



Gestion Login Utilisateur

Projet de fin d'études, choix
technologiques et analyse

07/02/2017

Table des matières

1.	Introduction :	1
2.	Choix technologiques :	2
2.1.	Back-end :	2
2.2.	Base de données :	3
2.3.	Front-end :	4
3.	Diagramme d'architecture :	5
4.	Outils utilisés :	6
4.1.	Git-GitHub :	6
4.2.	Trello :	6
4.3.	Visual Studio Code :	6
5.	Analyse du projet :	7
5.1.	Objet du marché :	7
5.1.1.	Fonctionnalités :	7
5.1.2.	Contraintes :	7
5.2.	Diagramme des cas d'utilisation :	8
	Acteurs :	8
	Description des cas d'utilisation :	9
5.3.	Base de données :	11
5.4.	Organisation du back-end Node.js :	12
5.4.1.	Organisation du code :	12
5.4.2.	Organisation des modules Node.js :	13
5.5.	IHM :	14
	Page d'accueil :	14
	Feuille de login :	14
6.	Conclusion :	14
7.	Bibliographie :	15

Chef de projet : Cedric Tavernier

Membres : Degrève Olivier, Denuit Maxime, Dubois Corenthin

1. Introduction :

Nous allons vous présenter les différents choix technologiques et les outils choisis pour l'outil de gestion demandé par l'Institut Paul Lambin.

Nous vous expliquerons ensuite pourquoi nous avons pris le parti de faire un Back-end en Node.js, un Front-end en JavaScript, HTML 5, CSS3 et JQuery et notre choix d'utiliser une base de données en PostgreSQL.

Nous avons décidé d'utiliser les outils suivants pour notre projet : Git-GitHub pour le partage de données, Trello pour l'organisation du projet et Visual studio Code pour l'implémentation.

2. Choix technologiques :

2.1. Back-end :

Nous utiliserons un back-end en Node.js. L'avantage de cette technologie est que nous avons les ressources nécessaires dans notre groupe pour l'utiliser et que la documentation est complète et libre d'accès sur internet. Une autre chose intéressante à propos de Node.js est sa grande modularité, ce qui nous permettra d'adapter le produit tout en maîtrisant les coûts. Il existe beaucoup de bibliothèques externes permettant d'effectuer des tâches pour un back-end - que ce soit des bibliothèques serveurs, de requêtes HTTP/HTTPS ou autres. Ce même genre de bibliothèque existe aussi pour le Front-end afin d'avoir une interface simple à mettre en place (Exemple : AngularJS) ou d'avoir des fonctions en temps réel grâce aux WebSockets de Node.js.

Si nous devons comparer Node.js au Java pour le Back-end, il est clair que Node.js présente clairement plusieurs avantages. Premièrement, l'utilisation de Node.js permet le développement d'une application de type « full-stack » (c'est-à-dire que l'entièreté du code est en JavaScript) - ce qui permet de ne pas avoir de « middleware » pour convertir d'éventuelles données transmises par le Front-end. Deuxièmement, Node.js permet aussi d'éviter les problèmes de conversion et de complexité du programme. En effet, si nous avions utilisé le Java cela aurait pu causer des problèmes de conversion et il nous aurait également fallu convertir le JSON avec Genson. Troisièmement, Node.js a l'avantage d'être plus rapide que Java pour faire de l'Input/Output et ne bloque pas d'éventuelles requêtes supplémentaires. En effet, il y a une bonne gestion de la concurrence en Node.js, ce qui permet de pouvoir effectuer des requêtes Input/Output en parallèle. Bien sûr, il est possible d'implémenter de la concurrence en Java mais cela requiert plus de temps et engendre potentiellement plus de problèmes s'ils sont mal implémentés. Quatrièmement, Node.js offre également l'avantage d'avoir une très bonne scalabilité. En effet, on peut déployer rapidement une application Node.js pour une entreprise - que ce soit une application web ou mobile. Cinquièmement, Node.js présente aussi l'avantage de présenter de très bonnes performances notamment en ce qui concerne la rapidité d'exécution des requêtes. Toutefois, il est à noter que Node.js sert principalement à effectuer des tâches peu complexes. L'application ne requérant pas des calculs complexes, nous pouvons donc utiliser le Node.js. Sixièmement, au niveau de la structure, Node.js peut utiliser des modules supplémentaires afin d'intégrer plus de fonctionnalités. Ces modules se situeront dans un dossier « node_modules » à la racine du dossier contenant le projet. Sinon, il est préférable d'avoir quelques dossiers avec les différentes parties du code et d'éviter les sous-dossiers pour ne pas avoir de chemins trop longs.

