Gestionnaire de club scientifique de l’ÉTS – GCS

**SimplETS**

Document d’architecture

Version <1.0>

**Fait par :**

Jean Bernier Vibert

Patrick Lavallée

Simon Turcotte-Langevin

Historique des révisions

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Auteur** |
| 26/06/2014 | 1.0 | Version initiale | Simon Turcotte-Langevin |
| 01/07/2014 | 1.1 |  | Simon Turcotte-Langevin |
|  |  |  |  |
|  |  |  |  |

Table des matières

1. Introduction 4

1.1 Contexte 4

1.2 Fonctionnalités 4

1.3 Attributs de qualité 4

1.3.1 Convivialité 5

1.3.2 Intégrité 6

1.3.3 Extensibilité 7

1.4 Contraintes 8

1.5 Parties prenantes 8

2. Présentation de l’architecture 9

2.1 Les services web 9

2.1.1 Principes architecturaux 9

2.1.2 Couches architecturales 11

2.2 Client des services 12

2.2.1 Principes architecturaux 12

2.2.2 Couches architecturales 12

2.3 Site web 13

2.3.1 Principes architecturaux 13

2.3.2 Couches architecturales 13

3. Infrastructure 15

3.1 Base de données 15

3.2 Serveur d’application 15

3.3 LDAP 15

3.4 Machines physiques 15

4. Futur de la solution 16

4.1 Utilisation de la pile Microsoft 16

[Figure 1 - Présentation sommaire de l'architecture 10](#_Toc394600269)

[Figure 2 - Visualisation de certains aspects 11](#_Toc394600270)

[Tableau 1 - SC01: Ajouter un nouvel étudiant 5](#_Toc394600300)

[Tableau 2 - SC02: Modifier son profil 5](#_Toc394600301)

[Tableau 3 - SC03: Ajouter un suivi 6](#_Toc394600302)

[Tableau 4 - SC04: Usager non authentifié 6](#_Toc394600303)

[Tableau 5 - SC05: Usager expiré 7](#_Toc394600304)

[Tableau 6 - SC06: Changement de contexte de club 7](#_Toc394600305)

[Tableau 7 - SC07: Contrôle des données nominatives 7](#_Toc394600306)

[Tableau 8 - SC08: Ajouter une librairie à la configuration Maven 8](#_Toc394600307)

[Tableau 9- Contraintes du projet 9](#_Toc394600308)

[Tableau 10 - Parties prenantes du projet 9](#_Toc394600309)

[Tableau 11 - Tableau des exceptions 12](#_Toc394600310)

[Tableau 12 - Langue offertes par SimplETS 12](#_Toc394600311)

Document d'architecture

# Introduction

## Contexte

Ce document est une référence architecturale pour le développement logiciel de la plateforme Web de gestion de club étudiant SimplETS. Dans ce document, vous trouverez les prérequis architecturaux, les technologies qui seront utilisées, l’architecture logicielle qui supportera les requêtes utilisateurs ainsi que l’illustration de certains concepts clé dont l’AoP (Aspect Oriented Programming) et l’inversion de contrôle.

## Fonctionnalités

Le système à concevoir permettra à un étudiant de l’ÉTS, qui est membre d’un club étudiant, la manipulation des données administrative de son club au travers une série d’outils qui seront à sa disposition. Les fonctionnalités qui seront mise de l’avant se segmentent en 3 modules distincts : la gestion des membres, la gestion des commanditaires et la gestion des fournisseurs.

Le module de gestion des membres permettra d’ajouter des nouveaux membres à un club à l’aide du code universel. Un profil sommaire devra se créer avec les données puisées dans l’Active Directory de l’école. Un membre pourra accéder à son profil et remplir les champs manquant s’il le souhaite. Seul ce dernier pourra modifier les informations nominatives de son profil. Le capitaine pourra supprimer un membre du club mais le profil utilisateur sera conservé.

Le module de gestion des commanditaires permettra d’ajouter de nouveaux commanditaires et devra permettre l’affichage d’un historique des suivies sur demande. Il devra s’assurer de l’étanchéité des données qu’il expose afin de conserver les données sensibles au sein du club.

