

Document d’implémentation

**Membres du groupe**

**NOUPELAH VLADIMIR**

**NYASSA NOAH**

**TANWI NKIAMBOH**

**KOUAYEP PATERNE**

**TCHUITSE DJOMO**

TABLE DES MATIERES

[I. PARTIES IMPLEMENTEES 2](#_Toc516815167)

[L’interface graphique 3](#_Toc516815168)

[La conversion 5](#_Toc516815169)

[La structuration 5](#_Toc516815170)

[II. OUTILS UTILISES 5](#_Toc516815171)

# PARTIES IMPLEMENTEES

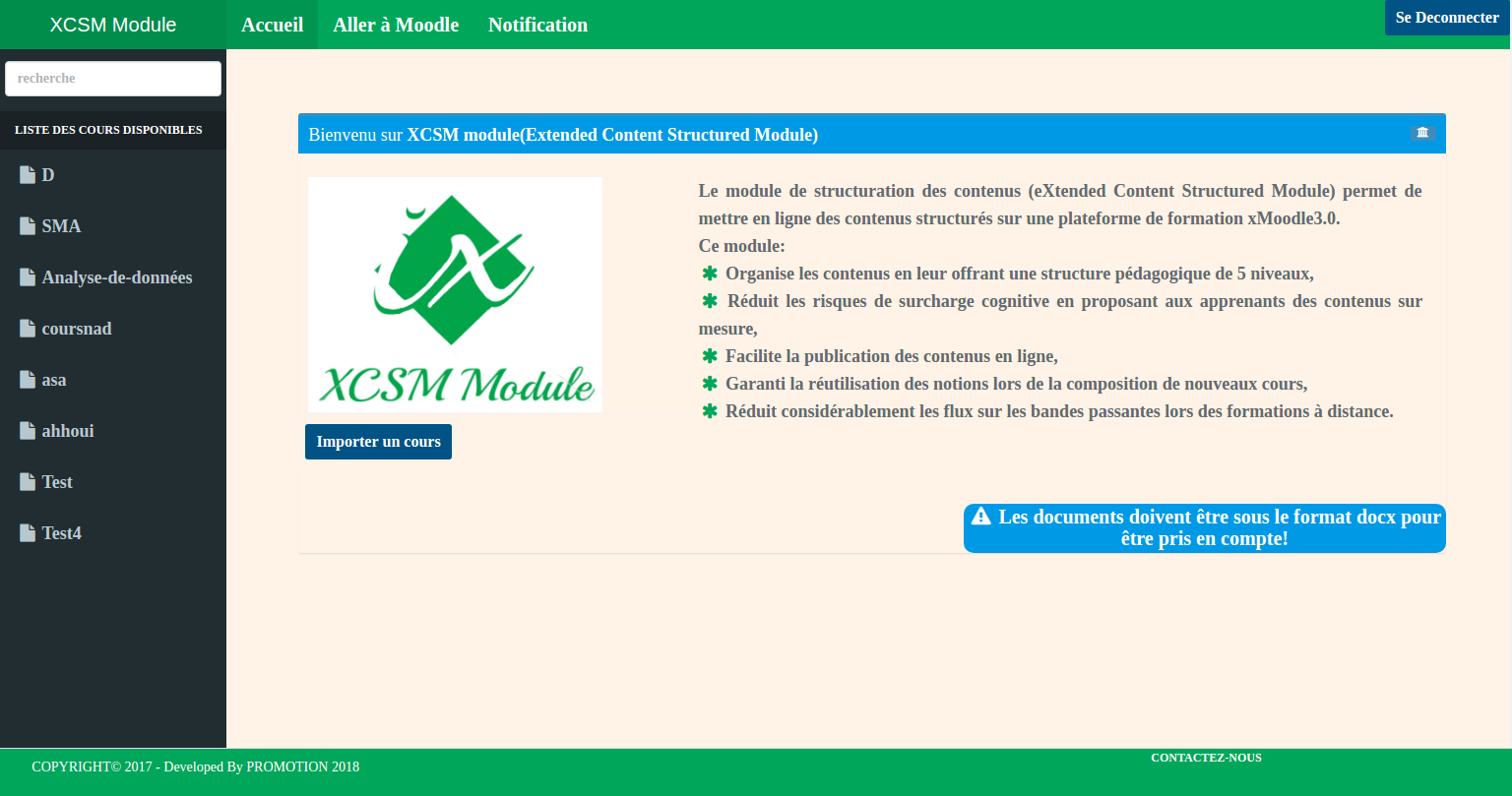
## L’interface graphique

L’ihm de l’application a été totalement redéfinie, et repensée. Un nouveau logo a également été pensé et créé, et inséré à l’interface. De façon générale, l’application a 4 pages principales à savoir :