Chef de projet : Cedric Tavernier

Membres : Degrevé Olivier, Denuit Maxime, Dubois Corenthin

2.2. Base de données :

Nous avons choisi de faire une base de données en SQL et non en NoSQL. Durant la rencontre avec le client, ce dernier nous a demandé d'implémenter un outil de gestion. Or, un tel outil implique de nombreuses relations entre les différentes données. Ceci explique donc notre choix de travailler en SQL. Nous n'envisageons pas NoSQL car l'implémentation d'une base de données relationnelle dans ce langage est possible mais complexe.

Nous faisons le choix du SQL, nous avons pris une base de données PostgreSQL, ce qui présente l'énorme avantage de pouvoir gérer un grand nombre de données. En outre, un support important existe et PostgreSQL étant une technologie libre, les coûts pour la société seront plus faibles (car il n'est pas nécessaire d'avoir une licence commerciale). Nous avons écarté le choix de PgSQL car ce dernier ne convient que pour de petits sites.

Chef de projet : Cedric Tavernier

Membres : Degrève Olivier, Denuit Maxime, Dubois Corenthin

2.3. Front-end :

Comme langage de programmation pour le front-end nous avons décidé d'utiliser le JavaScript pour plusieurs raisons.

Premièrement, nous avons décidé d'utiliser le JavaScript car il permet d'agir directement et ne doit pas attendre que les serveurs envoient une réponse. Ceci accélère l'ouverture des sites web sur les navigateurs des clients. De plus, le JavaScript ne nécessite pas un programme d'interprétation comme par exemple avec Flash Player. En outre, le JavaScript n'occupe pas une grande place sur les disques des sites web. Nous avons aussi décidé d'utiliser une bibliothèque JavaScript JQuery, ce qui nous assurera une compatibilité multi-navigateurs, une simplification et une normalisation des scripts.

Deuxièmement, comme nous avons décidé d'utiliser le Node.js comme langage back-end car cela nous permet de travailler en « full-stack », c'est-à-dire de pouvoir programmer toute l'application dans un même langage et ne pas devoir utiliser un langage intermédiaire pour rendre compatibles nos données entre les différents langages.

Pour la mise en forme du site nous avons décidé d'utiliser l'HTML5 (HyperText Markup Language 5) et le CSS3 (Cascading Style Sheet 3).

Nous avons choisi la dernière version de ces deux langages car ceux-ci nous permettent d'en utiliser toutes les dernières fonctionnalités.

Toutefois, il existe un inconvénient mineur et dont il faudra tenir compte : certains navigateurs ne prennent pas en charge la dernière version de l'HTML5. Pour régler ce problème, nous devons donc coder en fonction du navigateur le moins « up-to-date ».

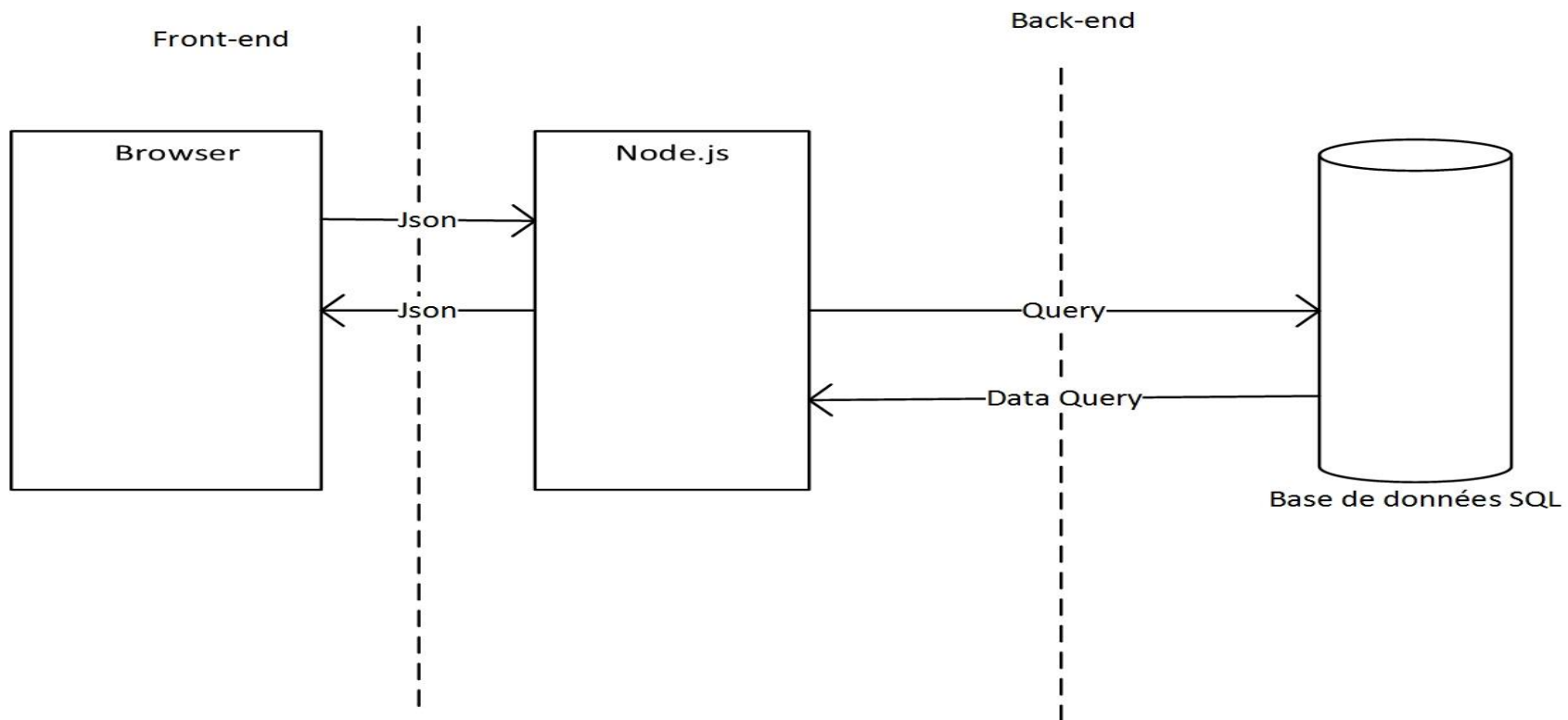
Pourquoi n'allons-nous pas utiliser Bootstrap ?