Le module de gestion des fournisseurs permettra d’ajouter de nouveaux fournisseurs et ses produits dans l’optique de construire un catalogue de produit. Les fonctionnalités de gestion d’inventaire faciliteront l’approvisionnement aux travers d’alertes notifiant le responsable des stocks.

## Attributs de qualité

L’emphase a été mise sur la convivialité, la sécurité et l’évolutivité du logiciel. Ces trois attributs de qualités sont primordiaux pour un système qui sera utilisé exposant des données administratives et financières. D’autres attributs tels que la disponibilité et la performance sont également importants; la section suivante décrira certains scénarios qui nécessitent ces attributs.

Dans cette section du document, vous retrouverez 8 scénarios de qualité logicielle ainsi que l’évaluation de ces scénarios.

### Convivialité

Cet attribut est très important, car le système sera utilisé par les clubs étudiants regroupant environ 300 usagers. La convivialité rend le système plus ergonomique donc plus facile d’utilisation minimisant ainsi la courbe d’apprentissage d’un nouvel usager. De plus, l’utilisateur doit être en mesure de déduire aisément les actions qu’il peut accomplir à l’aide de la description textuelle ou des images.

|  |  |
| --- | --- |
| **Nom** | SC01 |
| **Source** | Usager |
| **Stimulus** | Un capitaine désire ajouter un membre à son club en fournissant le code universel au système. A partir de la liste de membre actuelle, il appuis sur le bouton « Ajouter ». |
| **Artéfact** |  |
| **Environnement** | Exécution normale |
| **Réponse** | Le système valide le code universel et créer le profil de l’étudiant s’il n’existe pas déjà et associe l’étudiant au club. Le capitaine est notifié du résultat de l’ajout. |
| **Mesure de la réponse** | * L’étudiant est associé au club et peux consulter en lecture seulement les données du club. * L’ajout d’un membre doit se faire en 3 actions. |

Tableau 1 - SC01: Ajouter un nouvel étudiant

|  |  |
| --- | --- |
| **Nom** | SC01 |
| **Source** | Usager |
| **Stimulus** | Un usager désire modifier son profil utilisateur. |
| **Artéfact** |  |
| **Environnement** | Exécution normale |
| **Réponse** | Le système valide les entrées utilisateur et informe l’utilisateur du résultat de la mise à jour. |
| **Mesure de la réponse** | La mise à jour prend moins de 5 secondes. |

Tableau 2 - SC02: Modifier son profil

|  |  |
| --- | --- |
| **Nom** | SC03 |
| **Source** | Usager |
| **Stimulus** | Un usager ajoute un suivi à un commanditaire. |
| **Artéfact** |  |
| **Environnement** | Exécution normale |
| **Réponse** | Le système valide les entrées utilisateur et informe l’utilisateur du résultat de la mise à jour. |
| **Mesure de la réponse** | Le nouveau suivi est ajouté à la liste des suivis affichée en ordre décroissant de date de création (le plus récent s’affiche en premier) |

Tableau 3 - SC03: Ajouter un suivi

### Intégrité

Cet attribut permet de protéger les informations d’authentification, nominatives et de session utilisateur. Les requêtes utilisateurs sont encryptées protégeant ainsi les données qu’elles véhiculent. La solution étant déployée sur un serveur centralisé, les opérations s’exécutent selon le contexte de l’utilisateur qui les déclenche. Dans cette optique, chaque utilisateur ne voit que les données dont il a la permission de voir. Finalement, chaque requête nécessitant un usager authentifié est audité.

|  |  |
| --- | --- |
| **Nom** | SC04 |
| **Source** | Usager non authentifié |
| **Stimulus** | L’usager s'authentifie avec des informations erronées |
| **Environnement** | Exécution normale |
| **Réponse** | Le système refuse l’accès notifiant l’usager de la raison. |
| **Mesure de la réponse** | L’authentification prend moins de 5 secs. |

