Table des matières

[Introduction Générale 1](#_Toc33010989)

[Chapitre 1. Présentation du cadre de stage 2](#_Toc33010990)

[Introduction 3](#_Toc33010991)

[Conclusion 3](#_Toc33010992)

[Chapitre 2. Spécification des besoins 4](#_Toc33010993)

[Introduction 5](#_Toc33010994)

[1 Spécifications des besoins 5](#_Toc33010995)

[1.1 Les besoins fonctionnels 5](#_Toc33010996)

[1.2 Les besoins non fonctionnels 5](#_Toc33010997)

[Conclusion 6](#_Toc33010998)

[Chapitre 3. La Conception 7](#_Toc33010999)

[Introduction 8](#_Toc33011000)

[1 Méthode et Outils de Modélisation 8](#_Toc33011001)

[1.1 Langage de modélisation 8](#_Toc33011002)

[1.2 L’outils de modélisation 9](#_Toc33011003)

[2 Identification des acteurs et des cas d’utilisation 9](#_Toc33011004)

[3 Diagramme de cas d’utilisation 10](#_Toc33011005)

[3.1 Diagramme de cas d’utilisation global 10](#_Toc33011006)

[3.2 Diagramme de cas d’utilisation de l’utilisateur 10](#_Toc33011007)

[3.2.1 Description du cas d’utilisation gestion des Questionnaire 11](#_Toc33011008)

[4 Diagrammes des séquences 12](#_Toc33011009)

[4.1 Etude de quelques diagrammes des séquences 13](#_Toc33011010)

[4.1.1 Diagramme de Séquences Ajouter un Questionnaire 13](#_Toc33011011)

[4.1.2 Diagramme de Séquences Modifier un questionnaire 14](#_Toc33011012)

[5 Diagrammes de Classes 15](#_Toc33011013)

[6 La base de Données 15](#_Toc33011014)

[6.1 Schéma de la base de données 15](#_Toc33011015)

[6.2 Conception architecturale 16](#_Toc33011016)

[Conclusion 17](#_Toc33011017)

[Chapitre 4. La Réalisation 18](#_Toc33011018)

[Introduction 19](#_Toc33011019)

[1.1 Pourquoi utiliser un Framework Angular 19](#_Toc33011020)

[1.1.1 Le Framework Angular 19](#_Toc33011021)

[1.1.2 Architecture Angular 20](#_Toc33011022)

[1.2 Architecture du système et environnement de test 20](#_Toc33011023)

[1.2.1 Environnement matériel 21](#_Toc33011024)

[1.2.2 Environnent logiciel 21](#_Toc33011025)

Liste des figures

[Figure 1 Langage de Modélisation Unifié 8](#_Toc32541302)

[Figure 2 Outils de Modélisation Lucidchart 9](#_Toc32541303)

[Figure 3 Diagramme de cas d'utilisation Global 10](#_Toc32541304)

[Figure 4 Diagramme de cas d'utilisation de l'utilisateur 11](#_Toc32541305)

[Figure 5 Diagramme de Séquence Ajouter un Questionnaire 14](#_Toc32541306)

[Figure 6 Diagramme de Séquence Modifier un questionnaire 15](#_Toc32541307)

[Figure 7 Diagramme de Classe 16](#_Toc32541308)

[Figure 8 Représentation Arborescente de la base de données 17](#_Toc32541309)

[Figure 9 Pattern MVC 18](#_Toc32541310)

[Figure 10 Le Framework Angular 21](#_Toc32541311)

[Figure 12 Visuel Studio Code 23](#_Toc32541312)

[Figure 13 Plateforme Logicielle Node.JS 23](#_Toc32541313)

[Figure 11 Le Framework Express.JS 24](#_Toc32541314)

[Figure 14 Le SGBD MongoDB 24](#_Toc32541315)

[Figure 15 L'application Postman 25](#_Toc32541316)

[Figure 16 Le Framework Bootstrap 25](#_Toc32541317)

Liste des tableaux

[Tableau 1 Liste des acteurs dans l'application 8](#_Toc32469334)

[Tableau 2 Identification de l'acteur et des cas d'utilisation. 8](#_Toc32469335)

[Tableau 3 Description du cas d'utilisation Gestion des Questionnaire 11](#_Toc32469336)

Dédicaces

A mes parents

Pour tous les sacrifices qu’ils ont faits et pour tout le soutien qu’ils ont offert tout au long de mes études.

J’espère qu’ils puissent trouver dans ce modeste travail un témoignage d’amour et d’affection envers eux.

A mes amis et mes collègues

Pour leur encouragement et pour tous les bons moments qu’on a vécus ensemble.

J’espère que notre amitié dura éternellement

Hadj Hassine Jawher

Remerciement

Avant tout développement sur cette expérience professionnel, il apparaît opportunité de commencer ce rapport de stage par des remerciements, à ceux qui m’ont beaucoup appris au cours de ce stage, et même à ceux qui ont eu la gentillesse de faire de ce stage un moment très profitable.

Aussi, je remercie Mme Ayedi Hedayat et Mr Khayati Walid d’avoir accepté ma demande de stage, et Mr Ben Achour Saïf, mon maître de stage qui m’a formé et accompagné tout au long de cette expérience professionnelle avec beaucoup de patience et de pédagogie. Enfin, je remercie l’ensemble des employés de la Start-Up Dévolution Studio pour les conseils qu’ils ont pu me prodiguer au cours de ce mois.