Il nous semble intéressant de pouvoir créer de l'HTML / CSS « fait maison », car cela nous permettra d'utiliser facilement notre HTML/CSS dans nos codes JAVASCRIPT, avec des balises/identifiants qui sont clairs et lisibles. Bref, il nous a semblé plus intéressant de créer nous même un site responsif et plus judicieux de ne pas utiliser un Template tout fait, comme proposé par Bootstrap.

Chef de projet : Cedric Tavernier

Membres : Degrève Olivier, Denuit Maxime, Dubois Corenthin

3. Diagramme d'architecture :



4. Outils utilisés :

Pour mener à bien ce projet, trois outils distincts vont être utilisés : Git-GitHub, Trello et Visual Studio Code.

4.1. Git-GitHub :

Git-GitHub en est le premier. D'une part, l'hébergement de projets et de dépôts, est pris en charge par le site web GitHub. Ce site web facilite la collaboration sur des projets entre plusieurs intervenants. D'autre part, nous avons un logiciel qui permet aux utilisateurs de sauvegarder différentes versions de fichiers durant le cycle de vie d'un projet sur leur pc, à savoir Git. Cette application permet également aux utilisateurs d'envoyer leurs différentes versions sur leur compte GitHub ou sur un projet partagé. Ensuite, il y aura moyen de « merger » les fichiers de tous les collaborateurs du projet en un travail unique.

Pourquoi s'être tourné vers Git-GitHub ?

Nous nous sommes tournés vers Git-GitHub car elle est considérée actuellement comme étant la plateforme d'hébergement de référence. Elle réunit plus de 5,8 millions de développeurs actifs et la société lève en permanence des fonds importants pour améliorer ses services mais aussi pour y ajouter de nouveaux outils. De plus, GitHub va également nous permettre de communiquer directement avec Trello, notre second outil.

4.2. Trello :

Trello est un logiciel permettant d'organiser nos différentes tâches, de les assigner à chaque développeur et de visualiser l'évolution de notre projet. Grâce à la fonctionnalité GitHub Power-Up de Trello, il nous est permis de lier notre compte GitHub à Trello. Nous pourrions ainsi joindre un « commit », « un pull-request » et en voir son état d'avancement.

Mais pourquoi le logiciel Trello et pas un autre ?

Nous avons choisi Trello car ce logiciel comptabilise plus de 10 millions d'utilisateurs dans le monde et est utilisé par des entreprises renommées, telles que PayPal, Google, Adobe, ... Le logiciel est adapté aux mobiles, point important pour visualiser à n'importe quel moment les différentes tâches qu'il nous reste à réaliser.

4.3. Visual Studio Code :

Nous avons pris comme décision de développer sur Visual Studio Code car il est présent sur tous les systèmes d'exploitation et est gratuit, il possède un système de terminal intégré et est disponible chez le client. C'est donc un outil extensible et personnalisable.

Chef de projet : Cedric Tavernier

Membres : Degrève Olivier, Denuit Maxime, Dubois Corenthin

5. Analyse du projet :

5.1. Objet du marché :

L'objet du marché consiste à développer un outil de gestion des logins.

