****

*Réalisé par :*

*Brahim Mohamed Baheida*

*Mohamed Sidatt Alem*

*Encadré par :*

Dr. Ethmane ould Ebbe

Année Universitaire : 2016 – 2017

MEMOIRE

En vue de l’obtention

De Licence Professionnel en Informatique et développement

*D’Entreprises*

*République Islamique de Mauritanie*

*Honneur, Fraternité, Justice*

*Ministère chargée de l’Enseignement Supérieur*

*Institut Supérieur de Comptabilité d’Administration d’Entreprises*

# Dédicace

​ Nous dédions ce travail à :  
Nos mères, sources de tendresse et d’amours pour leurs soutiens tout le long de notre vie scolaire.  
Nos pères, qui nous ont toujours soutenus et qui ont fait tout possible pour nous aider.  
Nos frères et nos sœurs, que nous aimons beaucoup.  
Notre grande famille.  
Nos chers ami (e) s, et enseignants.  
Tous qui ont collaboré de près ou de loin à l’élaboration de ce travail.  
  
Que dieu leur accorde santé et prospérité.

# Remerciement

# Résumé

# Sommaire

[Dédicace 3](#_Toc483010402)

[Remerciement 4](#_Toc483010403)

[Résumé 5](#_Toc483010404)

[Sommaire 6](#_Toc483010405)

[Table de Figures 8](#_Toc483010406)

[Introduction 9](#_Toc483010407)

[I. Contexte du mémoire 10](#_Toc483010408)

[1. Présentation du projet : 10](#_Toc483010409)

[o Interaction : 10](#_Toc483010410)

[o Sécurité : 10](#_Toc483010411)

[o Etre responsive : 10](#_Toc483010412)

[o Optimisation des ressources : 10](#_Toc483010413)

[o Maintenabilité : 10](#_Toc483010414)

[2. Travail Demandé : 11](#_Toc483010415)

[3. Bilan : 11](#_Toc483010416)

[II. Analyse et conception : 12](#_Toc483010417)

[1. Etude préalable : 12](#_Toc483010418)

[2. Modélisation : 12](#_Toc483010419)

[a. Pourquoi Merise : 12](#_Toc483010420)

[b. JMerise : 12](#_Toc483010421)

[c. MCD : 12](#_Toc483010422)

[d. MLD : 18](#_Toc483010423)

[e. Pourquoi SQL Server : 18](#_Toc483010424)

[3. Les outils de développement : 19](#_Toc483010425)

[a. Visual Studio 2015: 19](#_Toc483010426)

[b. SQL Server Management Studio (SSMS 2016): 19](#_Toc483010427)

[c. GIT: 19](#_Toc483010428)

[d. Bitbucket: 19](#_Toc483010429)

[4. Les principes S.O.L.I.D : 20](#_Toc483010430)

[ S – Single Responsability Principle (SRP): 20](#_Toc483010431)

[ O – Open/Closed Principle (OCP): 20](#_Toc483010432)

[ L – Liskov Substitution Principle (LSP) : 20](#_Toc483010433)

[ I – Interface Segregation Principle (ISP): 20](#_Toc483010434)

[ D – Dependency Inversion Principle (DIP): 20](#_Toc483010435)

[5. L’architecture 3-tiers : 21](#_Toc483010436)

[III. Développement 22](#_Toc483010437)

[1. Couche D.A.L 22](#_Toc483010438)

[a) Entity Framework 22](#_Toc483010439)

[b) Repository Pattern : 23](#_Toc483010440)

[c) Log4Net 23](#_Toc483010441)

[2. Couche B.L.L 24](#_Toc483010442)

[a) Contraintes 25](#_Toc483010443)

[b) Complexité 25](#_Toc483010444)

[3. Couche Web 25](#_Toc483010445)

[a) Pourquoi MVC ? : 25](#_Toc483010446)

[b) Web APIs : 25](#_Toc483010447)

[c) Pourquoi Ninject ? : 25](#_Toc483010448)

[d) Contrôleurs : 26](#_Toc483010449)

[e) Vues : 27](#_Toc483010450)

[IV. Sécurité 28](#_Toc483010451)

[1. Pourquoi la sécurité ? 28](#_Toc483010452)

[2. C’est quoi OWASP ? 28](#_Toc483010453)

[3. OWASP TOP-10 : 28](#_Toc483010454)

[1. Oubli de valider les entrées des utilisateurs 28](#_Toc483010455)

[2. Contrôle d'accès inefficace 28](#_Toc483010456)

[3. Mauvaise gestion des sessions 28](#_Toc483010457)

