Projet: UVSP 2.0, University Visual Simple Planner 2.0

Ce rapport présente le projet USVP 2.0 en détaillant les différentes technologies choisies et l'organisation mise en place pour le développement.

L'équipe de travail :

Abdeslam Ahardane, Chenyang Gao, Rachid Hadjour, Yann Pugliese

Introduction

Dans le cadre de notre formation à Polytech Montpellier en section IG, il nous a été demandé de reprendre un projet en l'adaptant en application Web. Ainsi, nous avons formé une équipe de travail composé d'Abdeslam AHARANE, Chenyang GAO, Rachid HADJOUR et Yann PUGLIESE.

Le projet a pour but de développer une application de gestion de l'Université de Montpellier II. Nous avions déjà développé un client lourd pour ce projet l'an dernier. Nous allons détailler, dans ce présent rapport toute la démarche suivie pour arriver à nos fins. Nous commencerons par décrire le contexte du projet, puis nous aborderons l'étape de conception. Nous continuerons brièvement avec la partie développement pour terminer par les résultats obtenus.

Le projet

1. Les contraintes et les ressources

Avant d'aborder les objectifs de ce projet, il nous semble nécessaire d'expliciter les contraintes que nous devions respecter. La première contrainte, qui apparait dans la plupart des projets informatiques, est le temps. Nous avions 5 semaines pour réaliser ce projet.

La deuxième contrainte est très liée aux ressources que nous avions. Le projet devait être réalisé en groupe de 7 personnes. Cependant, nous n'avons pas pu former un groupe de 7 personnes contrairement aux autres groupes. Ainsi, nous devions réaliser le même projet, mais avec moins de ressources humaines.

Pour réaliser ce projet, nous avons suivi les cours d'AIOP ainsi que plusieurs TD. Il nous a donc été présenté plusieurs technologies orientées web permettant de réaliser le projet. De plus, ayant réalisé le même projet en client lourd, chaque groupe possède ces propres codes sources. Par conséquent, nous ne sommes pas tous partis avec les mêmes ressources.

2. Les objectifs

Les objectifs du projet n'ont pas été figés, chaque groupe a été libre de définir son périmètre d'action. Ainsi, pour définir nos objectifs, nous avons pris en considération les ressources et contraintes de ce projet. Chaque groupe possède un socle différent puisque chacun d'entre eux possède le code de son client lourd réalisé l'an passé.

En prenant tous ces éléments en considération, il nous a semblé évident que nous ne pouvions pas développer toutes les fonctionnalités que nos clients lourds possèdent. Nous avons donc restreint le périmètre du projet en choisissant les fonctions qui nous semblaient les plus importantes. De ce fait, nous avons décidé d'implémenter que quelques fonctions permettant de gérer les enseignements.

Nous avons donc décidé de garder les fonctions suivantes :

- Visualisation de l'emploi du temps des enseignants
- Gestion des réservations
- Gestion des enseignants
- Gestion des salles

Nous avons choisi la fonction de visualisation de l'emploi du temps d'un enseignant, car c'est pour nous la fonction principale du projet. La plateforme doit permettre aux enseignants de pouvoir connaitre leurs emplois du temps.

La gestion des réservations permet à ces enseignants de pouvoir faire une demande de réservation. Celle-ci pourra être validée, en attente de validation ou annulée.

La gestion des enseignants permet d'ajouter, de visualiser, de modifier et de supprimer un enseignant. Cette fonctionnalité est très importante, car il peut y avoir d'enseignements s'il n'y a pas d'enseignants.

Nous avons ensuite choisi de développer la gestion des salles car cela nous permet maintenant d'avoir un « modèle » de développement. Pour développer d'autres fonctionnalités, on peut maintenant réutiliser le code déjà produit en l'adaptant. C'est pour cela que l'on a choisi de développer ces fonctionnalités en particulier.