# Introduction Générale

Dans le cadre de la formation des futurs techniciens supérieurs, il est demandé d’effectuer un stage de perfectionnement au sein d’une entreprise d’une durée de 1 mois, et ce pour manipuler et appliquer les connaissances que j’ai acquis pendant mes études en technologies informatique à l’Institut Supérieur des Etudes Technologique de Radés.

L’organisme qui m’a accueilli est la Start-Up Dévolution Studio qui m’a proposé de faire une application web de gestion des Questionnaires.

Au cours de ce stage, j’ai pu m’intéresser au fonctionnement de cette direction aussi que développer mes compétences, il s’agit de faire la conception, de bien choisir les technologies web à utiliser, développer l’application, et chercher les solutions en cas des problèmes.

Plus largement, ce stage a été l’opportunité pour moi d’obtenir l’expérience nécessaire, développer mes connaissances sur ce secteur et d’être au courant des nouvelles politiques adoptées.

Au-delà d’enrichir mes connaissances professionnelles, ce stage m’a permis de comprendre dans quelle mesure il m’a beaucoup apporté dans mon futur parcours professionnel.

## **Chapitre 1. Présentation du cadre de stage**

# Introduction

Dans ce premier chapitre et avant de commencer nous présentons généralement l’entreprise d’accueil.

Tout d’abord, nous identifions l’organisme, ces services, et ces activités.

# Conclusion

## 

## **Chapitre 2. Spécification des besoins**

# Introduction

Après avoir présenté le lieu de stage ces activités, et ces services dans ce chapitre, nous allons spécifier les besoins nécessaires pour la réalisation du projet. C’est pour cela ont vas met l’accent sur les besoins fonctionnels ainsi que les besoins non fonctionnels pour éviter le développement d’une application non satisfaisante.

# Spécifications des besoins

Un acteur joue le rôle d’une entité externe (utilisateur humain, dispositif matériel, ou autre système) qui interagit directement avec le système étudier. L’acteur représenté dans notre application peut être soit un administrateur, soit un visiteur.

## Les besoins fonctionnels

* Consulter les informations des questionnaires enregistrer dans la base de données
* Gérer des questionnaires :
* Poser des nouveaux questionnaires
* Supprimer des questionnaires déjà existant
* Modifier des questionnaires

## Les besoins non fonctionnels

Les besoins non fonctionnels décrivent les objectifs liés aux performances du système et aux contraintes de son environnement. Ses exigences techniques sont souvent exprimées sous formes d’objectif spécifiques que doit atteindre le système.

* Assurer une cohérence et une crédibilité aux informations stockées dans la base de données.
* L’ergonomie des interfaces web
* La sécurité : les informations doivent être sécurités
* La fiabilité : l’application doit être fiable
* La rapidité : temps d’exécutions
* La portabilité : l’application doit être portable
* Efficacité : l’application utilise les ressources utilement
* Facile à manipule
* Interface agréable
* Facile à maintenir

# Conclusion

## **Chapitre 3. La Conception**

# Introduction

Penser avant d’agir, faire des plans avant de construire, concevoir d’abord, développer ensuite c’est la démarche qui doit être suivre lors du développement d’une application et pour réussir n’importe quel projet.

En effet, la conception d’un système informatique est une étape très importante qui va influencer la qualité et la fiabilité de toute application.

D’abord, nous allons commencer ce chapitre par l’explication du modèle de cycle de vie de projet qu’on a choisi. Ensuite nous allons passer à la partie de conception détaillée où nous présentons l’architecture globale de l’application. Enfin nous citons les différents diagrammes de cas d’utilisation, de séquences et le diagramme de classes.

# Méthode et Outils de Modélisation

## Langage de modélisation

Pour la conception de notre système nous avons adopté une méthode orientée. En effet cette dernière est une approche incontournable dans le cadre du développement des applications.

Pour mieux présenter l’architecture de notre système, on va choisir le langage de modélisation le plus adoptée UML :

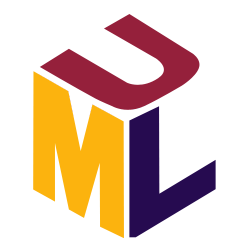


Figure 1 Langage de Modélisation Unifié

C’est un langage de modélisation, défini comme une norme de modélisation objet qui sert à décrire et à documenter un système d’information.

En utilisant ce langage, les objectifs de la modélisation objet suivant sont assurés :

* Formaliser la conception d’application.
* Faciliter la communication entre les différents intervenants au sein d’un projet informatique.
* Coordonner les activités entre les différents intervenants.
* Gérer l’évolution d’un projet informatique.
* Proposer des outils standardisés prenant en compte de nombreux aspects de la conception.

## L’outils de modélisation

Lucidchart est une plateforme de collaboration, basée sur le cloud, permettant la création de diagrammes et la visualisation de données, et autres cartes heuristiques et schémas conceptuels.



Figure 2 Outils de Modélisation Lucidchart

# Identification des acteurs et des cas d’utilisation

Les acteurs sont les entités externes qui interagissent directement avec le système et communiquent avec ce dernier par émission et réception de messages.