[4. Cross Site Scripting (XSS) 29](#_Toc483010458)

[5. Dépassement de mémoire tampon 29](#_Toc483010459)

[6. Injection de commandes 29](#_Toc483010460)

[7. Mauvaise gestion des erreurs 29](#_Toc483010461)

[8. Mauvaise utilisation du chiffrement 29](#_Toc483010462)

[9. Failles dans l'administration distante 29](#_Toc483010463)

[10. Mauvaise configuration du serveur web et des applications 30](#_Toc483010464)

[Conclusion 31](#_Toc483010465)

[Annexe 32](#_Toc483010466)

[Bibliographie 33](#_Toc483010467)

# Table de Figures

[Figure 1 : Modèle Conceptuel de donnees 12](file:///E:\PFE\ISCAE\rapport.docx#_Toc483010231)

[Figure 2 : Modèle Logique de Donnees 18](file:///E:\PFE\ISCAE\rapport.docx#_Toc483010232)

[Figure 3 : Architecture 21](file:///E:\PFE\ISCAE\rapport.docx#_Toc483010233)

[Figure 4 : Architecture DAL 22](file:///E:\PFE\ISCAE\rapport.docx#_Toc483010234)

[Figure 5 : Architecture BLL 24](file:///E:\PFE\ISCAE\rapport.docx#_Toc483010235)

[Figure 6 : Architecture WEB 25](file:///E:\PFE\ISCAE\rapport.docx#_Toc483010236)

# Introduction

# Contexte du mémoire

## Présentation du projet :

La solution qu’on a proposée est basée sur le plateforme .NET. Offre les opérations traductionnelles du site web de l’ISCAE et ajoute des autres fonctionnalités qui sont :

### Interaction :

La solution offre un environnement d’échange d’information et de collaboration entre les utilisateurs, les étudiants peuvent ajouter des documents, discuter entre eux. Les professeurs peuvent ajouter des documents, envoyer des messages aux étudiants. L’administration est capable d’ajouté des avis, des résultats, la liste des étudiants, activer et désactiver des comptes.

### Sécurité :

Tout au long du développement de la solution on a considéré les recommandations de la communauté de sécurité des applications web. Le site OWASP offre une liste des vulnérabilités des applications web chaque année. Cette considération va être le sujet de la partie sécurité (page xxx).

### Etre responsive :

L’utilisation du Framework Bootstrap, de la bibliothèque JQuery et d’autres nous permet de réaliser une interface d’utilisation responsive.

### Optimisation des ressources :

La programmation générique, et l’utilisation de quelque patron de conception (Design Pattern) nous servi à optimiser les ressources par la diminution de nombre des codes pour réaliser des taches similaires en plusieurs classes, et le nombre des instances dans la RAM.

### Maintenabilité :

La maintenance des logiciels est divisée en trois types :

#### Maintenance corrective :

Elle consiste à corriger les défauts de fonctionnement ou les non-conformités d'un logiciel.

#### Maintenance adaptive :

Sans changer la fonctionnalité du logiciel, elle consiste à adapter l'application afin que celle-ci continue de fonctionner sur des versions plus récentes des logiciels de base, voire à faire migrer l'application sur de nouveaux logiciels de base (un logiciel de base étant un logiciel requis pour l'exécution d'une application ; exemples : système d'exploitation, système de gestion de base de données).

#### Maintenance évolutive :

Cela consiste à faire évoluer l'application en l'enrichissant de fonctions ou de modules supplémentaires, ou en remplaçant une fonction existante par une autre, voire en proposant une approche différente.

Notre solution suivre les principes S.O.L.I.D qui vont être décrites au chapitre suivant. Elle est divisée en plusieurs couches chaque couche est dédié pour des taches spécifiques ce qui facilite l’accès en cas d’erreurs aux codes responsables et les corriger. Elle permet aussi l’adaptation et l’évolution des fonctionnalités.

## Travail Demandé :

Application 3-tiers

## Bilan :

# Analyse et conception :

## Etude préalable :

## Modélisation :

Merise et UML sont deux grands principes de « traduction » ou modélisation d'un système d'information. Néanmoins, ils ne sont pas aussi proches qu'on pourrait le penser.

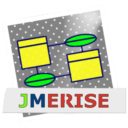
Le choix de l'un ou de l'autre se fait selon trois axes à savoir l'accessibilité, la précision et l'exploitabilité.

### Pourquoi Merise :



* MERISE présente l'intérêt d'avoir des modèles logiques moins détaillés facilement compréhensibles.
* MERISE est une méthode plus généraliste. Elle donne une vue globale de la solution sans autant entrer dans les petits détails.