3. La répartition des rôles

Pour être plus efficace, nous avons assigné à chaque personne un rôle bien défini. Ces rôles ont été choisis en fonction des compétences et affinités de chaque membre du groupe. Cependant, à part pour la personne en charge des tests, les autres personnes du groupe pouvaient travailler sur toutes les parties du programme mais en cas de changement important ou de modification, il fallait qu'elle en avertisse la personne responsable. Ainsi, si pendant le développement des fonctionnalités, on devait changer l'interface utilisateur, on pouvait faire un changement mineur, sinon il nous fallait avertir la personne en charge de l'interface graphique pour que celle-ci fasse le changement nécessaire.

Ainsi nous avons attribué les rôles suivants :

- AHARDANE : Conception et développement de l'interface graphique, testeur de l'application
- GAO : Développement des fonctionnalités de la couche métier et de l'API de notre application
- HADJOUR : Développement des fonctionnalités de la couche métier et responsable de la base de données
- PUGLIESE : Développement des fonctionnalités de la couche métier, responsable de la communication et de la coordination

La conception

Nous avions à notre disposition la conception de nos clients lourds. Ainsi, nous avons pu adapter la conception que nous avions réalisée.

1. Modélisation ergonomique

Dans un premier temps une étude de l'*ergonomie* du futur site UVSP 2.0 a été menée. Nous avons tout particulièrement étudié la pertinence des nouvelles normes de la technologie *web* HTLM5/CSS3.

Pourquoi HTML5-CSS3 ?

Tout d'abord de nombreux navigateurs internet et navigateurs en création sont compatibles HTML5-CSS3.

Technologie novatrice :

Le HTML5 apporte de nombreuses nouvelles fonctionnalités à la fois pour les développeurs et les utilisateurs finaux. La nouveauté la plus connue est les nouvelles balises qui améliorent la sémantique des pages HTML (voir figure 2 ci-contre).

Mais ce n'est pas la seule nouveauté, le support des fichiers multimédias (audio et vidéo), l'amélioration des formulaires avec la validation avant soumission et de nombreuses nouvelles *API* en JavaScript (et Jquery dans notre application). Une autre nouveauté intéressante est les *Web Sockets* qui permettent de faire communiquer un navigateur et un serveur très facilement, ce qui facilitera le *push* de données l'un vers l'autre.

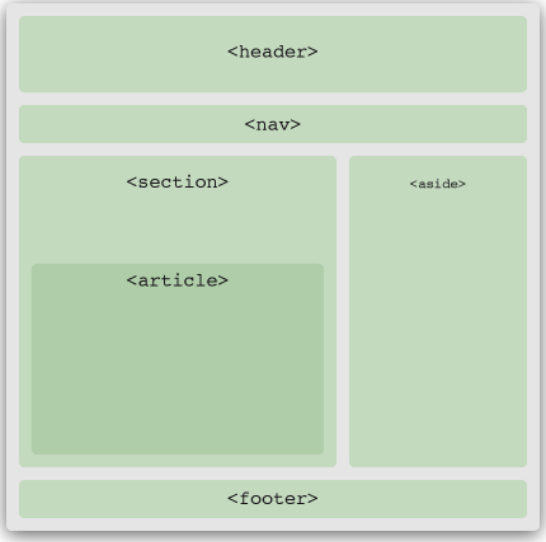


Figure 2 Ergonomie par balise HTLM5

Conclusion de l'étude d'ergonomie :

Le but de UVSP 2.0 est d'être un site internet *compatible avec l'architecture d'UVSP 1.0*, mais aussi d'avoir *une ergonomie optimale* pour obtenir le maximum de productivité avec un minimum d'effort. L'étape suivante sera de créer des maquettes qui permettront ensuite de créer le site internet.

2. Modélisation architecturale

Nous avons décidé de mettre en place une architecture Modèle Vue Contrôleur, car cette architecture apporte de réels avantages. Elle permet d'avoir une grande souplesse pour organiser le développement de l'application, car il y a une indépendance entre les données, l'affichage et les actions du système. Ainsi, la maintenance et l'évolution de l'application seront d'autant plus facilitées.

