </context-param>

    <context-param>
        <param-name>javax.faces.boardAgile</param-name>
        <param-value>Development</param-value>
    </context-param>

    <!-- Servlet/Controller -->

    <servlet>
        <servlet-name>Faces Servlet</servlet-name>
        <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>

    <servlet-mapping>
        <servlet-name>Faces Servlet</servlet-name>
        <url-pattern>*.xhtml</url-pattern>
    </servlet-mapping>

</web-app>
```

6.6 Code pour les Tests

6.6.1 Code Teste JUnit - User DAO

```
package com.board.tests.dao;

import java.util.List;

import org.junit.Assert;
import org.junit.Rule;
import org.junit.Test;
import org.junit.rules.ExpectedException;

import com.board.business.entities.User;
import com.board.dao.factory.DAOFactory;

public class UserDAOTest {

    @Rule
    public ExpectedException thrown = ExpectedException.none();

    @Test
    public void testAddUser() {

        User user = new User();
        //Developer role
        user.setIdRole(1);
        //User mail
        user.setEmail("toto@example.com");
        //User firstname
        user.setFirstNameUser("Eric");
        //User lastName
        user.setLastNameUser("DUPONT");
        //User company
        user.setCompany("ORANGE");
        //User pass
        user.setPassword("#####");

        //DB user insertion
        DAOFactory.getUserDAO().insert(user);

        //Retrieve all users in DB
        List<User> usersDBList = DAOFactory.getUserDAO().selectAll();

        //Test results

        //DB table row size
        Assert.assertEquals(1, usersDBList.size());
        //check mail (expected, result)
        Assert.assertEquals("toto@example.com", usersDBList.get(0).getEmail());
        //check company (expected, result)
        Assert.assertEquals("ORANGE", usersDBList.get(0).getCompany());
        //check firstname (expected, result)
        Assert.assertEquals("Eric", usersDBList.get(0).getFirstNam-
eUser());
        //check lastname (expected, result)
        Assert.assertEquals("DUPONT", usersDBList.get(0).getLastNam-
eUser());
        //check password (expected, result)
```

```

Assert.assertEquals("#####", usersDBList.get(0).getPass-
word());

//check password (expected, result)
Assert.assertEquals(1, usersDBList.get(0).getIdRole());
//check id (expected, result)
Assert.assertEquals(1, usersDBList.get(0).getIdUser());

}

@Test
public void testgetUserById() {

    //Retrieve a user by ID in DB
    User user = DAOFactory.getUserDAO().getUserById(1);

    //Test results
    //check role (expected, result)
    Assert.assertEquals(1, user.getIdUser());
    //check mail (expected, result)
    Assert.assertEquals("toto@example.com", user.getEmail());
    //check company (expected, result)
    Assert.assertEquals("ORANGE", user.getCompany());
    //check firstname (expected, result)
    Assert.assertEquals("Eric", user.getFirstNameUser());
    //check lastname (expected, result)
    Assert.assertEquals("DUPONT", user.getLastNameUser());
    //check password (expected, result)
    Assert.assertEquals("#####", user.getPassword());
    //check role (expected, result)
    Assert.assertEquals(1, user.getIdRole());

}

@Test
public void excptMailNotUnique(){

    User user = new User();
    //Developer role
    user.setIdRole(1);
    //User mail
    user.setEmail("toto@example.com");
    //User firstname
    user.setFirstNameUser("Elois");
    //User lastName
    user.setLastNameUser("DARIN");
    //User company
    user.setCompany("SOGETI");
    //User pass
    user.setPassword("#####");

    thrown.expect(Exception.class);
    thrown.expectMessage("Duplicate entry 'toto@example.com' for key
'email'");

    //DB user insertion
    DAOFactory.getUserDAO().insert(user);

}
}

```

6.6.2 Script pour l'insertion de données de test dans la BD

```
--
-- Dumping data for table `projects`
--

LOCK TABLES `projects` WRITE;
/*!40000 ALTER TABLE `projects` DISABLE KEYS */;
INSERT INTO `projects` VALUES (1,'FastProject','2016-07-21','2016-12-21'),(2,'eCommercy','2016-05-11','2017-04-09');
/*!40000 ALTER TABLE `projects` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Dumping data for table `roles`
--

LOCK TABLES `roles` WRITE;
/*!40000 ALTER TABLE `roles` DISABLE KEYS */;
INSERT INTO `roles` VALUES (1,'Developer'),(2,'Product Owner'),(3,'SCRUM master');
/*!40000 ALTER TABLE `roles` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Dumping data for table `users`
--