### JMerise :

 **JMerise** est un logiciel dédié à la modélisation des modèles conceptuels de donnée pour Merise

**JMerise** permet les relations réflexives, la généralisation et la spécialisation des entités. Il génère le MLD et le script SQL Server, MySQL …etc.

### MCD :

Figure 1 : Modèle Conceptuel de données

#### Les Entités :

|  |  |  |  |
| --- | --- | --- | --- |
| Entité | Description | Attribue | Description |
| Administrateurs |  | AdministrateurId | Clé primaire |
| Nom | Nom complet de l’utilisateur |
| Login | Pseudo unique pour s’authentifier (username) |
| Password | Mot de passe |
| Telephone | Numéro du téléphone |
| Email | Adresse électronique |
| isActive | Compte capable de s’authentifier ou non |
| ProfilePath | L’emplacement de la photo de profile |
| Groupe | De quel groupe administrative |
| Annonces |  | AnnonceId | Clé primaire |
| Titre | Titre de l’annonce |
| Contenu | Contenu de l’annonce |
| DocumentOfficiel |  | DocumentOfficielId | Clé primaire |
| Titre | Titre du document |
| Emplacement | Emplacement du document |
| Type | Type de document(encodage) |
| DocumentNonOfficiel |  | DocumentNonOfficielId | Clé primaire |
| Titre | Titre du document |
| Emplacement | Emplacement du document |
| Type | Type de document(encodage) |
| Professeurs |  | ProfesseurId | Clé primaire |
| Nom | Nom complet du professeur |
| Login | Pseudo unique pour s’authentifier (username) |
| Password | Mot de passe |
| Telephone | Numéro du téléphone |
| Email | Adresse électronique |
| IsActive | Compte capable de s’authentifier ou non |
| ProfilePath | L’emplacement de la photo de profile |
| Etudiants |  | EtudiantId | Clé primaire |
| Nom | Nom complet d’étudiant |
| Matricule | Matricule |
| Login | Pseudo unique pour s’authentifier (username) |
| Password | Mot de passe |
| Telephone | Numéro du téléphone |
| Email | Adresse électronique |
| IsActive | Compte capable de s’authentifier ou non |
| NNI | Numéro national d’authentification |
| ProfilePath | L’emplacement de la photo de profile |
| Questions |  | QuestionId | Clé primaire |
| Contenu | Contenu de la question |
| Titre | Titre de la question |
| DateQuestion | Date d’ajout de la question |
| Attachement | Emplacement d’une image si elle existe |
| Reponses |  | ReponseId | Clé primaire |
| Contenu | Contenu de la réponse |
| DateReponse | Date d’ajout de la réponse |
| Resultats |  | ResultatId | Clé primaire |
| Path | Emplacement du document |
| Semester |  |
| Annee | Année universitaire |
| Specialites |  | SpecialiteId | Clé primaire |
| Designation | Désignation de la spécialité |
| Messages |  | MessageId | Clé primaire |
| Titre | Titre du message |
| Contenu | Contenu du message |
| Modules |  | ModuleId | Clé primaire |
| Designation | Designation du module |
| Notifications |  | NotificationId | Clé primaire |
| ActorId | ID le responsable de l’action |
| TargetId | ID le cible de la notification |
| Message | Message de la notification |
| NotificationStatus | L’état de la notification |
| TableName | Table responsable de la notification |
| RecordId | ID l’enregistrement du TableName responsable de la notification |
| DateNotification | Date de la notification |
| Recoveries |  | RecoveryId | Clé primaire |
| Matricule | Matricule l’étudiant |
| Email | Email l’étudiant |
| Code | Code de validation |