L'idée est de séparer les données de la présentation et des traitements. En effet MVC organise l'interface entre l'homme et la machine :

- un modèle qui correspond aux données
- une vue qui est l'interface utilisateur
- le contrôleur qui récupère les informations, effectue des traitements et renvoie à la vue les données qui doivent être affichées

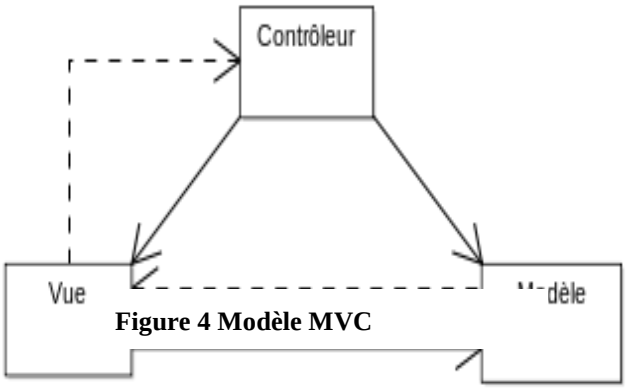


Figure 4 Modèle MVC

3. Choix technologiques

Nous avons dû effectuer différents choix technologiques pour la réalisation du projet. Tout d'abord, nous avons choisi le langage ASP.NET avec le framework MVC et le langage C# pour développer l'application. Ces technologies nous semblaient les plus adaptées au vu du temps imparti.

Pour les vues, l'ergonomie Bootstrap a attiré notre attention. En effet compatible HTML5-CSS3, Bootstrap inclus du LESS. LESS permet de mettre en place un parton HTML (layout) qui détecte quel média est connecté et adapte le rendu au support (gestion des media-queries pour les navigateurs compatibles). Nous avons aussi pu intégrer Jquery et JavaScript. De plus, pour intégrer du code dans nos vues, nous avons utilisé le « Razor View Engine ». Cela permet de générer du code sans surcharger la vue.