Notre système présente une seule partie : une application web qui concerne un seul acteur :

|  |  |
| --- | --- |
| Type d’application | Acteur |
| Web | Utilisateur (Client, Admin) |

Tableau 1 Liste des acteurs dans l'application

L’acteur et les cas d’utilisations sont résumés dans le tableau suivant :

|  |  |
| --- | --- |
| Acteur | Cas d’utilisation |
| Client Admin | Consulter des Questionnaires |
| Gérer des Questionnaires |
| Contacter le développeur |

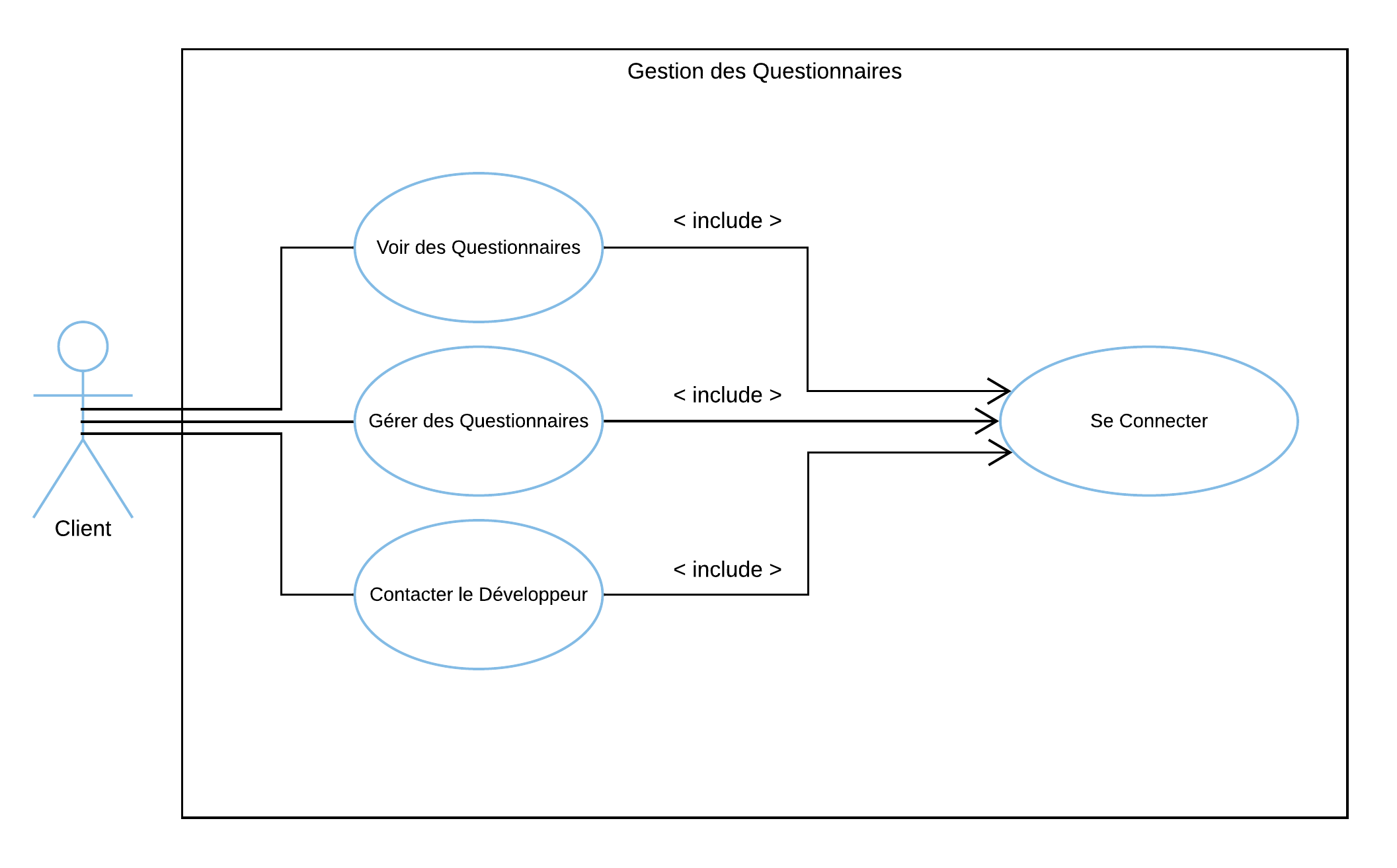
Tableau 2 Identification de l'acteur et des cas d'utilisation.

# Diagramme de cas d’utilisation

Un diagramme de cas d’utilisation est un graphe d’acteurs, un ensemble de cas d’utilisation englobés par la limite du système, des relations de communication entre les acteurs et les cas d’utilisation, et des généralisations de ces cas d’utilisation.