LOCK TABLES `users` WRITE;
/*!40000 ALTER TABLE `users` DISABLE KEYS */;
INSERT INTO `users` VALUES (1,'Eric','DUPONT','toto@example.com','#####','ORANGE','2016-12-09 00:00:00',1),
(27,'Tonio','PALUD','our','yyy','Sogeti HT','2016-12-09 00:00:00',1),
(28,'Nico','LAUPOU','bobo@gigi.fr','gigi','Thales','2016-12-09 23:15:29',2);
/*!40000 ALTER TABLE `users` ENABLE KEYS */;
UNLOCK TABLES;
```

CHAPITRE 7 - Déploiement

7.1 Le diagramme de déploiement

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.

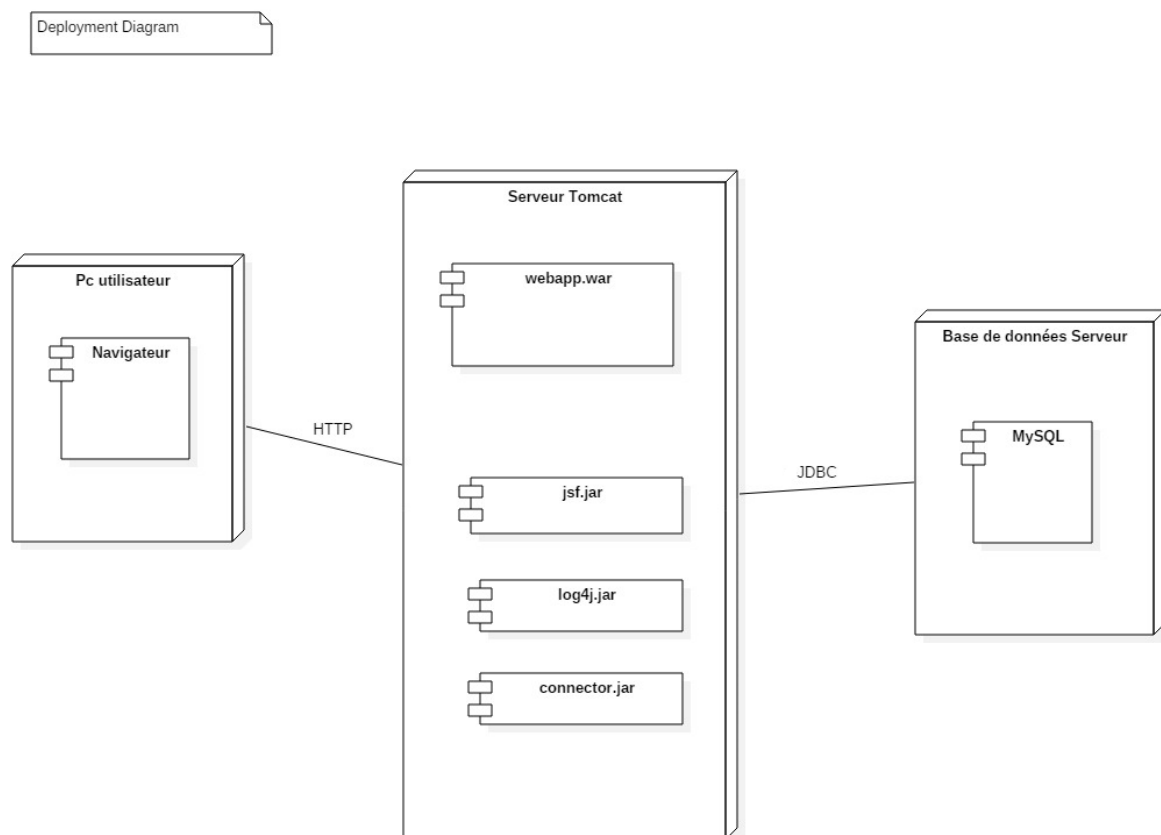
Les éléments utilisés par un **diagramme de déploiement** sont principalement les **noeuds**, les **composants**, les **associations** et les **artefacts**.

Les **noeuds** (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.



CHAPITRE 8 - Conclusion et perspectives

Le développement d'un projet informatique est un processus évolutif, assimilable à de la recherche.

Je suis très fière d'avoir eu la chance de travailler avec une équipe très qualifiée qui a partagé ses compétences et son envie de faire toujours mieux : de mieux organiser le travail et de répondre aux besoins des utilisateurs.

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

Malheureusement, la durée du stage était trop courte pour terminer l'implémentation de ce projet. C'est la raison pour laquelle j'ai décidé de m'investir davantage et d'accepter de continuer mon stage de conception/développement du Board Agile.

Cela me permettra notamment de poursuivre la seconde partie du projet : « la mise en place du système de réalité virtuelle pour la détection des mouvements et interaction avec l'application web ».

CHAPITRE 9 - Annexes

9.1 Correspondances Projet/Reac






COMPETENCES DU REAC	CORRESPONDANCES DANS LE PROJET
Développer des composants d'interface	
<i>Maquetter une application</i>	UML (Diagrammes de Cas d'Utilisation, d'activité et de navigation) avec StarUml Maquettes de l'IHM avec Pencil Evolus
<i>Développer une interface utilisateur</i>	JSF
<i>Développer des composants d'accès aux données</i>	DAO Java
<i>Développer des pages web en lien avec une base de données</i>	Application web n-tiers (boundary -> controller-> entity -> DAO -> database)
Développer la persistance des données	
<i>Concevoir une base de données</i>	Les règles de gestion UML (DCL, MCD, MLD, MPD) avec StarUml. Schéma de base de données avec MySQL WorkBench
<i>Mettre en place une base de données</i>	SQL (LDD) : Création de la BD, Création des tables et mise en place des contraintes (FK) avec phpMyAdmin
<i>Développer des composants dans le langage d'une base de données</i>	Gestion de la persistance avec un Entity (ORM)
<i>Utiliser l'anglais dans son activité professionnelle en informatique</i>	Diagrammes UML, maquettes, développement
Développer une application n-tiers	
<i>Concevoir une application</i>	UML (Diagramme de Séquence Détaillé, ...)
<i>Collaborer à la gestion d'un projet informatique</i>	Gestion du projet avec la méthode SCRUM et avec Taïga