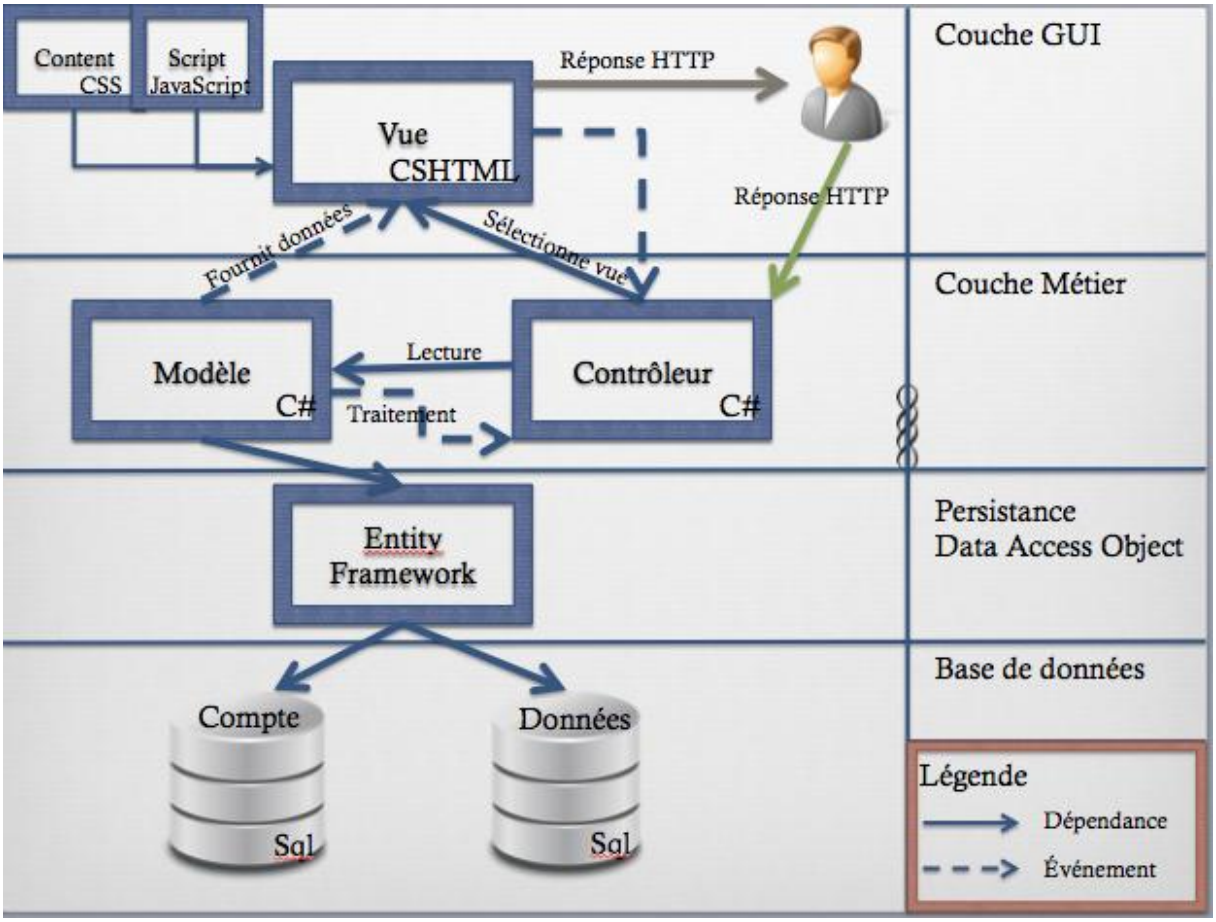
Pour la gestion des bases de données, nous avons utilisé le SGBD : SQL server, car il est fourni et intégré à notre environnement de développement. ADO .NET Entity framework nous a permis de générer automatiquement le modèle basé sur la base de données réalisée l'an dernier. Celle-ci a pu être recréée en utilisant le script de création adapté pour SQL Server.

Lors de la création d'une application Web avec l'IDE : Visual studio, la gestion des comptes est proposée. Nous avons donc utilisé WebSecurity car il respecte des normes de sécurités.

Ces choix ont été guidés par la volonté d'implémenter un maximum de framework afin de se concentrer sur la partie métier de notre application.

4. Résumé schématique

Voici un résumé schématique des choix technologiques ainsi que de la structure de notre application.



L'implémentation

1. Ergonomie et visuelle

La mise en place des maquettes a permis de créer plusieurs prototypes de fenêtre. L'ergonomie principale concerne la fenêtre de consultation, de création ou de gestion (figure 3, ci-après).

Ainsi le choix de la maquette précédente permet d'uniformiser toutes les pages du projet UVSP 2.0. Les futures maquettes respectent certaines règles d'ergonomie mise en place spécialement pour UVSP :

Règle 1 : L'édition et les boutons seront automatiquement mis à droite de la fenêtre et seront fixés à l'écran malgré la taille de la fenêtre. L'utilisateur aura toujours à sa disposition ces boutons. Le fait de les fixer améliore la qualité de navigation pour l'utilisateur.

Règle 2 : La hiérarchie doit être réalisée sous forme d'accordéon, c'est-à-dire que les UE englobent les enseignements qui contiennent les matières.

Règle 3 : Interdiction d'utiliser des Pop-ups (fenêtre qui s'ouvre). Tout doit se réaliser sur la même page, ceci permet d'utiliser le logiciel sur Tablette ou portable (où la pop-up n'existe pas ou peut être très contraignant). Les fenêtres Pop-ups sont remplacées par des Boxs.

Règle de visibilité : les boutons, les lettres, les tableaux et les images ont une taille prédéfinie pour être toujours visible et accessible.