## Diagramme de cas d’utilisation global

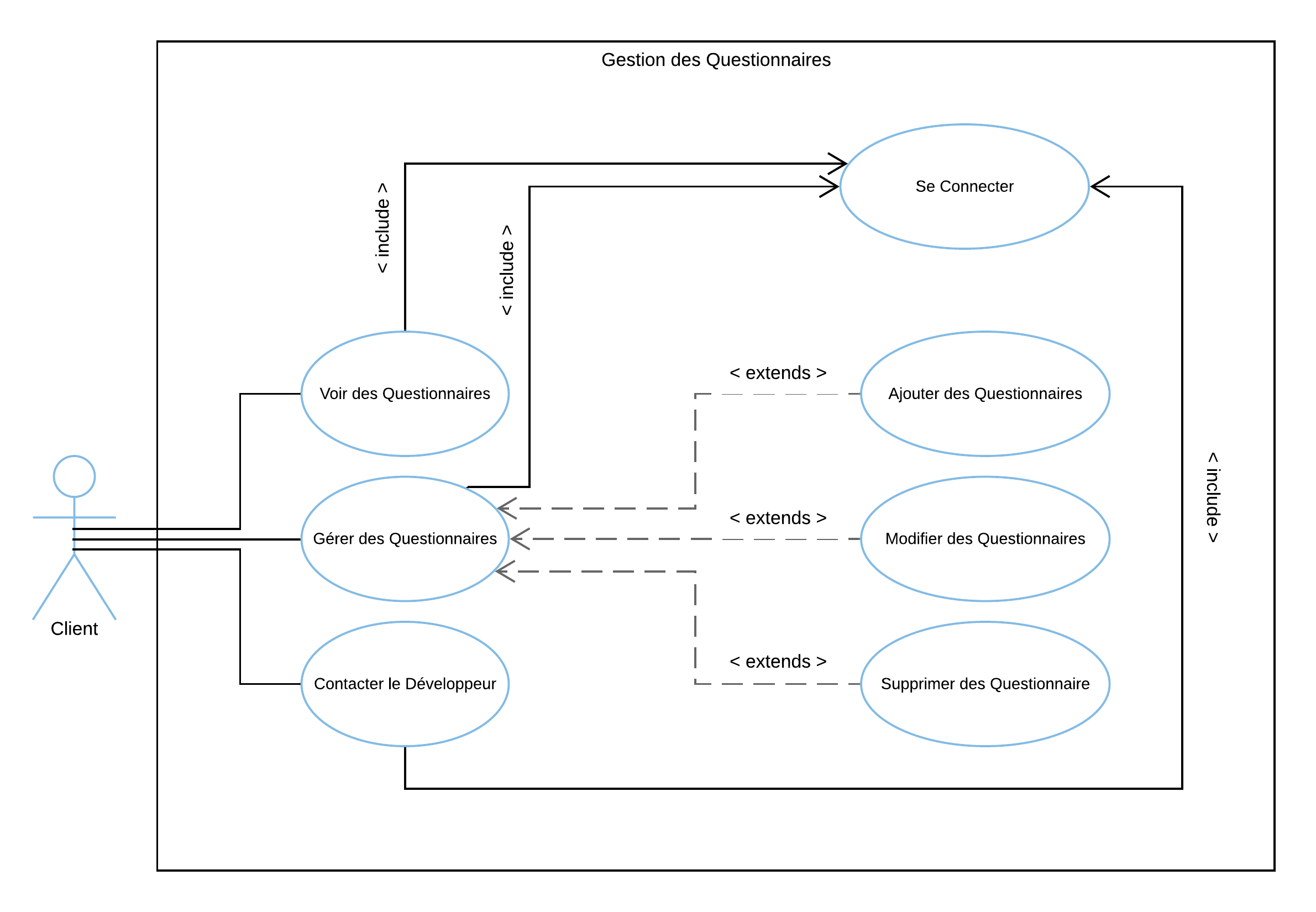
Ce diagramme présente le système entier avec l’application, cette application possède son propre cas d’utilisation et son propre acteur comme présente dans le diagramme ci-dessous.

Figure 3 Diagramme de cas d'utilisation Global

## Diagramme de cas d’utilisation de l’utilisateur

Le client du système peut gérer les questionnaires en ajoutant une nouvelle Questionnaire ou /et en modifiant ou en supprimant une qui existe déjà.

En effet après la vérification du nom d’utilisateur et du mot de passe par le système, l’utilisateur à la possibilité de consulter les questionnaires, de contacter le développeur en cas des problèmes.

Figure 4 Diagramme de cas d'utilisation de l'utilisateur

### Description du cas d’utilisation gestion des Questionnaire

|  |  |
| --- | --- |
| Cas d’utilisation | Gestion des questionnaires |
| Acteur | L’utilisateur |
| Résumé | L’utilisateur peut gérer les questionnaires qui existent dans le système : Il peut ajouter, modifier ou supprimer une Questionnaire. |
| Scénario nominale | * 1. L’administrateur peut afficher et consulter une liste détaillée des questionnaires qui sont déjà créer.   2. L’administrateur saisie les informations de questionnaire.   3. Le système vérifie les informations saisies.   4. Le système enregistre le questionnaire.   5. L’administrateur reçoit un message de succès.   6. L’administrateur choisie un questionnaire.   7. L’administrateur change les informations.   8. Le système enregistre les nouvelles données   9. L’administrateur reçoit un message du succès.   10. L’administrateur choisie un questionnaire.   11. Le système supprime le questionnaire choisi.   12. L’administrateur reçoit un message de succès   13. L’administrateur saisie son nom, prénom, et le problème.   14. Le système envoi les informations au développeur à travers son E-mail.   15. L’administrateur reçoit un message de succès. |
| Scénario d’erreur | L’administrateur saisie des données non valide.  L’administrateur ne saisit rien. |
| Pré-Condition | Utilisateur Connecter. |
| Post-condition | Questionnaire Ajouter.  Questionnaire Modifié.  Questionnaire Supprimer. |

Tableau 3 Description du cas d'utilisation Gestion des Questionnaire

# Diagrammes des séquences

Le diagramme de séquence, décrit les scénarios de chaque cas d’utilisation en mettant l’accent sur la chronologie des opérations en interaction avec les objets. Un diagramme de séquence montre une interface présentée en séquence dans le temps. En particulier, il montre aussi les objets qui participent à l’interaction par leur « ligne de vie » et les messages qu’ils échangent présentés en séquence dans le temps.