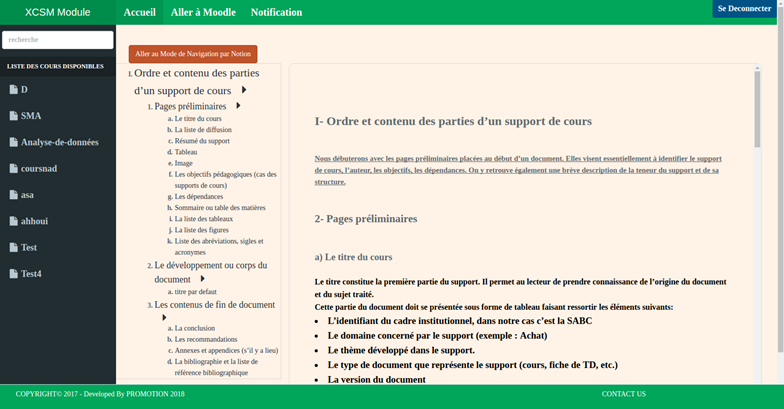
* La page de connexion : c’est elle qui permet de recueillir les informations d’identification afin d’accorder ou non l’accès à l’application



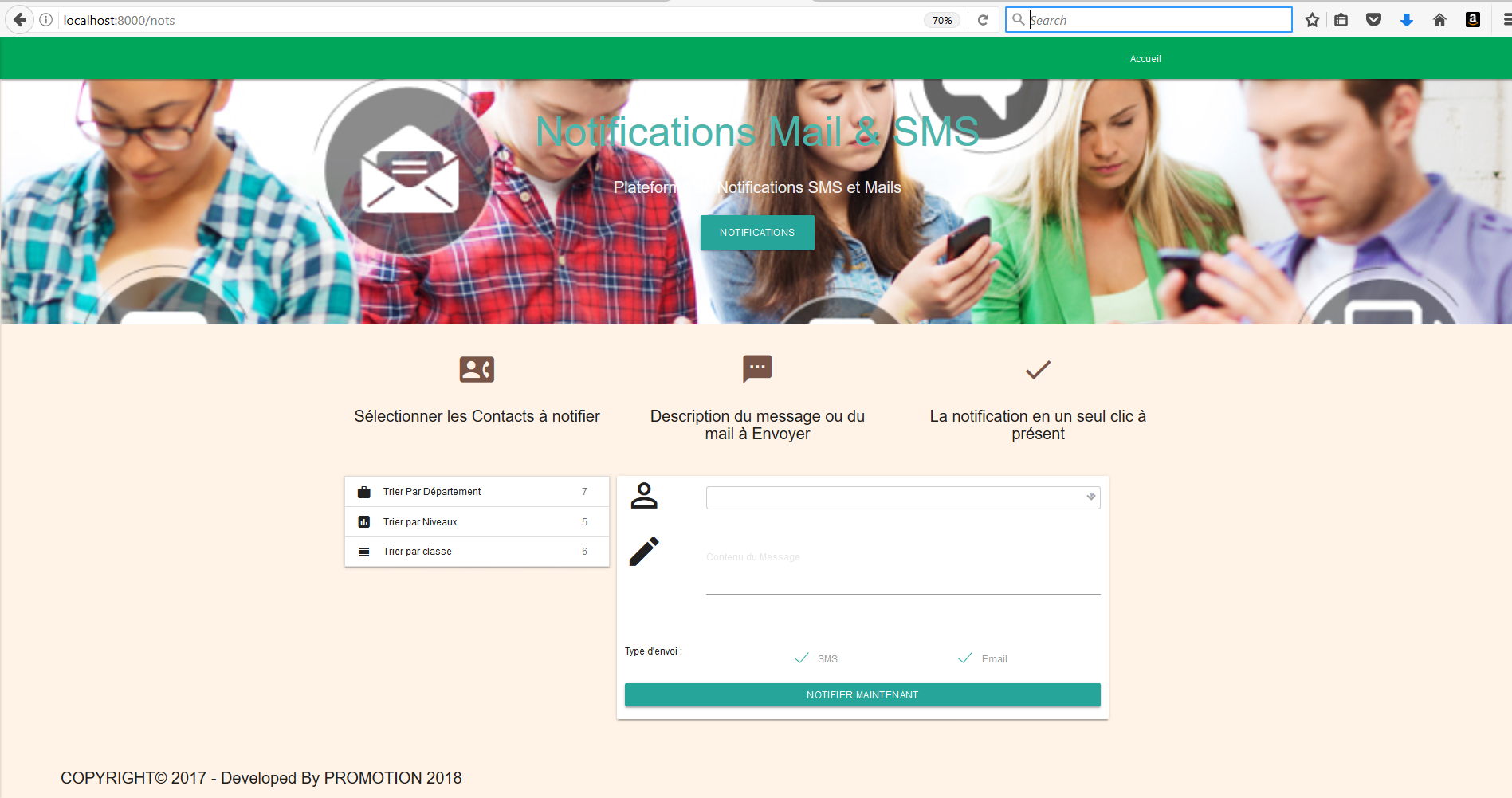
* L’accueil : après une identification réussie on arrive sur la page d’accueil qui présente les différents menus et la hiérarchie