Après validation des maquettes, la réalisation du CSS3 (feuille de style) et des différentes fenêtres en HTML5 permettront de finaliser l'interface visible par l'utilisateur.

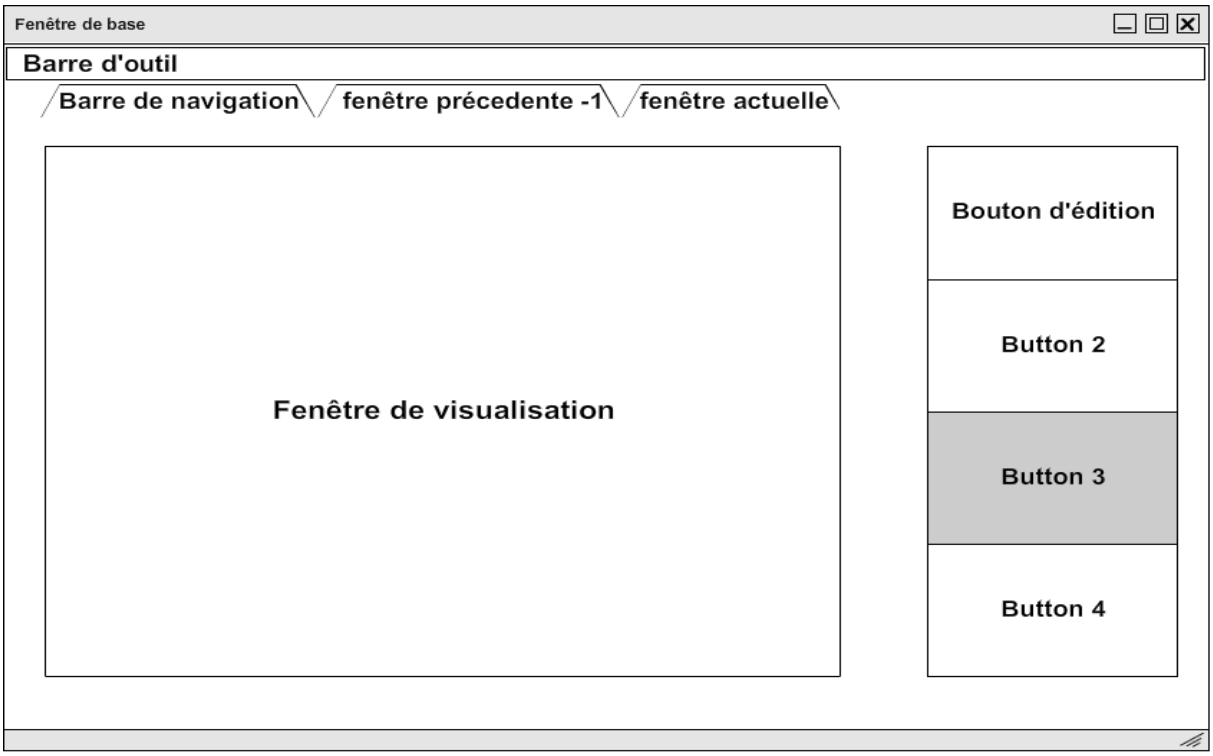


Figure 3 Maquette principale d'UVSP 2.0

2. Application Web

Nous avons choisi des technologies que nous n'avions jamais utilisées. Il nous a fallu donc prévoir une semaine de formation pour être opérationnels sur la technologie. Pendant cette semaine, la personne en charge de l'interface graphique a commencé son travail. Par conséquent, une fois que notre formation fut finalisée, nous avons pu nous pencher directement sur l'application en ayant une interface.

La première étape du développement de notre solution a été la mise en place de la base de données. Pour ce faire, nous avons utilisé le script de création que nous avions à notre disposition. L'IDE intègre un SQL server sur lequel nous avons exécuté le script. Par la suite, nous avons utilisé ADO .NET Entity Framework qui, à partir de la base de données, a généré les modèles de données qui permettent d'accéder à la base de données.