Voici quelques notions de base du diagramme :

* Scénario : une liste d’actions qui décrivent une interaction entre un acteur et le système.
* Interaction : un comportement qui comprend un ensemble de messages échangés par un ensemble d’objets dans un certain contexte pour accomplir une certaine tâche.
* Message : un message définit une communication particulière entre des lignes de vie (objets ou acteurs).

## Etude de quelques diagrammes des séquences

### Diagramme de Séquences Ajouter un Questionnaire

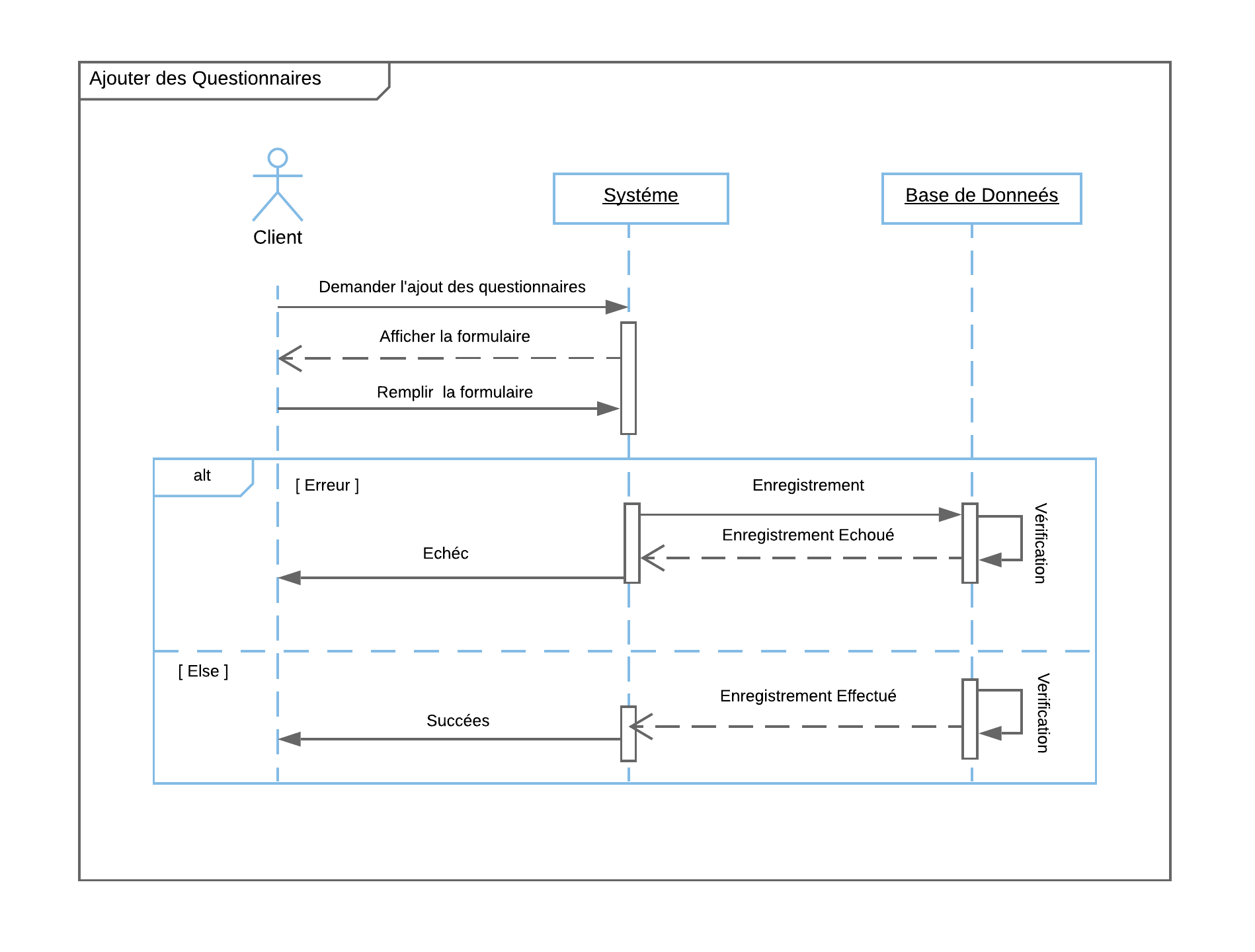


Figure 5 Diagramme de Séquence Ajouter un Questionnaire

Pour ajouter un questionnaire, l’utilisateur de l’application saisie le nom du questionnaire, son type, privée et/ou validé, aussi que les questions et leurs réponses …

Après la vérification des données par le système le questionnaire est enregistré dans la base de données et un message de succès s’affiche à l’utilisateur. Si le questionnaire n’était pas enregistré le système envoie un message d’erreur pour lui informer que le questionnaire n’a pas été enregistré et lui demande de saisir les informations une autre fois.