#### Les associations :

|  |  |  |  |
| --- | --- | --- | --- |
| Association | Entre | Cardinalités | Description |
| Avoir | Questions – Reponses | 0,n – 1,1 | Une question peut avoir plusieurs réponses et une seule réponse va concerner une seule question |
| Avoir | Resultats – Specialites | 1,1 – 0,n | Une spécialité peut avoir plusieurs résultats et un seul résultat va concerner une seule spécialité |
| Ajouter | Administrateurs – Resultats | 0,n – 1,1 | Un administrateur peut ajouter plusieurs résultats et un seul résultat va être ajouter par un seul administrateur |
| Ajouter | Administrateurs – Annonces | 0,n – 1,1 | Un administrateur peut ajouter plusieurs annonces et une seule annonce va être ajouter par un seul administrateur  L’attribue DateAjout représente la date de l’ajout de l’annonce |
| Inscrire | Etudiants – Specialites | 1,1 – 0,n | Un étudiant peut s’inscrire dans une seule spécialité et une spécialité peut avoir plusieurs inscriptions.  L’attribue Niveau représente le niveau de l’étudiant |
| Concerner | Messages – Specialites | 1,1 – 0,n | Un message peut concerner une seule spécialité et une spécialité peut recevoir plusieurs messages.  L’attribue Niveau représente le niveau de la spécialité |
| Faire | Etudiants – Reponses | 0,n – 1,1 | Un étudiant peut faire plusieurs réponses et une réponse va être écrite par un seul étudiant |
| Poser | Etudiants – Questions | 0,n – 1,1 | Un étudiant peut poser plusieurs questions et une question va être poser par un seul étudiant |
| Envoyer | Professeurs – Messages | 0,n – 1,1 | Un professeur peut envoyer plusieurs messages et un message va être envoyer par un seul professeur L’attribue DateEnvoiMessage représente la date de l’envoi u message |
| AjoutNonOfficiel | Etudiants – DocumentNonOfficiels | 0,n – 1,1 | Un étudiant peut ajouter plusieurs documents et un document va être ajouter par un seul étudiant L’attribue DateAjoutNonOfficiel représente la date de l’ajout du document |
| AjoutOfficiel | Professeurs – DocumentOfficiels | 0,n – 1,1 | Un professeur peut ajouter plusieurs documents et un document va être ajouter par un seul professeur L’attribue DateAjoutOfficiel représente la date de l’ajout du document |
| ProfesseurModule | Professeurs – Modules | 0,n – 0,n | Un professeur peut enseigne plusieurs modules et un module peut être enseigner par plusieurs professeurs |
| SpecialiteModule | Specialites – Modules | 0,n – 0,n | Une spécialité peut avoir plusieurs modules et un module peut être enseigner en plusieurs spécialités |
| ProfesseurSpecialite | Professeurs – Specialites | 0,n – 0,n | Un professeur peut enseigne plusieurs spécialités et une spécialité peut être enseigner par plusieurs professeurs |
| DocumentNonOfficielModules | DocumentNonOfficiels – Modules | 1,1 – 0,n | Un document peut être ajouter pour un seul module et un module peut avoir plusieurs documents |
| DocumentOfficielModules | DocumentOfficiels – Modules | 1,1 – 0,n | Un document peut être ajouter pour un seul module et un module peut avoir plusieurs documents |

### MLD :

Figure 2 : Modèle Logique de Données

### Pourquoi SQL Server :



Lors du choix d’un système de gestion de bases de données de nombreuses sociétés se concentrent sur les bénéfices initiaux, mais ne réalisent pas les avantages à plus long terme. La scalabilité, la redondance, la génération de rapports et la synchronisation sont des caractéristiques dont vous pouvez profiter dès l’installation avec Microsoft SQL Server 2016

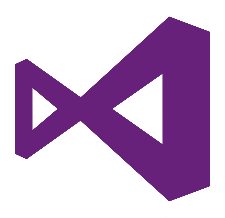
## Les outils de développement :

Un outil de développement est un logiciel qui aide un développeur dans le déroulement d’une activité de développement.

Dans le cas général, les outils de développement nous aident à :

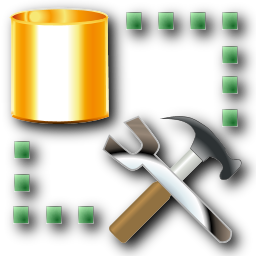
* Implanter une phase d’un processus de développement logiciel.
* Automatiser des taches importantes *et* ennuyeux.
* Être plus efficace.

### Visual Studio 2015:



Visual Studio s’ouvre à toutes les tendances du moment, et devient un véritable « couteau suisse » dans le monde du développement, qui pourra satisfaire divers profils de développeurs et permettre de développer tout type d’application : développement Cloud, développement mobile, développement Web, développement multiplateforme, développement desktop, etc.

### SQL Server Management Studio (SSMS 2016):



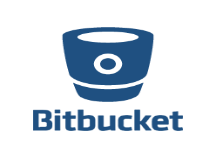
SQL Server Management Studio est un environnement intégré qui permet d'avoir accès, de configurer, de gérer, d'administrer et de développer tous les composants de SQL Server. SQL Server Management Studio associe un groupe d'outils graphiques à des éditeurs de script performants pour permettre aux développeurs de tous les niveaux de compétence d'avoir accès à SQL Server.

### GIT:



C’est un logiciel de gestion de versions très puissant et récent, il a été créé par Linus Torvalds, qui est entre autres l’homme à l’origine de Linux. Il se distingue par sa rapidité et sa gestion des branches qui permettent de développer en parallèle de nouvelles fonctionnalités.