Tableau 4 - SC04: Usager non authentifié

|  |  |
| --- | --- |
| **Nom** | SC05 |
| **Source** | Usager avec une session expiré |
| **Stimulus** | L’usager rafraîchit la page actuelle |
| **Environnement** | Exécution normale |
| **Réponse** | Le système redirige l’usager sur la page d’authentification |
| **Mesure de la réponse** | La redirection se fait en moins de 2 secondes. |

Tableau 5 - SC05: Usager expiré

|  |  |
| --- | --- |
| **Nom** | SC06 |
| **Source** | Usager authentifié appartenant à au moins 2 clubs étudiants |
| **Stimulus** | L’usager change le contexte du club |
| **Environnement** | Exécution normale |
| **Réponse** | Le système rafraîchit la page actuelle et charge les données appropriées. |
| **Mesure de la réponse** | * Le rafraîchissement se déroule en moins de 6 secondes. * L’opération est auditée et journalisée. |

Tableau 6 - SC06: Changement de contexte de club

|  |  |
| --- | --- |
| **Nom** | SC07 |
| **Source** | Usager authentifié |
| **Stimulus** | L’usager modifie l’état de visibilité des données nominative |
| **Environnement** | Exécution normale |
| **Réponse** | Un retour visuel confirme à l’utilisateur que ses données ne sont pas affichées dans le profil public. |
| **Mesure de la réponse** | Les données nominatives ne sont pas cachées du profil public. |

Tableau 7 - SC07: Contrôle des données nominatives

### Extensibilité

L’évolutivité de la solution est un attribut primordial. Le produit est développé dans un contexte scolaire et les développeurs subséquents seront des étudiants qui voudront bonifier le système dans le cadre d’un cours ou d’un projet. Dans cette optique, le système doit offrir des outils de développement et de configuration génériques qui permettront d’ajouter des fonctionnalités supplémentaires en facilitant la gestion des dépendances. L’usage du paradigme d’AoP (Aspect Oriented Programming) allège la lecture du code en injectant les méthodes appropriées (Journalisation, Audit, Sécurité) aux méthodes sollicitées.

|  |  |
| --- | --- |
| **Nom** | SC08 |
| **Source** | Maintenance |
| **Stimulus** | Une nouvelle technologie ou une nouvelle version d’une technologie déjà utilisée est requise par l’application. |
| **Artéfact** |  |
| **Environnement** | Développement et compilation |
| **Réponse** | La nouvelle technologie est reconnue par Maven et est ajoutée dans le « classpath » de l’application |
| **Mesure de la réponse** | La totalité (100%) des dépendances sont mises à jour. |

Tableau 8 - SC08: Ajouter une librairie à la configuration Maven

## Contraintes

Il y a plusieurs contraintes qu’il faut prendre en compte lors de la réalisation de l’architecture. Dans le tableau ci-dessous, vous trouverez les différentes contraintes ainsi qu’une brève explication.

|  |  |
| --- | --- |
| **Contrainte** | **Explication** |
| Connexion sur VPN | Un étudiant doit absolument établir une connexion avec le VPN de l’école afin de pouvoir s’authentifier auprès du LDAP en place lorsqu’à l’extérieur de l’enceinte de l’école. |
| Gestion des données nominatives | Les étudiants ont le droit d’afficher ou pas les informations nominatives qui les concernent. La politique de l’école à cet égard doit être respectée. |

Tableau 9- Contraintes du projet

## Parties prenantes

Dans le tableau ici-bas, on peut voir qu’il y a plusieurs groupes organisationnels qui sont impliqués dans le développement et chacun à ses propres priorités et attentes face au développement du logiciel.

|  |  |  |
| --- | --- | --- |
| **Partie prenante** | **Demande** | **Priorité** |
| Avocate | Protection des informations sensibles d’un étudiant. | Haute |
| Responsable des clubs étudiants | Automatisation dans l’obtention des listes des membres des clubs. | Haute |
| Usager | Logiciel convivial et performant | Haute |
| Développeur | Flexibilité d’ajout de nouvelle fonctionnalité | Moyenne |
| Administrateur Système | Facilité d’installation et de configuration | Basse |
| Client | Livré dans les temps | Moyenne |