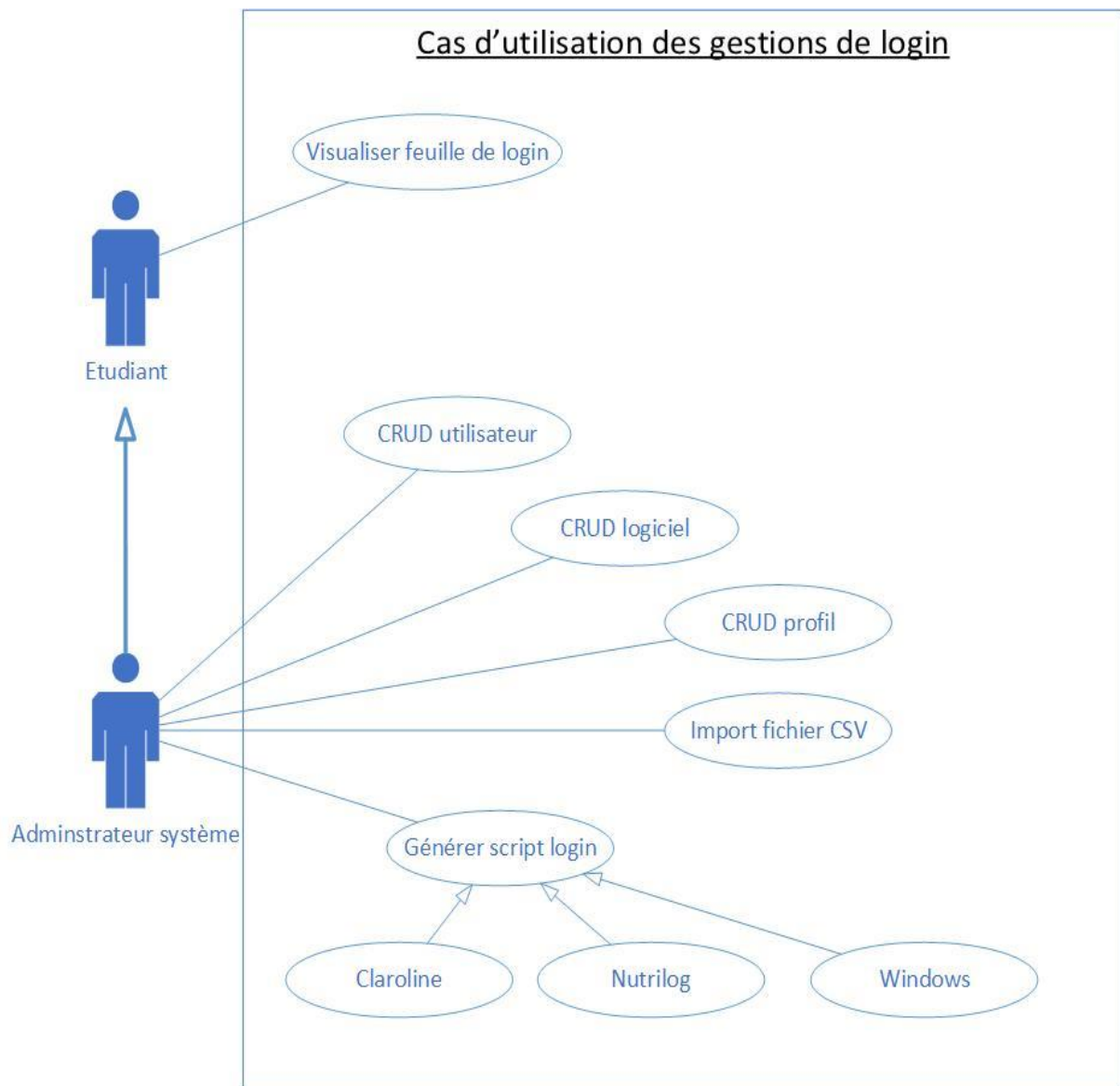
5.1.1. Fonctionnalités :

- La gestion des utilisateurs et des profils
- La gestion des logiciels
- La génération des scripts de login
- La consultation de feuilles de login

5.1.2. Contraintes :

- Séparation client et serveur
- Format du fichier CSV défini par le client (syntaxe imposée)
- Seuls les étudiants ayant un matricule peuvent consulter en ligne leur feuille de login
- Les professeurs et personnes invitées devront aller chercher leur feuille de login chez les administrateurs systèmes qui généreront celle-ci.
- Architecture adaptable pour mobile

5.2. Diagramme des cas d'utilisation :



Acteurs :

Acteurs de l'application :

- Administrateurs systèmes
- Etudiants

Acteurs indirectement concernés par l'application :

- Personnes invitées
- Professeurs

Description des cas d'utilisation :

1. Visualiser la feuille de login

Cette fonctionnalité est d'application pour les étudiants et les administrateurs systèmes. À partir de son matricule d'étudiant, chaque étudiant ne pourra avoir accès qu'aux informations qui correspondent à ce matricule (affichage des informations personnelles et des différents logiciels auxquels l'étudiant a accès).