### Diagramme de Séquences Modifier un questionnaire

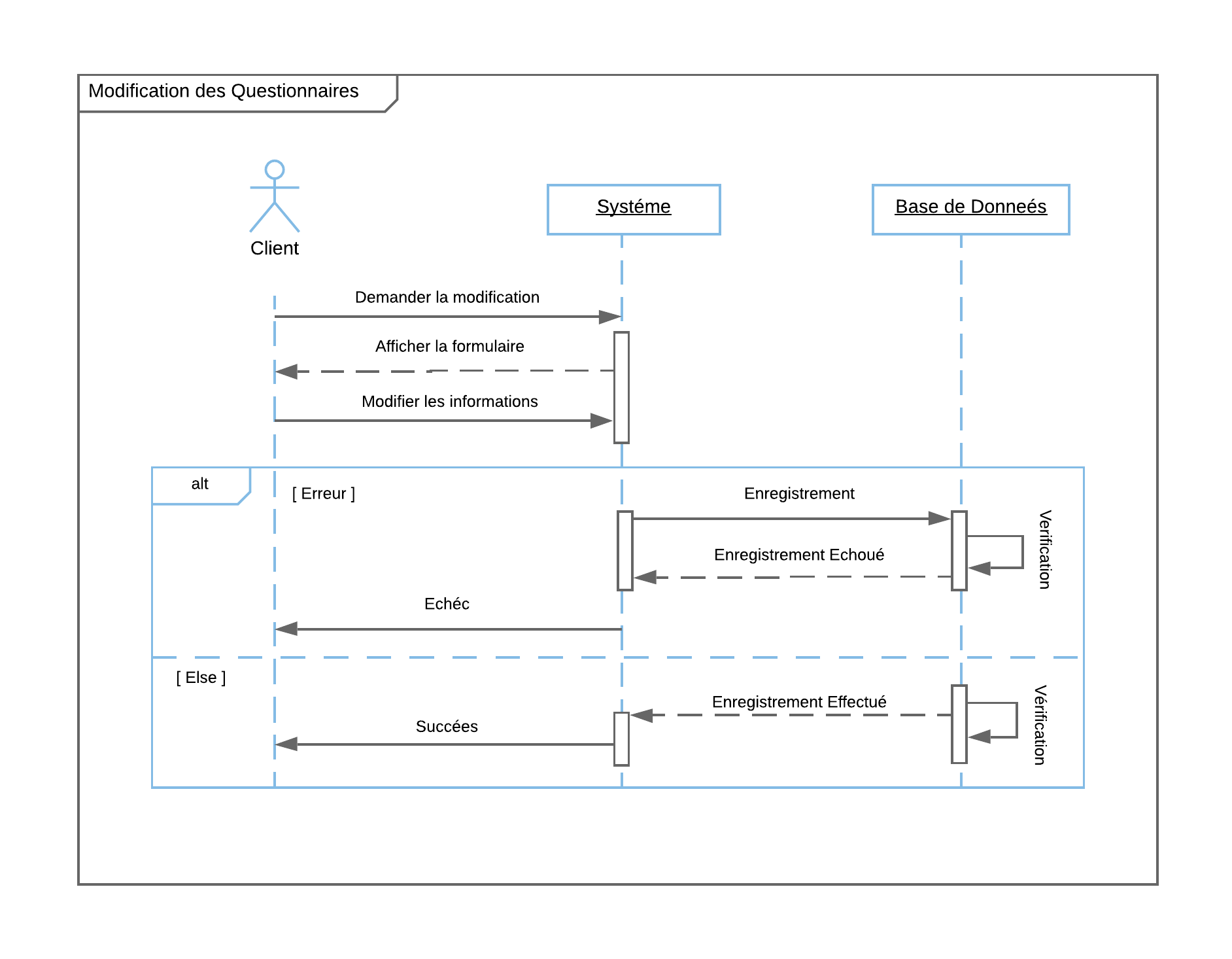


Figure 6 Diagramme de Séquence Modifier un questionnaire

Pour modifier un questionnaire, l’utilisateur de l’application saisie les nouvelles informations au lieu des anciennes dans les champs correspondants de la formulaire …

Après la vérification des données par le système le questionnaire modifier est remplacer et enregistré dans la base de données et un message de succès s’affiche à l’utilisateur. Si le questionnaire n’était pas enregistré le système envoie un message d’erreur pour lui informer que le questionnaire modifier n’a pas été enregistré et lui demande de saisir les informations désiré une autre fois.

# Diagrammes de Classes

Le diagramme de classes est un schéma utilisé en génie logiciel pour présenter les classes et les interfaces des systèmes ainsi que les différentes relations entre celles-ci. Ce diagramme fait une partie de la partie statique d’UML car il fait abstraction des aspects temporels et dynamiques.

Dans notre diagramme de classes on à une seule classe (Questionnaire), l’utilisateur peut ajouter un questionnaire selon un type bien déterminer, choisir sa date d’ouverture et de fermeture, et privé ou validé, aussi l’utilisateur peut choisir un nom de section qui se compose des questions et des réponses sur un produit d’une catégorie bien défini.