Tableau 10 - Parties prenantes du projet

# Présentation de l’architecture

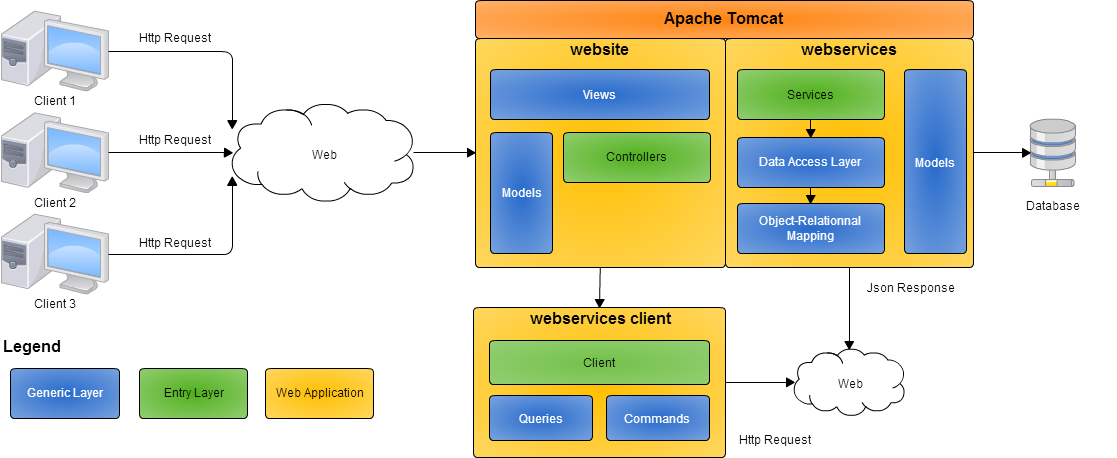


Figure 1 - Présentation sommaire de l'architecture

Dans un premier ordre d’idée, SimplETS devait être une application centralisée afin de minimiser l’effort de déploiement. Il a donc été décidé d’opter pour une architecture web. Le choix n’a pas été difficile à faire puisque tous les membres de l’équipe étaient à l’aise avec la programmation d’applications web. Afin de découpler la logique d’affaire de la logique d’affichage, il a été convenu que l’application devrait être scindée en deux entités : le site et les services. De plus, afin de faciliter l’intégration future avec d’autres systèmes, une troisième entité a été introduite : le client des services. Celle-ci vise à encapsuler toute la logique d’interfaçage avec les services web.

## Les services web

Cette entité architecturale était celle qui a été la plus complexe à concevoir et développer, car l’entièreté du modèle d’affaire de SimplETS s’y retrouve. Les sections suivantes décriront les enjeux de cette entité architecturale.

### Principes architecturaux

#### Authentification

Premièrement, les services doivent en tout temps valider l’identité des usagers. En effet, tel que discuté précédemment, la sécurité est primordiale, et c’est dans les services que toute la sécurité se retrouve. Un module d’authentification permet à un usager de valider son identité via le protocole « Kerberos ». Lorsque l’identité de l’usager est validée, un jeton de session est généré. Ce jeton est ensuite concaténé avec l’adresse IP de l’utilisateur, haché puis retourné à l’utilisateur. Cet ultime jeton sera nécessaire lors de toute requête utilisateur.

#### Autorisation

Deuxièmement, un jeton d’authentification n’autorise pas un utilisateur à faire comme bon lui semble. Chaque utilisateur est associé à des clubs étudiants, et n’a le droit de prendre action que sur ces clubs. Chaque requête utilisateur visant à modifier les données d’un club doit être autorisée; les services web s’assurent donc que l’étudiant fait partie de ce club, et qu’il a le niveau d’autorisation nécessaire pour entreprendre cette action. Afin de faciliter les ajustements au niveau des droits, un module utilitaire permet de gérer les droits, par module, à l’aide d’une configuration Xml.