2. CRUD utilisateur

Cette fonctionnalité va permettre à l'administrateur système à la fois de créer, lire, mettre à jour mais aussi de supprimer les informations d'un utilisateur (professeur, invité).

Lors de l'insertion d'un utilisateur, certains critères devront être introduits, dépendant du type de l'utilisateur.

Pour un professeur : nom, prénom, type et le profil lié.

Pour un Invité : nom, prénom, type et le profil lié.

Le login de l'utilisateur et son mot de passe seront automatiquement créés après l'insertion des données de l'utilisateur.

Toutefois, il faudra faire attention à trois choses :

- il n'est pas possible d'insérer un administrateur via l'application. Cela ne pourra se faire que via la base de données ;
- un étudiant ne pourra être introduit manuellement dans l'application. Ceci ne sera géré que par les scripts ;
- le profil devra être créé préalablement s'il n'existe pas encore.

3. CRUD logiciel

L'administrateur pourra créer, lire, mettre à jour et supprimer des logiciels du système.

L'insertion d'un logiciel se fait via son nom. Attention : les doublons de noms ne sont pas possibles.

4. CRUD profil

L'administrateur pourra créer, lire, mettre à jour et supprimer des profils du système. Un profil est composé d'un nom unique.

5. Import fichier CSV

Cette fonctionnalité va permettre de créer, mettre à jour ou supprimer les utilisateurs de la base de données. La gestion de ceux-ci se fera via l'appel d'un fichier CSV qui comprendra toutes les informations nécessaires d'un utilisateur. La création de login et mot de passe se feront automatiquement pour chaque utilisateur (étudiant) dans le cas d'une insertion ou d'une mise à jour.

Chef de projet : Cedric Tavernier

Membres : Degrève Olivier, Denuit Maxime, Dubois Corenthin

6. Générer des scripts login

L'administrateur système pourra générer un script de login pour Claroline, Windows et Nutrilog. D'autres logiciels pourront être ajoutés, modifiés et/ou supprimés dans le futur.

Trois fonctionnalités de base seront donc disponibles.

Un fichier CSV sera créé pour Claroline et contiendra les informations suivantes : nom de l'étudiant, prénom de l'étudiant, son email et son mot de passe.

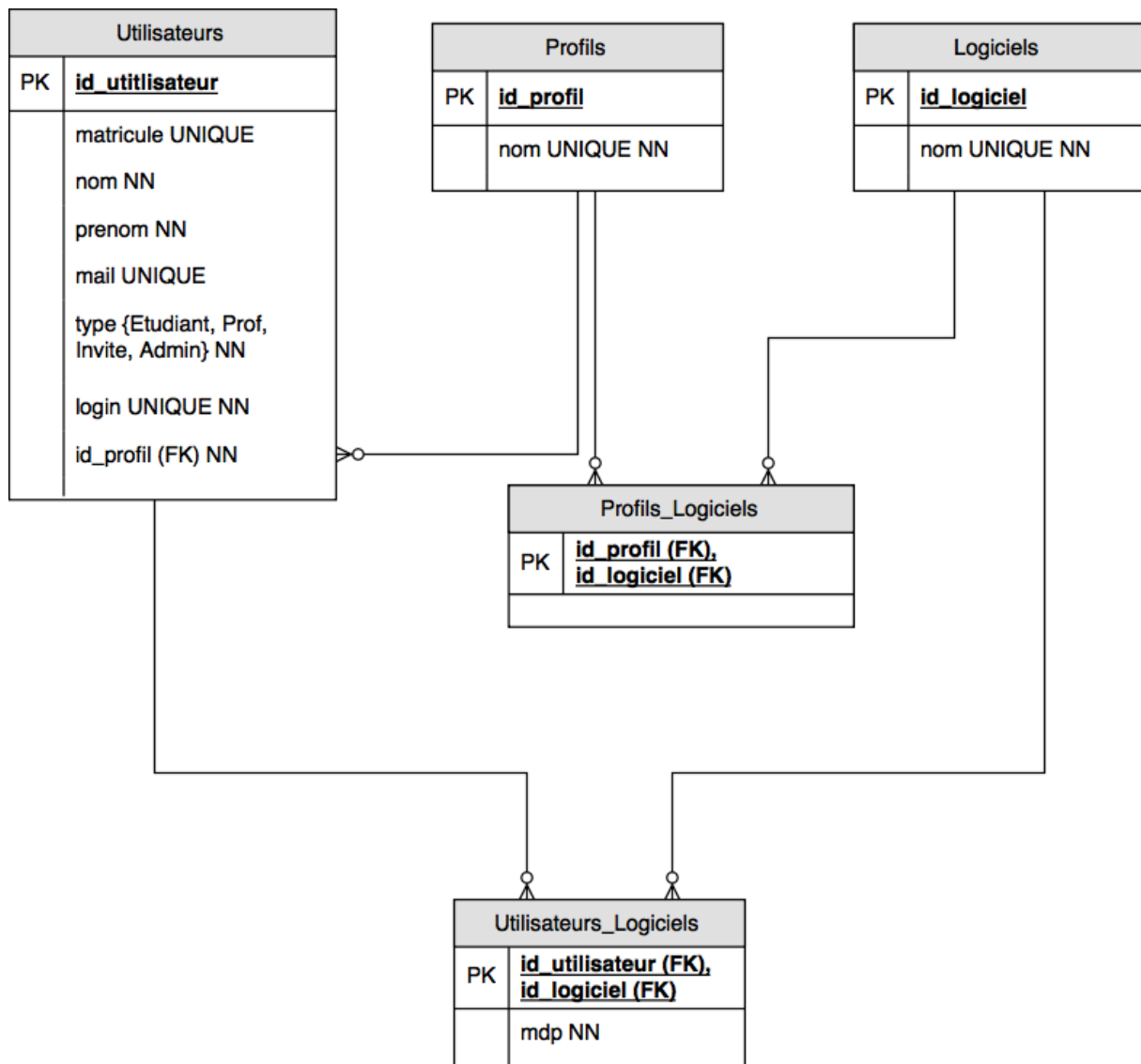
Un fichier texte (.bat) qui comprendra le nom de l'étudiant, le prénom de l'étudiant et son mot de passe.