<i>Développer des composants métier</i>	Classes « entités » avec Java (POJO)
<i>Construire une application organisée en couches</i>	Application du design pattern ECB (cf les diagrammes de séquences détaillés)
<i>Développer une application de mobilité numérique</i>	
<i>Préparer et exécuter les plans de tests d'une application</i>	Tests unitaires (JUnit)
<i>Préparer et exécuter le déploiement d'une application</i>	UML (Diagramme de Déploiement) avec StarUml Installation de la BD Test

9.2 Liste de mots-clés

Abréviations et définitions utilisées :



Mot clé	Description
UML	Unified Modeling Language
Merise	est une méthode d'analyse, de conception et de gestion de projet informatique.
Java EE	Java Enterprise Edition
ORM	Object Relational Mapping
API	Application Programming Interface (Bibliothèques)
IDE	Integrated Development Environment
HTML	Hyper Text Markup Language (Langage de balisage hypertexte)
CSS	Cascading Style Sheet (Feuilles de style en cascade)
HTTP	Hyper Text Transfert Protocol: Protocole de communication entre 2 logiciels (un client et un serveur)
JS	JavaScript
AJAX	Asynchronous JavaScript and Xml ⁷
URL	Uniform Resource Locator
CRUD	Concerne le LMD (Langage de manipulation de données). C : create R : read U : update D : delete
IHM	Interface Home Machine
DCL	Diagramme de classe
MCD	Modèle conceptuel de données
MLD	Modèle logique de données
MPD	Modèle physique de données
SGBDR	Système de Gestion de Bases de Données Relationnelles
SQL	Structured Query Language (Langage de Requête Structurée)
LDD	Data Definition Language (Langage de Définition de Données)
LMD	Data Manipulation Language (Langage de manipulation de données)
JDBC	Java DataBase Connectivity
JDK	Java Development Kit
JRE	Java Runtime Environment
JSF	Java Server Faces
DAO	Data Access Object
IDE	Integrated Development Environment (environnement de développement)
ECB	Entity – Control- Boundary
AGL	Atelier de génie logiciel

9.3 Les langages

	<p>UML= Unified Modeling Language se définit comme un langage de modélisation graphique et textuel destiné à comprendre et décrire des besoins, spécifier et documenter des systèmes, esquisser des architectures logicielles, concevoir des solutions et communiquer des points de vue.⁸</p>
<p>HTML CSS</p> 	<p>HTML 5 = HyperText Markup Language 5 (langage de balisage hypertexte) est le langage utilisé pour décrire et définir le contenu d'une page web.⁹</p> <p>CSS 3 = Cascading Style Sheets (feuilles de style en cascade) sont utilisées pour décrire l'apparence du contenu Web.¹⁰ langage utilisé pour décrire la mise en page et le style d'un document écrit en HTML</p>
 <p>JavaScript</p>	<p>JS = JavaScript est un langage de script léger, orienté objet, principalement connu comme le langage de script des pages web.¹¹</p>
	<p>Java est un langage de programmation informatique orienté objet multiplateformes (le programme créé, fonctionnera automatiquement sous Windows, Mac, Linux, etc.)¹²</p>