#### Journalisation

Troisièmement, afin d’améliorer l’expérience des débuggeurs, tous les services contiennent une journalisation des événements afin de pouvoir réagir rapidement en cas de faute du système. Ainsi, les bogues majeurs pourront être facilement répertoriés et corrigés afin de toujours améliorer l’expérience de l’utilisateur.

#### Audit

Quatrièmement, chaque requête utilisateur qui nécessite un usager authentifié et autorisé sera auditée. L’adresse IP de l’usager ainsi que les informations contextuelles à sa requête seront conservées afin de pouvoir retracer les potentiels usagers malveillants.

#### Validation

Cinquièmement, chaque requête utilisateur est validée afin que les préconditions de chaque contrat soient satisfaites. Des messages d’erreur sont générés pour chaque précondition afin que l’utilisateur puisse facilement comprendre pourquoi sa requête a été rejetée.

#### Programmation orientée aspect

Afin d’implémenter facilement et uniformément les points [2.1.1.1](#_L’authentification) à [2.1.1.5](#_Validation), il a été décidé qu’une approche orientée aspect était de mise. Ainsi, les métadonnées nécessaires à l’application contrôlée des aspects ont été créées sous la forme d’annotations. Conséquemment, les services qui nécessitent de la journalisation ont l’annotation *@Loggable*, et ceux qui nécessitent de la validation ont l’annotation *@Validatable*. À travers la solution, plusieurs aspects ont été développés afin de maximiser la réutilisation du code.

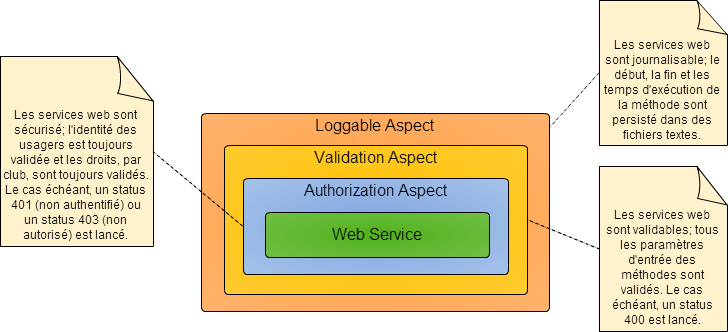


Figure 2 - Visualisation de certains aspects

#### Inversion de contrôle

Conformément aux architectures web modernes, notre solution applique de l’inversion de contrôle. Ainsi, chacune des dépendances est exprimée via des interfaces. Les dépendances sont résolues lors de l’exécution à l’aide du conteneur d’inversion de contrôle de « Spring ». Grâce à ce paradigme, chacune des composantes peut être facilement substituée afin d’offrir un comportement supplémentaire, ou différent.

#### Gestion des exceptions

Afin de permettre aux utilisateurs de connaître les raisons des fautes systèmes, une utilisation rigoureuse des exceptions a été de mise. Grâce au cadriciel Jersey, les exceptions peuvent être mappées à des statuts http, ce qui permet d’être standard au paradigme REST.

|  |  |  |
| --- | --- | --- |
| **Exception** | **Statut http** | **Description** |
| EntityNotFoundException | 404 (Not Found) | La ressource demandée n’existe pas, ou plus. |
| ValidationException | 400 (Bad Request) | La requête utilisateur n’a pas passé le test de validation; la requête ne peut être exécutée. |
| UnauthorizedException | 403 (Forbidden) | L’usager n’a pas le niveau de droit nécessaire pour entreprendre cette action. |
| WrongCredentialsException | 403 (Forbidden) | L’usager a tenté de valider son identité, mais cette validation a échoué. |
| NotAuthenticatedException | 401 (Unauthorized) | L’usager a tenté d’accéder à une ressource privée qui nécessite un usager authentifié. |
| Throwable | 500 (Internal Server Error) | Un scénario non envisagé par les développeurs est survenu. |

Tableau 11 - Tableau des exceptions

#### Multilinguisme

Tous les messages dans l’application sont traduits dans la langue de l’utilisateur. Un module de traduction permet de substituer des clés de message par un message valide dans la langue de l’utilisateur. Les clés de message et leur traduction sont configurables dans des fichiers xml.