Un fichier CSV sera aussi créé pour Nutrilog contenant : le numéro d'identification de l'étudiant, son nom, son prénom et son mot de passe.

Quand l'administrateur demande de générer le script, le fichier en question se mettra automatiquement à jour.

5.3. Base de données :

Gestion de login :



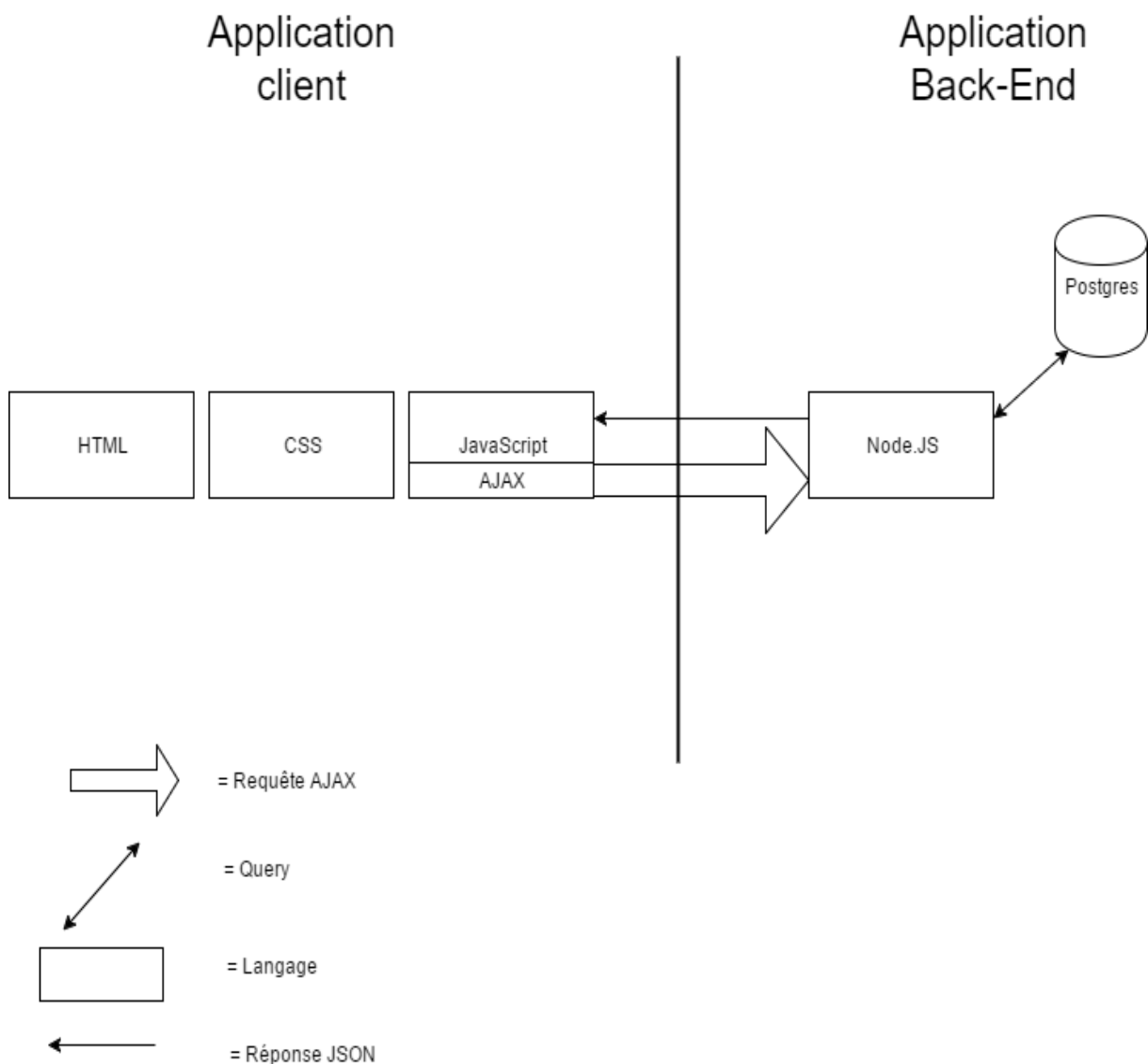
Au niveau de la base de données, nous avons créé 5 tables : une table Utilisateurs, une table Profils, une table Logiciels, une table Profils_Logiciels et enfin une table Utilisateurs_Logiciels. Nous n'avons pas créé pas de table Administrateur car celle-ci peut être évitée par la création d'un champ « mot de passe » dans la table intermédiaire Utilisateurs_Logiciels. Il faudra dès lors ajouter dans la table Logiciels, un « tuple » qui correspond à l'application globale. Pour l'application globale et un utilisateur, nous emploierons le mot de passe de l'administrateur. Ceci évite la création d'une table Administrateur qui ne contiendrait qu'un seul « tuple » et par conséquent un usage inutile de l'espace mémoire. Néanmoins, il existe une petite subtilité dont il va falloir tenir compte lors de suppression de données dans la base de données : il faudra obliger la base de données à ne pas supprimer les « tuples » directement liés à l'administrateur et à l'application globale. Attention, le matricule de l'utilisateur n'accepte que les entiers et le login que les caractères.