À partir de cette étape, nous avons pu séparer le travail entre les différents développeurs de notre application. Nous avons assigné à chaque développeur une partie de l'application. Chaque développeur était donc indépendant, et avait des parties du programme qui n'interfère en rien avec le travail des autres développeurs.

Pour synchroniser les travaux des différentes personnes et permettre de mieux gérer le projet dans son ensemble, nous avons mis en place une forge : Team Foundation Server. Cela nous a permis entre autres de gérer nos sources et le projet.

Ainsi chaque développeur a travaillé sur plusieurs contrôleurs, à générer des vues et, si cela était nécessaire, il a ajouté des modèles. La personne en charge de l'interface graphique à intégrer son travail sur chacune des vues crée par les développeurs. Par conséquent, les développeurs n'ont pas été gênés par l'avancement de l'intégration de l'interface graphique.

3. API

Dans notre projet « USPV », étant donné que nous avons choisi d'utiliser l'architecture ASP.NET MVC, nous avons décidé d'utiliser l'API Web, qui est fourni par Microsoft. Celui-ci nous affirme qu'elle est compatible avec .Net et qu'elle est parfaitement fonctionnelle.

Qu'est-ce qu'une API

API Web est un Framework pour construire des services Web sur.NET Framework. La raison pour laquelle nous avons choisi l'API Web plutôt que d'autres technologies comme SOAP, est que l'API Web est facile à apprendre, parfaitement compatible avec ASP.NET. Par ailleurs, la raison la plus importante est que l'architecture est similaire à .NET MVC4 qui a accéléré le développement de notre projet.

Comment fonctionne-t-il?

Comme l'architecture est similaire à ASP.NET MVC4, un contrôleur est une classe qui gère les requêtes HTTP. La différence est que le contrôleur ne renvoie pas forcément une vue. Au lieu de cela, nous utilisons des « JSON » comme « pont » entre contrôleur API et terminal Web (navigateur Web ou Mobile). En ce qui concerne le principe de routage (il faut configurer dans WebApiConfig.cs dans le dossier App_Start). Il existe plusieurs méthodes, dont 2 méthodes très utilisées :

1. Visiter le routage pour « api / contrôleur »
2. Visiter le routage pour « api / contrôleur / action »

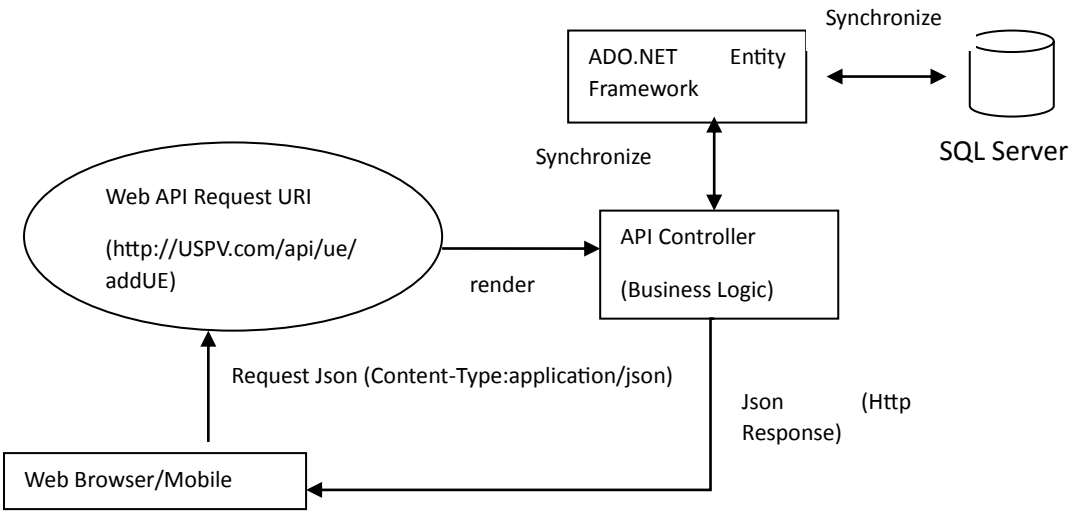
Nous avons configuré la deuxième méthode de routage, car il est beaucoup plus clair. Cette méthode prend également en charge le développement d'applications RESTful dans l'avenir. Son principe est de spécifier explicitement la méthode HTTP d'une action en décorant la méthode avec le HttpGet, HttpPut, HttpPost, ou avec un attribut HttpDelete.