|  |  |
| --- | --- |
| **Langue** | **Justification** |
| Français | Langue d’origine du Québec et langue officielle de l’ÉTS |
| Anglais | Langue seconde du Québec et langue principale au Canada. |

Tableau 12 - Langue offertes par SimplETS

### Couches architecturales

#### Couche services

La couche service est le point d’entrée de l’application. C’est cette couche qui permet d’associer un comportement à un Url. Avant l’exécution de ces méthodes, tous les aspects valide les différentes préconditions du module afin de rencontrer les standards de sécurité et d’intégrité requis. De plus, c’est dans la couche service que se retrouve l’intégralité des cas d’utilisation de l’application. Dans cette optique, la programmation orientée aspect a grandement aider à clarifier le code afin que les méthodes ne contiennent que la logique d’affaire.

#### Couche « Data Access Object (DAO) »

La couche des objets d’accès aux données contient toutes les requêtes à la base de données. Cette couche permet de regrouper les requêtes par module, afin de mieux séparer cet accès aux données. Cependant, à ce niveau, il n’est pas défini si nous utilisons un gestionnaire de base de données ou bien un autre type de stockage. En effet, cette couche a été implémentée à l’aide de l’interface programmable offert par Hibernate, qui permet d’abstraire le concept de base de données.

#### Couche « Object-Relational Mapping (ORM) »

La couche de mapping objet-relationnel, offerte par Hibernate, permet d’abstraire le concept de base de données. À l’aide d’une configuration, Hibernate est en mesure de comprendre comment obtenir et modifier les données, indifféremment du support de stockage.

#### Couche modèle

La couche du modèle contient chacun des objets qui définissent le modèle d’affaire. Ces objets contiennent la configuration Hibernate; ils doivent donc refléter le schéma de base de données.

## Client des services

### Principes architecturaux

#### Cache

La librairie des clients offre un cache configurable afin de pouvoir conserver en mémoire les réponses peut susceptibles d’être modifier. Ainsi, la performance des objets clients et considérablement améliorée, puisqu’aucune requête au serveur des services n’est requise.

#### Point d’intégration

La librairie des clients offre un point d’intégration fiable aux systèmes qui désirent interfacer avec le nôtre. Il est de loin favorable d’avoir ce point d’intégration plutôt que de se fier à des requêtes SQL provenant de l’extérieur. En s’assurant que toutes les requêtes passent par notre passerelle, il devient très improbable que le contenu de notre base de données soit corrompu.

#### Sérialisation

Toutes les requêtes utilisateur sont sérialisées en Json. Il n’est pas prévu de supporter un autre format. La sérialisation Json nécessite très peu de configuration, puisque le Json est une notation objet, et c’est un format compact, pour du texte.

#### Beans

La librairie des clients contient toutes les « beans », ces objets simples et sérialisables qui représentent le modèle. Les « beans » sont configurables afin de définir leurs règles de validation et leurs règles d’affichage. Un algorithme a été développé afin de facilement transformer un objet du modèle en « bean », et vice-versa.

### Couches architecturales

#### Requêtes/Réponses

Cette couche contient des classes sérialisables qui représentent les paramètres d’entrée et de sortie de la couche client.

#### Beans

Cette couche contient d’autres classes sérialisables qui permettent de représenter un objet du modèle. Ces représentations peuvent être partielles ou complètes, selon le niveau d’autorisation de l’utilisateur.

#### Clients

Ce sont les classes qui permettent d’utiliser les fonctionnalités de SimplETS. Elles connaissent les Url (Universal Resource Locator) à appeler ainsi que le format d’échange afin d’obtenir des réponses positives du serveur.

## Site web

### Principes architecturaux

#### Patron « Front Controller »

Toutes les requêtes http transigeant dans le système sont prises en charge par une seule entité : le « Front Controller ». Celui-ci, offert par la librairie « Spring », permet d’abstraire le principe des servlets, et d’utiliser des Contrôleurs; ceux-ci jouent conséquemment le rôle de contrôleur dans le paradigme MVC.