5.4. Organisation du back-end Node.js :

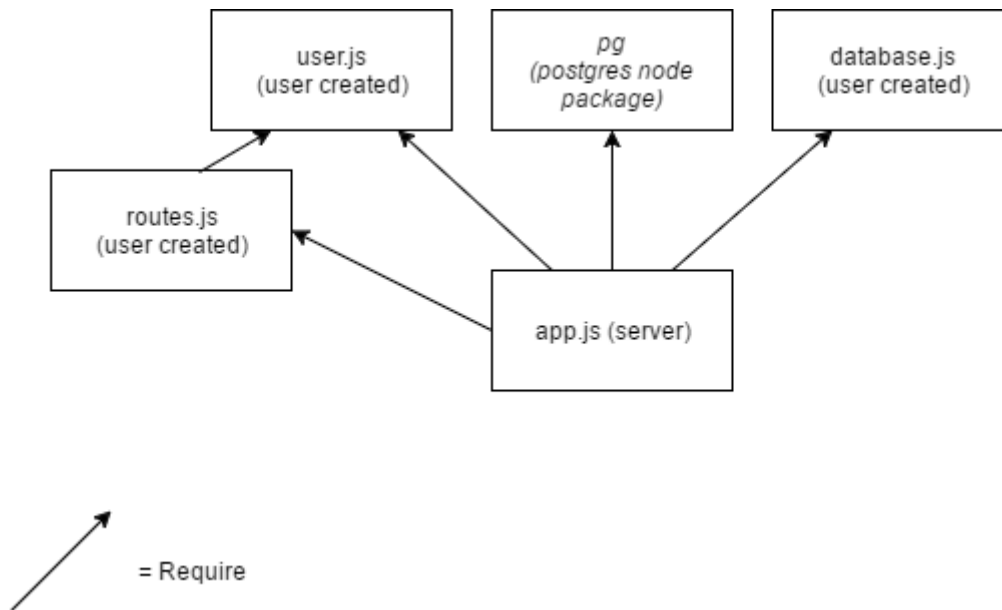
5.4.1. Organisation du code :

Le projet comportera deux applications. La première sera dédiée au client et comportera une interface graphique (HTML, CSS, ...) qui communiquera avec la deuxième application grâce à des requêtes Ajax. La deuxième application agit donc comme back-end de notre première application. En bref, cette seconde application recevra des tâches à effectuer qui proviennent du client (c'est-à-dire, l'administrateur système dans le cas de notre projet) en utilisant la première application. Ce back-end effectuera alors les tâches qui lui ont été confiées et fera si nécessaire des requêtes via la base de données PostgreSQL.