### Bitbucket:



Bitbucket est un [service web](https://fr.wikipedia.org/wiki/Service_web) d'hébergement et de gestion de [développement logiciel](https://fr.wikipedia.org/wiki/D%C3%A9veloppement_de_logiciel) utilisant les [logiciels de gestion de versions](https://fr.wikipedia.org/wiki/Logiciel_de_gestion_de_versions) [Git](https://fr.wikipedia.org/wiki/Git) et [Mercurial](https://fr.wikipedia.org/wiki/Mercurial).

Nous préférons bitbucket, pour les raisons suivantes :

* Compte gratuit de 1 à 5 personnes
* Repository privée gratuit
* Intégration avec Jira et hipchat (Atlassian)

## Les principes S.O.L.I.D :

SOLID est un acronyme introduit au début des années 2000 par Robert Cecil Martin connu par Uncle (l’oncle) Bob, résume 5 principes clés à appliquer en POO pour produire du code facile à maintenir.

### S – Single Responsability Principle (SRP):

Chaque objet est en charge d'une seule *responsabilité*, laquelle doit être complètement encapsulée dans la classe.

Responsabilité = raison de changer (R. C. Martin).

Une classe ne doit être susceptible de changer que pour une seule raison.

### O – Open/Closed Principle (OCP):

Les entités logicielles (classes, modules, fonctions, etc.) doivent être ouvertes aux extensions, mais fermées aux modifications.

On utilise au maximum les interfaces, figées. Mais elles peuvent être implémentées librement et augmentées par héritages.

### L – Liskov Substitution Principle (LSP) :

Concrètement, là ou un objet de type **T** est attendu, on doit pouvoir passer un objet de type **Y** sans que le comportement dudit objet soit changé. C’est-à-dire que tout invariant inhérent à la classe **T** être respecte dans ses classes filles.

LSP permet d’assurer que notre hiérarchie suit un fonctionnement logique et cohérent.

### I – Interface Segregation Principle (ISP):

Une interface doit ne comporter que des méthodes en rapport avec l’interface elle-même, de façon à ce que les clients d’une interface ne connaissent que les méthodes en rapport avec cette interface.

### D – Dependency Inversion Principle (DIP):

Dans les architectures classiques, les composants de haut niveau dépendent des composants de bas niveau sur lesquels ils reposent. Le **DIP** établit au contraire que :

* Les modules de haut niveau ne doivent pas dépendre des modules de bas niveau. Les deux doivent dépendre de l’abstractions.
* Les abstractions ne doivent pas dépendre de détails, mais les détails doivent dépendre des abstractions.

## L’architecture 3-tiers :

Figure 3 : Architecture

Son nom provient de l'[anglais](https://fr.wikipedia.org/wiki/Anglais) « ***tier »*** signifiant étage ou niveau. Il s'agit d'un modèle logique d'architecture applicative qui vise à modéliser une [application](https://fr.wikipedia.org/wiki/Application_%28informatique%29) comme un empilement de trois couches [logicielles](https://fr.wikipedia.org/wiki/Logiciel) (ou niveaux, étages, tiers) dont le rôle est clairement défini :

* **Présentation** des données, correspondant à l'affichage, la restitution sur le poste de travail, le dialogue avec l'utilisateur, dans notre cas le couche web.
* Le **traitement métier** des données, correspondant à la mise en œuvre de l'ensemble des règles de gestion et de la logique applicative, dans notre cas couche BLL.
* L'**accès aux données** [persistantes](https://fr.wikipedia.org/wiki/Persistance_%28informatique%29) : correspondant aux données qui sont destinées à être conservées sur la durée, voire de manière définitive, dans notre cas couche DAL.

# Développement

## Couche D.A.L

Figure 4 : Architecture DAL

La DAL c’est l’abréviation de Data Access Layer en français couche d’accès aux données n'aura qu'un seul rôle, celui de gérer la relation entre la source de données et la couche métier. Elle reçoit les entités et selon la méthode du CRUD appelée, les données sont insérées, chargées, mises à jour ou effacées.

### Entity Framework

#### **Qu’est-ce que O/RM :**

ORM est un outil pour stocker des données depuis des objets de domaine vers une base de données relationnelle comme MS SQL Server, de manière automatisée, sans beaucoup de programmation. O / RM comprend trois parties principales : objets de classe de domaine, objets de base de données relationnels et informations de mappage sur la façon dont les objets de domaine correspondent à des objets de base de données relationnelles (tables, vues et procédures stockées). ORM nous permet de garder notre conception de base de données distincte de notre conception de classe de domaine. Cela rend l'application maintenable et extensible. Il automatise également l'opération CRUD standard (Créer, Lire, Mettre à jour et Supprimer) afin que le développeur n'a pas besoin de l'écrire manuellement.