Par exemple, nous pouvons utiliser l'API addUE en précisant : « http://USPV.com/api/ue/addUE »

```
[HttpPost]
   public HttpResponseMessage addUE (Ue ue) {}
```

Pour se connecter à notre base de données, nous utilisons également ADO.NET Entity Framework afin de gérer les opérations de base (Create, Read, Update, Delete).

Voici une présentation de la façon dont il fonctionne pour notre api web



Dans notre projet, le but de l'API Web est de faire des fonctionnalités supplémentaires non implémentées par l'application. Notre application ne dépend pas du service Web. Nous avons utilisé un outil appelé **Fiddler** pour tester les API web.

Notre API Web offre les services suivants :

Cours	Ajouter un cours
Enseignement	Ajouter un enseignement Supprimer un enseignement Visualisation de tous les enseignements
UE	Ajouter une UE Visualisation de toutes les UEs
Enseignant	Visualisation de tous les enseignants
Groupe	Visualisation de tous les groupes
Matière	Visualisation de toutes les matières
Réservation	Visualisation de toutes les réservations
TypeCour	Visualisation de tous les types de cours

Afin d'accélérer le processus de développement du projet, l'application ne dépend pas du web service. Toutefois, après avoir terminé les fonctionnalités principales, nous avons découvert qu'il y a quelques fonctions supplémentaires qui doivent être réalisées comme « enseignement complémentaire » ou « ajouter un cours ». Pour les fonctions simples, mais nécessaires, nous avons décidé d'utiliser l'API Web.

4. Test de l'application

Les tests permettent la vérification partielle du système mise en place. La norme utilisée est IEEE 829-1998 (standard des tests logiciels). Les différentes étapes sont : d'effectuer les tests, de diagnostiquer les différentes erreurs, leurs causes et de les corriger.

Phase de test manuel :

La correction des différentes erreurs visuelles consiste à vérifier l'affichage des images, le positionnement des icônes ou boutons… Cette phase permet d'uniformiser tout le site web. Ainsi le thème et les couleurs doivent être respectés et certaines pages ont dû être totalement modifiées pour améliorer l'ergonomie.

Phase de test unitaire :

Cette étape permet de s'assurer du bon fonctionnement d'une partie du programme. On vérifie ainsi que chaque fonction renvoi le bon résultat.

Conclusion des tests

Les tests ont permis de détecter et de corriger certains bogues. Il faut savoir qu'avec la méthode ExtremeProgramming, nous avons pu éviter beaucoup d'erreurs. De plus, ils nous ont permis d'épurer et d'uniformiser l'interface de notre application web.