* La page de cours : c’est la page qui permet de lire et de parcourir un document importé et structuré



* La page de notification : c’est à partir de cette page qu’on effectue les notifications. Il s’agira de choisir les personnes à notifier, de saisir le message à envoyer et d’effectuer l’envoi soit par message soit par mail.



## La conversion

Il s’agit du processus de « parsage ». La conversion ici consiste à partir d’un document au format PDF pour un autre au format fo, contenant exactement les même informations que le document de départ. Ce nouveau format obtenu est celui exploitable par le module de structuration.

## La structuration

Ce module permet de lire le fichier xslfo produit par le module de structuration afin d’en ressortir la hiérarchie sous la forme :

* Partie
* Chapitre
* Paragraphe
* Notion

Ainsi le document pourra donc être parcouru suivant ces grandes parties, ce qui facilite la lecture et l’apprentissage.

# OUTILS UTILISES

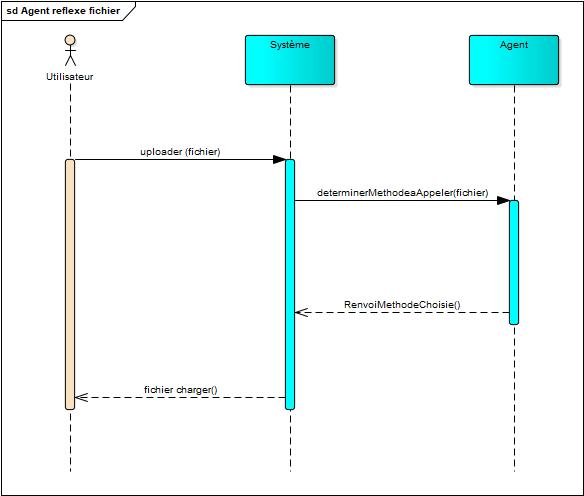
|  |  |  |  |
| --- | --- | --- | --- |
| Outil/Technologie | Version | Description | Rôle dans l’implémentation |
| DOCX4J | 3.3.3 | Bibliothèque java permettant la manipulation des fichiers au format docx | Utilisé lors du parsage pour convertir du format docx au format xslfo |
| JAVA SE  D:\ENSP\4GI\S2\SMA\xcsm\logos\main-qimg-fae127a58a86bb853f1f9cbab4a91c7b.png | 8.74 | Langage de programmation permettant de développer aussi bien des applications de bureau que sur des serveurs | Langage utilisé pour programmer les processus de conversion et de structuration |
| Laravel  D:\ENSP\4GI\S2\SMA\xcsm\logos\laravel-5.4-blog1.jpg | 5.4 | Framework PHP facilitant la création d’applications en PHP | Utilisé pour réaliser les interfaces de présentation, les pages de l’application |
| PHP  D:\ENSP\4GI\S2\SMA\xcsm\logos\index.png | 7 | langage de programmation côté serveur | C’est le support du Framework laravel |
| Javascript ES  D:\ENSP\4GI\S2\SMA\xcsm\logos\illu_tout-sur-le-javascript.png | 6 | langage événementiel permettant la dynamisation des pages côté client | Utilisé pour définir les animations sur les pages |
| Bootstrap  D:\ENSP\4GI\S2\SMA\xcsm\logos\bootstrap.gif | 3 | Framework css facilitant la présentation des pages | Utilisé pour réaliser l’ihm générale de l’application (agencements des couleurs, des blocks de contenus, etc.) |
| Materialize css  D:\ENSP\4GI\S2\SMA\xcsm\logos\materialize.png |  | Framework css facilitant la présentation des pages | Utilisé pour réaliser l’ihm du module de structuration |
| Moodle  D:\ENSP\4GI\S2\SMA\xcsm\logos\moodle.ai-converted.png | 3.1 | plateforme utilisée pour la gestion des contenus à caractère éducatif | Utilisé comme support d’intégration et hébergeur final de l’application |
| PdfBox | 3.1 | Bibliothèque java permettant la manipulation de documents au format PDF | Utilisé lors du parsage pour convertir du format PDF au format xslfo |

# INPLEMENTATION DES AGENTS

Pour faciliter l’apprentissage, nous allons implémenter quelques agent dans notre application. Pour se faire, nous allons implémenter des agents reflex simples et des agents basés sur les buts.

1. Les agents reflex simple

Un agent reflex simple comme son nom l’indique exécute une action en fonction de l’action courante. Ainsi lorsque nous importons un fichier que l’application doit parser, on peut mettre sur pied un agent qui détermine la nature la nature du fichier ( son extension ). Ainsi, si à l’avenir l’application doit prendre en compte dix (10) cathégories de fichier, il ne serait pas nécessaire .d’afficher dix boutons pour chacune des catégorie. On entre juste le fichier et l’agent détermine sa nature (son extension) et effectue l’action appropriée. Son comportement peut être décrit par le diagramme de séquence suivant :



Comme autre agent reflex simple, nous comptons mettre sur pied un agent permettant de faciliter les recherches dans l’application. Ainsi, connaissant juste une notion dans un cours, l’agent nous donne tous les cours, tous les enseignants… en rapport avec cette notion

Nous pouvon également implémenter un agent qui réflex qui enverra automatiquement les notifications par mail à tous les étudiants concernés par le cours lorsqu’une notification est faite au lieu de le faire manuellement comme c’est le cas maintenant.

1. Les agents basés sur les buts

Le but ici peut être de proposer aux apprenants des cours en rapporta avec ceux sur lesquels ils mettent un certain nombre de temps. Pour se faire nous pouvons mettre sur pied un SMA contitué d’un agent reflex et d’un agent basé sur les buts. Lorsqu’un étudiant dépasse le temps limite sur une notion, l’agent reflex déclenche l’agent orienté sur les but qui recherchera tous les autres cours qui contiennent cette notion. Ce second agent se doit d’être rationnel. Le comportement de ce SMA peut être décrit par le le diagramme de séquence suivant :