Le plateforme .NET offre plusieurs framework O/RM comme NHibernate, Entity framework …etc.

#### **Qu’est-ce que Entity framework**

Entity Framework représente la solution tant attendue de mapping objet-relationnel de Microsoft. Elle constitue une bonne alternative à des outils comme NHibernate, grâce à son architecture ingénieuse du modèle EDM et des entités.

Ce framework comprend trois approches :

* Model first : Consiste à créer un modèle de données en premier, puis à générer une base de données. Le designer EF permet de créer toutes les entités, associations, propriétés, … que l’on souhaite voir en base de données.
* Database first : C’est L’approche la plus utilisée, la plus simple et la plus granulaire. Elle offre un gain de temps considérable aux développeurs. Consiste à créer une base de données en premier puis à générer le modèle de données. Elle permet de sélectionner tout ou partie des éléments de la base de données pour générer leur pendant objet.
* Code first : Consiste à générer et de faire évoluer le schéma de la base de données en se basant sur les classes des entités que l’on manipule dans le projet.

### Repository Pattern :

C’est un design pattern assez intéressant puisqu’il permet à la fois de détacher l’implémentation des entités de sa persistance (ou de son stockage), et simplifier les tests unitaires.

Outre le fait que le pattern repository fournis un code de meilleure qualité, il facilite grandement le changement **d'ORM** et nous permet d'effectuer des tests unitaires en passant desobjets nommés **"mock**" à notre repository.

IMAGE ici …

### Log4Net

Les logs se présentent en général sous la forme de fichiers texte, d'enregistrements en base ou de mails.

Log4Net est un outil pour aider le développeur à logger des informations vers diverses cibles. Log4net est un port de l'excellent framework log4j vers .NET.

Log4net a trois composants principaux :

* **Les loggers** : Définissent le type de message ainsi que son niveau
* **Les appenders** : Destination du message
* **Les layouts** : Formatage du message

Log4net possède 2 configurateurs : le BasicConfigurator et le XmlConfigurator. Nous avons choisi d’utiliser la configuration par xml puis XmlConfigurator permet de charger une configuration définie dans le fichier xml séparé. Cela permet de modifier la façon dont les informations sont loggées sans avoir à recompiler l'application.

## Couche B.L.L

Figure 5 : Architecture BLL

Nous allons recevoir de la part de ma couche web, toutes les saisies des utilisateurs sur ses critères de recherche (qui peuvent être vide) et y faire les vérifications nécessaire (du style : une date doit être une date, un numérique doit être numérique, un champ ne peut être vide...) et une fois ces vérifications terminer nous allons envoyer ces paramètres à la couche DAL qui nous retournera le résultat que nous revoyions à la couche web.

### Contraintes

### Complexité

## Couche Web

Figure 6 : Architecture WEB



ASP.NET est la plateforme (ensemble de composants permettant de construire une application web) de développement de Microsoft permettant la réalisation d'applications web. ASP.NET nous propose deux logiques pour construire une application web : WebForms et MVC.

### Pourquoi MVC ? :

Nous avons choisi ASP.NET MVC parce que elle offre un cadre de réalisation des applications web, grâce à ce patron de conception réputé ; une espèce de ligne directrice qui va nous guider dans la réalisation d'une application web. Mais ASP.NET MVC donne également plus de liberté dans le rendu du HTML en permettant notamment d’utiliser des bibliothèques Javascript externes afin d’améliorer l’expérience utilisateur et de créer des applications web 2.0 puissantes, grâce à l’AJAX.

### Web APIs :

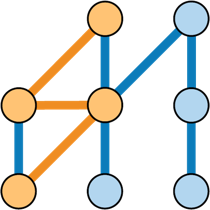
Web API est un framework pour la création et la consommation de services HTTP qui permet de rejoindre un nombre illimité de clients : Navigateur web, téléphone, tablettes … Nous pouvons utiliser XML ou JSON (les 2 plus utilisés) avec notre API, mais aussi d’autre type. JSON est efficace pour les applications mobiles avec des connexions lentes par exemple. Vous pouvez aussi appeler l’API à partir de jQuery.

### Pourquoi Ninject ? :

Les conteneurs IoC sont des outils spécifiquement conçus pour faciliter l'injection de dépendances. Originellement, ces utilitaires viennent du monde Java, mais, au fil de ces dernières années, des conteneurs IoC propres au framework .NET sont apparus, issus du monde open source aussi bien que de Microsoft.