Résultats

1. Avancements

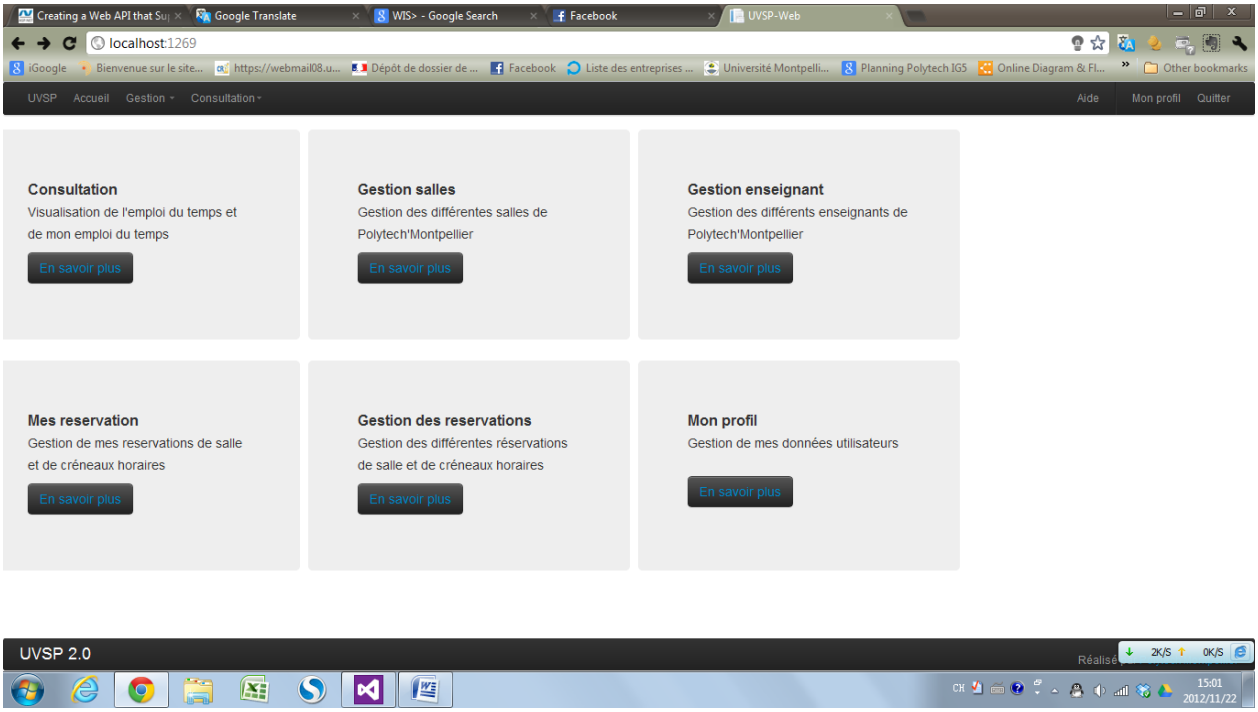
Les objectifs définis précédemment ont été atteints :

- la gestion des enseignants,
- la visualisation de l'emploi du temps des enseignants,
- La gestion des réservations,
- la gestion des salles.

De plus, nous avons pu mettre en place une sécurité ainsi que la mise en place de droits. Ainsi, un utilisateur étant seulement enseignant ne pourra gérer les enseignants.

Nous avons pu développer une API qui ajoute des fonctionnalités que nous n'avons pas développées dans notre application.

Pour illustrer tous ces propos, voici une capture d'écran de l'accueil de notre application



2. Dans l'avenir

Ce projet ne peut pas être utilisé, car certaines fonctionnalités manquent. En effet, il faudrait ajouter la possibilité de créer, par exemple, des matières, des UE ou des enseignements. Ces fonctionnalités sont possibles grâce à l'API. Il serait judicieux de les rendre aussi disponibles dans l'application. Cependant, les fonctions créent gère une grande partie de l'application et il est possible de les adapter pour créer les fonctionnalités manquantes.

Conclusion

Ce projet fut réalisé en tenant compte des contraintes et nous avons atteint les objectifs que nous nous étions fixés. Grâce à l'organisation mise en place tout au long du projet nous avons pu développer certains aspects que nous n'avions pas prévus initialement.

Ce projet nous a permis de développer de nouvelles compétences et d'apprendre de nouvelles technologies. Nous avons appris à gérer les contraintes et obtenir les résultats que nous avions définis. De même, nous avons été une nouvelle fois confrontée à la difficulté de gérer le travail en équipe. Cette étape est primordiale pour nous avenir professionnel.

Ce projet a donc été pour nous un succès.