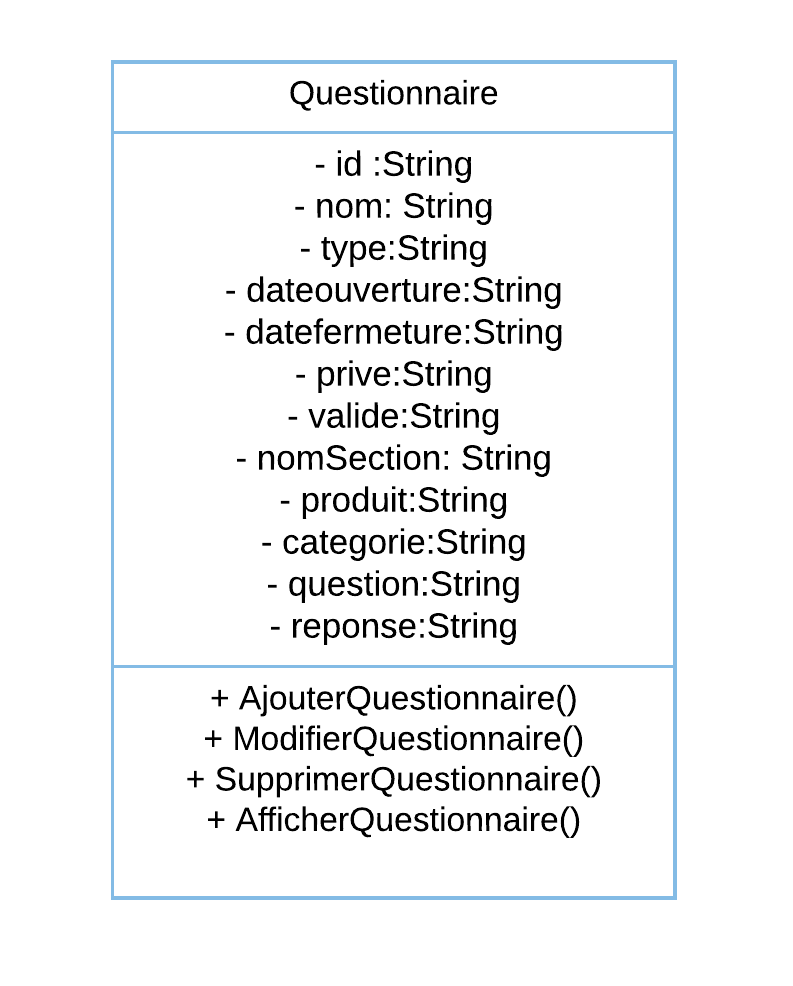


Figure 7 Diagramme de Classe

# La base de Données

## Schéma de la base de données

Nous avons utilisé le système de gestion de base de données NoSQL, orienté document MongoDB pour enregistrer les données de notre système.

Nous avons choisi ce SGBD[[1]](#footnote-1) vu qu’il est très léger et simple à utiliser et configurer, de plus il est open source et tous les hébergeurs web le fournie avec leur pack d’hébergement.

La figure ci-dessous montre les relations entre les différentes entités de notre système.