	<p>Java EE = Java Enterprise Edition (J2EE) est dédiée à la réalisation d'applications pour entreprises.¹³</p>

9.4 Les outils

	<p>StarUML est un logiciel de modélisation UML en open source.¹⁴</p>
	<p>MySQL Workbench est un logiciel de gestion et d'administration de bases de données MySQL.¹⁵</p>
	<p>Notepad++ est un éditeur de code source qui prend en charge plusieurs langages.¹⁶</p>
	<p>phpMyAdmin est une application Web de gestion pour les systèmes de gestion de base de données MySQL et MariaDB réalisée en PHP et distribuée sous licence publique générale.^{17,18}</p>
	<p>Pencil Evolus est conçu dans le but de fournir un outil de prototypage gratuit et open-source GUI que les gens peuvent facilement installer et utiliser pour créer des maquettes dans les plates-formes de bureau populaires.¹⁹</p>
	<p>Taiga est une plate-forme de gestion de projet gratuite et open source pour les projets développés en méthode agile.²⁰</p>

	<p><i>PowerAMC</i> est un logiciel de conception créé par la société SDP, qui permet de modéliser les traitements informatiques et leurs bases de données associées.²¹</p>
	<p><i>Eclipse</i> est un IDE, <i>Integrated Development Environment</i> (EDI environnement de développement intégré en français), un logiciel qui simplifie la programmation en proposant un certain nombre de raccourcis et d'aide à la programmation.²²</p>

CHAPITRE 10 - Bibliographie et Webographie

- ¹ <https://fr.wikipedia.org/wiki/Sogeti>.
- ² Frédéric Fillon, Cours Intro Méthodes Agile
- ³ <http://ineumann.developpez.com/tutoriels/merise/initiation-merise/>
- ⁴ Christian Soutou, UML 2 pour les bases de données, Eyrolles, 2007.
- ⁵ <http://www.enseignement.polytechnique.fr/profs/informatique/Julien.Cervelle/eclipse/>
- ⁶ Pascal Buguet, Cours Java - JSP
- ⁷ <https://developer.mozilla.org/fr/docs/AJAX>
- ⁸ Pascal Roques, UML 2, Modéliser une application web, Eyrolles, 2008
- ⁹ <https://developer.mozilla.org/fr/docs/Web/HTML>
- ¹⁰ <https://developer.mozilla.org/fr/docs/Web/CSS>
- ¹¹ <https://developer.mozilla.org/fr/docs/Web/JavaScript>
- ¹² <https://openclassrooms.com/courses/apprenez-a-programmer-en-java>
- ¹³ <https://www.jmdoudoux.fr/java/dej/chap-j2ee-javaee.htm>
- ¹⁴ <https://fr.wikipedia.org/wiki/StarUML>
- ¹⁵ https://fr.wikipedia.org/wiki/MySQL_Workbench
- ¹⁶ <https://notepad-plus-plus.org/fr/>
- ¹⁷ <https://fr.wikipedia.org/wiki/PhpMyAdmin>
- ¹⁸ <https://www.phpmyadmin.net/>
- ¹⁹ <http://pencil.evolus.vn/>
- ²⁰ <https://taiga.io/>
- ²¹ <https://fr.wikipedia.org/wiki/PowerAMC>
- ²² <http://www.eclipse.org/>
- ²³ <http://www.mysql.com/>
- ²⁴ <http://sqlpro.developpez.com/>
- ²⁵ <http://www.oracle.com/technetwork/java/javaee/documentation/index.html>
- ²⁶ Christian Soutou, Programmer avec MySQL, Eyrolles, 2015
- ²⁷ Christian Soutou avec la contribution de Frédéric Brouard, « UML 2 pour les bases de données », Eyrolles, 2012.
- ²⁸ Pascal Buguet, Cours Java
- ²⁹ Pascal Buguet, Cours UML
- ³⁰ <http://www.w3schools.com/sql/>