Voici un schéma représentant ces 2 applications :



5.4.2. Organisation des modules Node.js :



Italique = librairie / module externe

Le back-end Node.js comportera une structure précise. L'« `app.js` » sera la pièce centrale du back-end et se chargera d'initier en premier la connexion avec la base de données PostgreSQL. Pour cela, il sera fait appel à une « `database.js` » qui sera un fichier de configuration possédant la configuration nécessaire pour se connecter à la base de données. En effet, le but est de permettre la modification des informations de la base de données - si besoin est - sans devoir modifier l'`app.js`. Pour permettre une connexion PostgreSQL, nous utiliserons la librairie externe « `pg` ». De plus, l'`app.js` fera appel à des « fichiers modèles » pour le Json. À titre d'exemple, nous utiliserons le modèle « `user.js` » qui définira quelles sont les informations d'un utilisateur.

Enfin, « `routes.js` » contiendra les différentes méthodes qui seront exécutées en fonction de la requête Ajax envoyée par l'application client.

Chef de projet : Cedric Tavernier

Membres : Degrève Olivier, Denuit Maxime, Dubois Corenthin

5.5. IHM :

Page d'accueil :

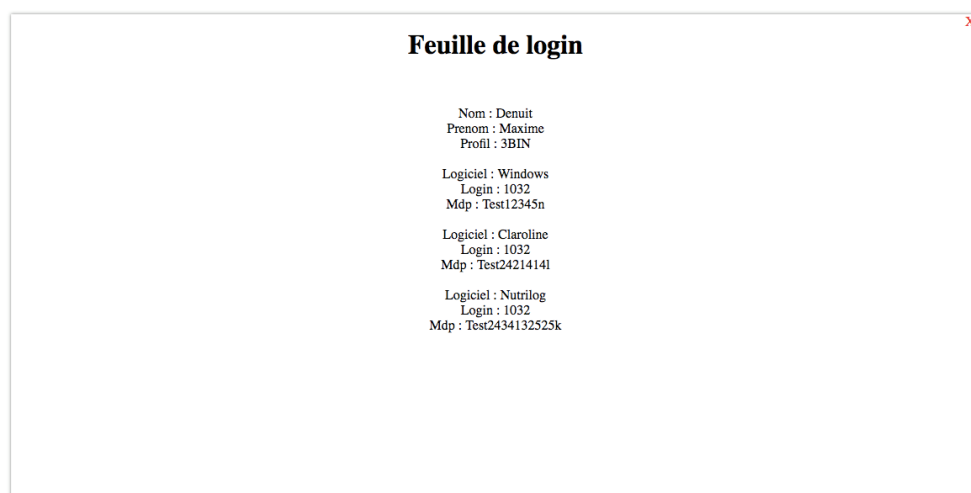


Lorsqu'un utilisateur se connecte à l'application, il tombe sur la page de matricule.

Si c'est un étudiant, il va dès lors rentrer son matricule et va recevoir ses informations de login et mot de passe comme affiché ci-dessous.

Si c'est un administrateur système, il va soit avec le matricule d'un étudiant afficher sa feuille de login, soit cliquer sur le bouton connexion où un pop-up de connexion s'affichera. L'administrateur système pourra alors se connecter en rentrant les informations de connexion.

Feuille de login :



Feuille de login	
Nom : Denuit	
Prenom : Maxime	
Profil : 3BIN	
Logiciel : Windows	
Login : 1032	
Mdp : Test12345n	
Logiciel : Claroline	
Login : 1032	
Mdp : Test2421414l	
Logiciel : Nutrilog	
Login : 1032	
Mdp : Test2434132525k	

6. Conclusion

Après avoir analysé la demande du client, nous pensons avoir développé une stratégie adéquate qui permet de développer une application selon ses souhaits.

7. Bibliographie :

1) Documentation concernant la base de données :

- <http://blog.postgresql.fr/index.php?post/drupal/216>
- <https://www.mongodb.com/fr>

2) Documentation concernant le Back-end :

- <http://www.infoworld.com/article/2975233/javascript/why-node-js-beats-java-net-for-web-mobile-iot-apps.html>
- <https://www.quora.com/What-are-the-advantages-of-using-Node-js>
- <https://blog.xervo.io/top-10-reasons-to-use-node>

3) Documentation concernant le Front-end :

- <http://www.punkchip.com/why-dont-you-use-bootstrap/>
- <http://blog.udacity.com/2014/12/front-end-vs-back-end-vs-full-stack-web-developers.html>
- <https://www.nczonline.net/blog/2013/10/07/node-js-and-the-new-web-front-end/>

4) Documentation concernant les outils utilisés :

- <http://www.journaldunet.com/web-tech/developpeur/1185911-github-le-reseau-social-des-developpeurs/>
- <http://learnosm.org/fr/advanced/github-sharing/>