#### Pages partielles

Afin de d’optimiser la réutilisation du html, le contenu a été divisé en pages partielles. Il existe deux types de pages partielles : les pages partielles de contenu, qui seront chargée à l’aide d’Ajax, et les pages d’agencement (« Layout »), qui permettent la segmentation et la réutilisation du html.

#### Chargement Ajax

Certaines pages partielles seront chargées à l’aide de la technologie Ajax. Vu qu’une grande partie de l’application ne change pas entre les pages, il est plus optimal de ne chargé que le contenu, via cette technologie asynchrone. Elle permet de sauver beaucoup de trafic internet.

#### SSL et Https

L’application n’est disponible que sous Https. Un filtre a été mis en place afin d’assurer que toutes les requêtes http transigeant dans notre système aie connexion sécurisée par SSL. Le cas échéant, l’application redirige automatiquement l’utilisateur sur une connexion sécurisée.

#### Authentification

Un usager doit être authentifié aux services web s’il veut pouvoir accéder à l’application. Le cas échéant, un filtre interceptera la requête http, et redirigera l’utilisateur sur la page d’authentification.

### Couches architecturales

#### Contrôleurs

La couche des contrôleurs est le point d’entrée des requêtes http. Celle-ci est responsable de coordonner les requêtes vers les services web et d’appliquer la logique de navigation. Elle s’assure que les requêtes provenant de l’utilisateur sont valides, et cette couche décide de la ressource qu’elle doit retourner, ainsi que les actions qu’elle doit entreprendre.

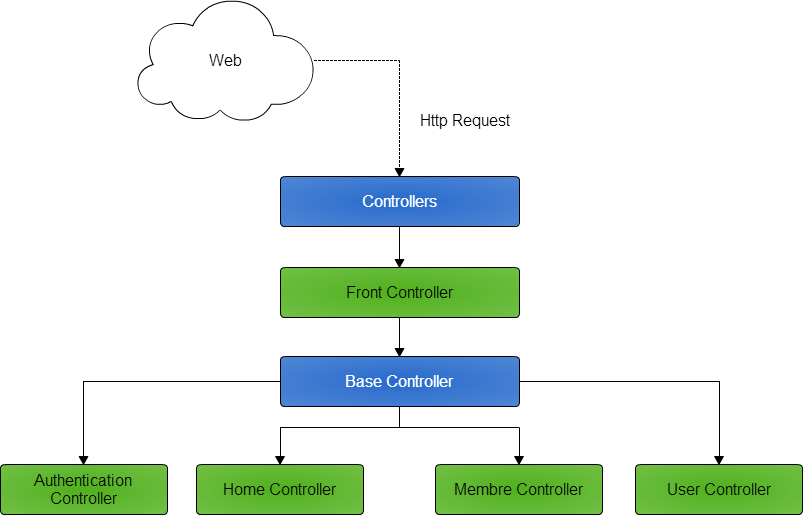


Figure 3 - Hiérarchie des contrôleurs et le « Front Controller »

#### Vues

La couche des vues regroupent l’ensemble des ressources de SimplETS. C’est dans celle-ci que la logique d’affichage est appliquée afin de fournir aux clients une interface intuitive. Ces vues contiennent du Html, du Javascript et du CSS afin d’offrir des pages dont le contenu est cohérent.