**Spring.NET** a l'avantage, sur les autres conteneurs, de la maturité. C'est en effet un portage en .NET de Spring en Java.

Il permet de faire tout ce qu'on peut vouloir demander à un framework IoC, mais souffre de son âge (pas de typage générique), toute la configuration et la récupération des dépendances est basée sur des chaines, ce qui peut entrainer des erreurs à l'exécution.

**StructureMap** est un conteneur open source, dont le développement est en cours depuis quatre ans (soit le plus vieux framework en dehors de Spring). Contrairement à Spring, il ne s'agit pas d'un portage d'un projet Java existant, mais bien d'un framework pensé et développé pour l'inversion de contrôle en .NET.



**Ninject** est une framework d'injections de dépendances, d'IOC, c’est un petit framework rapide à prendre en main et très limité. Mais il est loin d'être limité.

Pour l’utilisation de Ninject on a téléchargé la bibliothèque Ninject et on a créé une classe appelée NinjectWebCommon et dans cette classe on a appelé des assemblies de la bibliothèque pour injecté les dépendances au lancement de l’application et pour libéré la mémoire lors d’arrêt. Ensuite on a ajouté la correspondance pour chaque interface.

### Contrôleurs :

Le contrôleur est au cœur de MVC, il fait le lien entre la vue et le modèle ; c’est lui qui gère les actions de l’utilisateur. Il interprète la requête HTTP entrante et choisit la vue à afficher dans le navigateur.

Dans notre solution nous avons créé dix contrôleurs :

|  |  |  |
| --- | --- | --- |
| Contrôleur | Description | Vue(s) |
| Administrateur | Gère les actions des administrateurs | Index, AddEtudiants, Etudiants, AddAvis, UserProfile, Professeurs, AddProfesseur, Specialites, AddSpecialite |
| Annonce | Gère l’affichage et le téléchargement des résultats et des avis | Index, Avis |
| Discussion | Gère la discussion entre les étudiants | Index, Discussion |
| Document | Gère le partage des documents | Index, Officiel, NonOfficiel |
| Emploi | Gère l’emploi du temps |  |
| Error | Gère les exceptions http | NotFound, Forbidden |
| Etudiant | Gère les actions des étudiants | Index, UserProfile, Module |
| Home | Gère les actions des visiteurs anonymes | Index, Register, Directeur, Avis, Formation, Etudiants, Professeurs |
| Message | Gère les messages entre professeurs et étudiants | Index, Boite |
| Professeur | Gère les actions des professeurs | Index, UserProfile |

### Vues :

#### Layout :

Le principe est de regrouper, dans une page spécifique, les éléments communs de mise en page pour le site (tels que le menu, l'entête, le pied de page, ou autres scripts JavaScript utilisés par toutes les pages).

Nous avons comme pages layout : HomeLayout, AdministrateurLayout, ProfesseurLayout, EtudiantLayout, RegisterLayout

#### Vue partielle :

Les vues partielles viennent avec un helper qui permet de renvoyer une vue sous la forme d’une chaîne de caractères. Cette chaîne peut ainsi être utilisée à l’intérieur d’une autre vue pour produire du HTML qui se répète.

#### De l’application :

Les images ici …

# Sécurité

## Pourquoi la sécurité ?

Les plates-formes hébergeant les applications Web sont de mieux en mieux sécurisées. En particulier, l’utilisation d’un firewall est maintenant considérée comme une condition *sine qua non*.

Cependant, la présence d'un firewall procure un faux sentiment de sécurité, car il est trop souvent perçu comme la panacée des solutions de sécurité. Or il ne couvre que certains aspects de la protection globale que nécessite une application Web complexe ouverte sur Internet. En particulier, certaines attaques ne peuvent être arrêtées par un firewall, pour la simple raison qu’elles sont noyées au milieu des requêtes de pages Web, qui sont bien sûr autorisées à travers le firewall.

Nous allons, dans les paragraphes qui suivent, définir les différentes sortes d’attaques auxquelles une application web peut être sensible, ainsi que les parades respectives permettant de sécuriser une telle application.

## C’est quoi OWASP ?

L'OWASP (Open Web [Application](http://www.futura-sciences.com/sciences/definitions/mathematiques-application-13200/) Security Project) est un projet Open Source. Il réunit des experts bénévoles autour d'un objectif commun : éduquer les professionnels du [web](http://www.futura-sciences.com/tech/definitions/internet-internet-3983/) en [matière](http://www.futura-sciences.com/sciences/definitions/matiere-matiere-15841/) de sécurité. Le groupe vient pour cela de publier la liste des dix [failles](http://www.futura-sciences.com/planete/definitions/structure-terre-faille-1065/) (Une faille c'est en fait une faiblesse dans un code qui peut être exploitée pour détourner un site de sa fonction première) les plus courantes sur le web. Elles sont pour la plupart connues depuis très longtemps mais, étrangement, de nombreux projets web en sont encore victimes aujourd'hui.

## OWASP TOP-10 :

### Oubli de valider les entrées des utilisateurs

#### **Définition :**

Un classique, qui permet aux pirates de faire accepter des commandes au [serveur](http://www.futura-sciences.com/tech/definitions/internet-serveur-1950/) à travers un formulaire web ou une simple URL, ou d'exécuter des contenus dynamiques (Javascript, par exemple) chez les autres utilisateurs d'un site.

#### **Sécurité :**

### [Contrôle d'accès](http://www.futura-sciences.com/tech/definitions/informatique-controle-acces-1678/) inefficace

#### **Définition :**

Mauvaise mise en œuvre des outils de contrôle d'accès (fichier .htpasswd lisible par tous, [mots de passes](http://www.futura-sciences.com/tech/definitions/informatique-mot-passe-1765/) nuls par défaut, …etc.).

#### **Sécurité :**

### Mauvaise gestion des sessions

#### **Définition :**

Cela permet aux pirates de "voler des sessions" d'autres utilisateurs.

#### **Sécurité :**

### [Cross Site Scripting](http://www.futura-sciences.com/tech/definitions/internet-cross-site-scripting-4657/) (XSS)

#### **Définition :**

Un autre grand classique, lui aussi lié à un manque de contrôle des entrées de l'utilisateur. Cette faille touche les sites web qui laissent les internautes publier du code [HTML](http://www.futura-sciences.com/tech/definitions/internet-html-480/) susceptible d'être vu par les autres utilisateurs du site (dans un forum, par exemple). Cela permet d'exécuter des contenus dynamiques sur les [navigateurs](http://www.futura-sciences.com/tech/definitions/internet-navigateur-3986/) des internautes, avec les droits associés au site web.

#### **Sécurité :**

### Dépassement de [mémoire tampon](http://www.futura-sciences.com/tech/definitions/informatique-memoire-tampon-574/)

#### **Définition :**

Une faille vieille comme le monde, qui frappe certains langages de programmation plus que d'autres (le C, par exemple). Si des composants CGI sont (mal) écrits dans ces langages, il peut être simple de compromettre totalement le serveur par une telle attaque.

#### **Sécurité :**

### Injection de commandes

#### **Définition :**

Là encore, la source de la faille est un manque de contrôle des entrées de l'utilisateur. Elle permet au pirate de faire exécuter des commandes au serveur (au [système d'exploitation](http://www.futura-sciences.com/tech/definitions/informatique-systeme-exploitation-11820/) ou à un serveur [SQL](http://www.futura-sciences.com/tech/definitions/informatique-sql-2524/), par exemple) en les attachant à une entrée web légitime avant que celle-ci ne soit transmise au serveur.

#### **Sécurité :**

### Mauvaise gestion des erreurs

#### **Définition :**

Les messages d'erreur utiles aux développeurs le sont souvent aussi pour les pirates ! Il faut donc penser à les supprimer une fois le développement terminé.

#### **Sécurité :**

### Mauvaise utilisation du [chiffrement](http://www.futura-sciences.com/tech/definitions/tech-chiffrement-1722/)

#### **Définition :**

La mise en œuvre du chiffrement au sein des applications web se révèle ardue. Des développeurs non spécialisés peuvent commettre des erreurs difficiles à déceler et créer ainsi une protection illusoire.

#### **Sécurité :**

### Failles dans l'administration distante

#### **Définition :**

C'est la voie royale : si les pages réservées aux administrateurs du site ne sont pas réellement protégées ([authentification forte](http://www.futura-sciences.com/tech/definitions/securite-authentification-forte-15025/) du client, chiffrement, contrôles réguliers...), un pirate peut prendre le contrôle du site sans avoir à pirater le serveur. Une aubaine, en quelque sorte.

#### **Sécurité :**

### Mauvaise configuration du serveur web et des applications

#### **Définition :**

Un classique : le serveur web qui permet de lister n'importe quel répertoire, ou les outils de développement qui laissent des versions temporaires des fichiers, lisibles par tout le monde. Avant de mettre un serveur en ligne, il est bon de faire le ménage et de bien comprendre toutes les options de ses fichiers de configuration...

#### **Sécurité**

# Conclusion

# Annexe

# Bibliographie