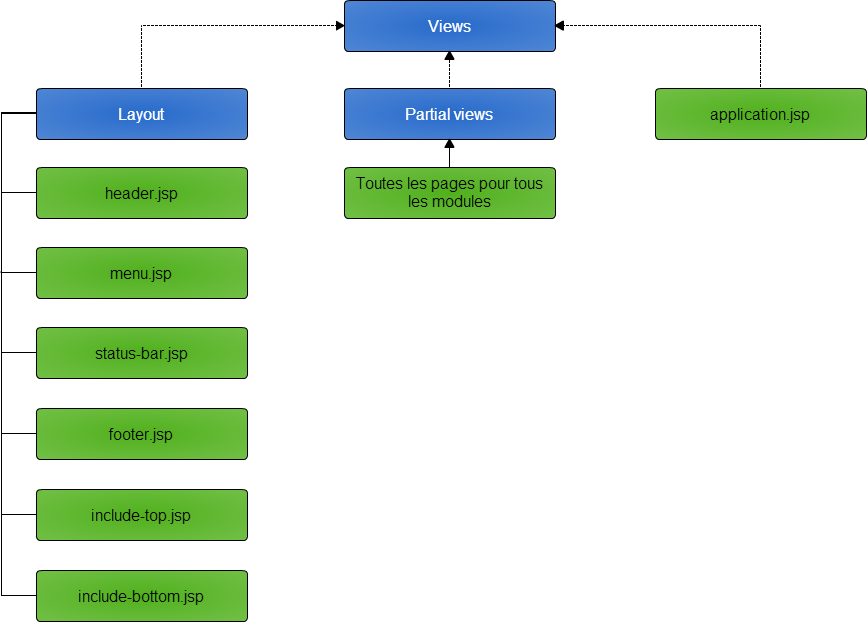


Figure 4 - Liste des pages partielles

# Infrastructure

## Base de données

Une base de données PostgreSQL sera installée afin d’offrir de la persistance. Le choix s’est arrêté sur celle-ci puisqu’elle est la meilleure alternative gratuite.

## Serveur d’application

Un serveur Apache Tomcat 7 sera installé afin d’héberger les applications web nécessaires au bon fonctionnement du système. La configuration initiale du serveur devra contenir un connecteur SSL afin d’assurer les connexions sécurisées. Dans le futur, ce serveur d’application pourrait être abandonné pour une solution un peu plus professionnelle, telle que Glassfish, mais les fonctionnalités de Tomcat sont suffisantes pour le moment.

## LDAP

Le serveur LDAP de l’école sera utilisé afin d’authentifier les usagers. Ce choix était le plus avantageux puisqu’il permet aux usagers d’utiliser les mêmes informations d’authentification que dans les autres services offerts par l’école.

## Machines physiques

Un seul serveur sera nécessaire pour héberger SimplETS. Vu que nous n’estimons pas une surcharge des serveurs, le site web, les services web et la base de données seront localisés sur la même machine. Éventuellement, il pourrait être décidé de séparer la charge sur plus d’une machine, afin d’augmenter la capacité d’utilisateurs concurrents.

# Futur de la solution

## Utilisation de la pile Microsoft

L’utilisation d’une pile de technologies hétérogènes « open-source » a sérieusement entravé les performances de l’équipe. En effet, faire cohabiter des projets tels que AspectJ, JavaEE, Jersey, Spring et Hibernate a été très difficile, et la configuration nécessaire pour assembler le tout est non-négligeable. L’ajout de nouvelles technologies est encore difficile, malgré l’utilisation de Maven. Par exemple, Lombok, une librairie d’utilitaire en java, aurait simplifié de loin notre projet. Cependant, elle ne peut cohabiter avec AspectJ; nous avons donc dû l’abandonner.

La plateforme .NET serait en mesure de remplacer bon nombre des librairies utilisées. En effet, nous pourrions facilement utiliser « ASP.NET MVC » pour les services web, « SQL Server » pour la base de données et « LINQ to SQL » remplacerait Hibernate. Pour quelqu’un qui connaît à la fois le java et le C#, la traduction d’un langage à un autre serait très rapide.

Ultimement, un passage vers la plateforme .NET permettrait de doubler l’efficience des développeurs. Le concept de solution dans « Visual Studio » permet de partager les configurations beaucoup plus facilement entre les développeurs. De plus, la plateforme .NET supporte le paradigme fonctionnel, qui permet de traiter des méthodes comme des données. Dernièrement, « Visual Studio » est tout simplement le meilleur IDE possible. Couplé avec « Resharper », un outil d’instrumentation de code, ce logiciel est tout bonnement génial. Il améliore de beaucoup l’efficience des développeurs, ainsi que la qualité du code produit.