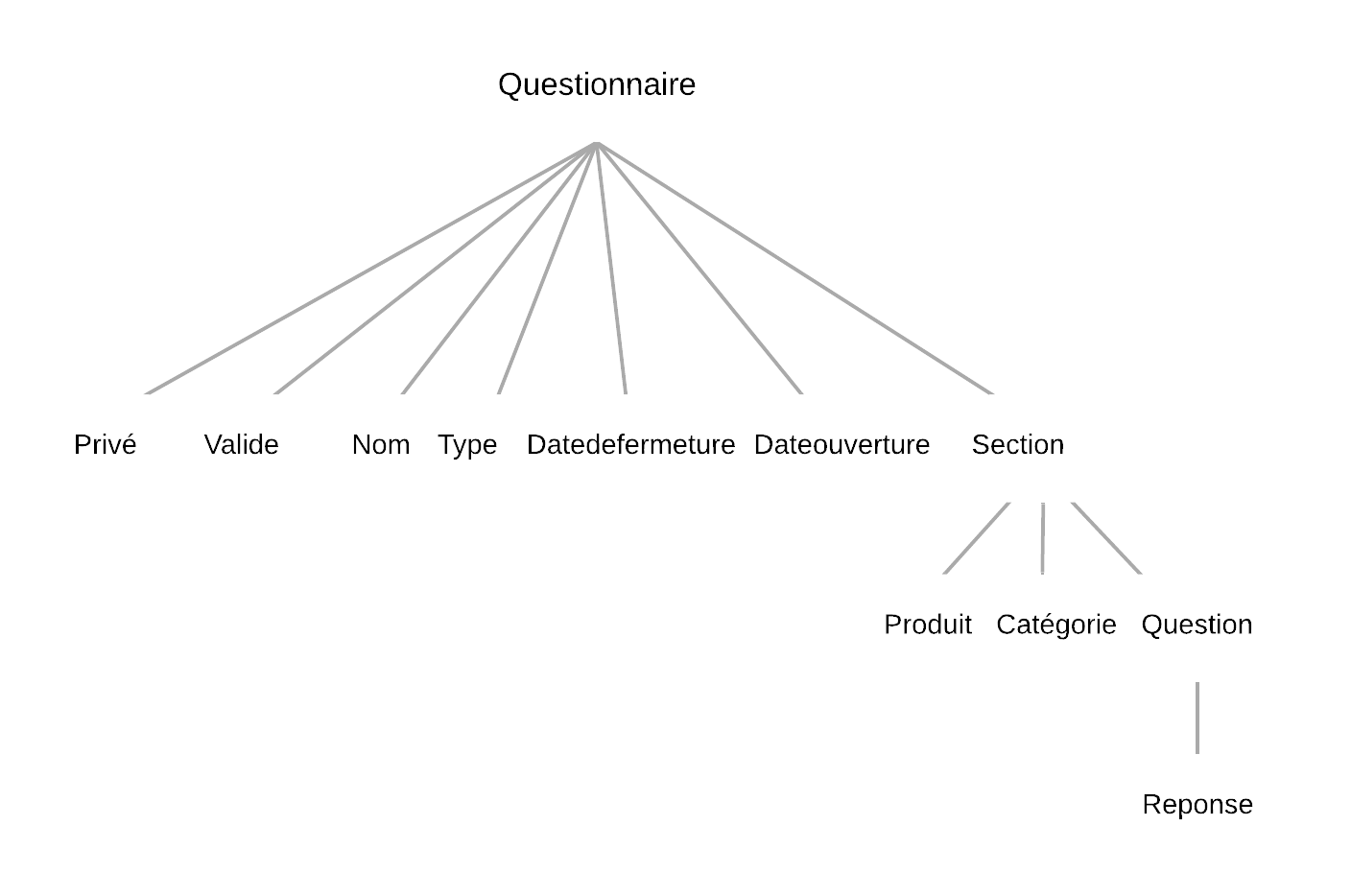


Figure 8 Représentation Arborescente de la base de données

## Conception architecturale

Le pattern MVC[[2]](#footnote-2) est un modèle destiné à répondre aux besoins des applications interactives en séparent les problématiques liées aux différents composants au sein de leur architecture respective.

Ses avantages :

* Séparation des compétences (Design, Base de données, Application).
* Simplicité de mise à jour.
* Vitesse de création de pages.

Ce paradigme regroupe les fonctions nécessaires en trois catégories :

* Un Modèle (Model de données).
* Une Vue (Présentation, Interface utilisateur).
* Un Contrôleur (Logique de contrôle, Gestion des événements, Synchronisation).

Nous expliquons ces trois parties l’une après l’autre :

1. Le modèle

Le modèle représente les structures de données. Typiquement, les classes modèles contiennent des fonctions qui aident à récupérer, à insérer et mettre à jour des informations de la base de données.

1. La vue

La vue correspond à l’interface avec laquelle l’utilisateur interagit. Elle se présente sous la forme d’une Template représentant l’interface.

1. Le contrôleur

Il gère l’interface entre le modèle et le client. Il va interpréter la requête de ce dernier pour lui envoyer la vue correspondante. Il effectue la synchronisation entre le modèle et les vues.



Figure 9 Pattern MVC

# Conclusion

Dans ce chapitre, nous avons pu concevoir un système de gestion des questionnaires en se basant sur les diagrammes du langage UML à savoir le diagramme de cas d’utilisation, le diagramme de séquence et le diagramme de classes.

## **Chapitre 4. La Réalisation**

# Introduction

Ce chapitre représente le dernier volet de ce rapport, il sera consacré à l’implémentation de notre application. Nous commençons par la présentation des ressources matérielles et logicielles utilisées. Nous passons ensuite à présenter des captures d’écran dans le but de mettre en évidence l’aspect ergonomique et fonctionnel des interfaces développées.

1. Architecture du système et environnement de test

## Pourquoi utiliser un Framework Angular

Un Framework est un ensemble d’outils et de composants logiciels organisés conformément à un plan d’architecture et des patterns, l’ensemble formant ou promouvant un « squelette » de programme. Il est souvent fourni sous la forme d’une bibliothèque logicielle, et accompagné du plan de l’architecture cible du Framework.

Les avantages des Framework sont nombreux. En effet, un Framework est portable, de la part de son abstraction de la base de données et de la gestion générique du cache. Les temps de développent avec un Framework sont réellement plus courts. Tous les outils essentiels sont déjà écrits. Le développement des applications sécurisées est facile. Grace aux systèmes d’authentifications, à la gestion des requêtes de la base de données ainsi que la protection CSRF (cross-site-Request-Forgery) qui est gérée par la plupart des Framework.

De plus vu que les Framework sont largement déployés, la chance de trouver les correctifs des problèmes recentrés est plus grande.

Donc les Framework sont la solution la plus apte pour notre application en matière d’extensibilité et l’importance de leurs communautés, ainsi que la nature de projet à réaliser.

### Le Framework Angular

Angular est un Framework JavaScript Open Source développé par Google. Il utilise l’architecture MVM[[3]](#footnote-3), proche du modèle MVC. Cela va permettre de structurer son code et bien séparer la vue (l’interface) des modèles (fonctionnement).

Jusqu’à maintenant, Angular a eu droit à 9 versions majeures. La 1ére version « Angular 1 », Appelé AngularJS et à partir de la 2ème version « Angular 2 », il est appelé simplement Angular.

Il est considéré comme un langage « côté client », ceux-ci permettent de gérer l’interface utilisateur de chaque page (affichage, interactions…) de façon dynamique et viennent en complément aux langages côté serveur.

### Architecture Angular

Le bloc de base d'une application Angular est le [module](https://ionic.mobiletuto.com/v3/GLOSSARY.html#module) (NgModules) qui sert de contexte de compilation et d'exécution à un autre élément nommé Composant. Un composant peut être vu comme la combinaison :

* D'une **Vue** : du contenu HTML
* D'un **Modèle** de données : les informations qui vont être affichées dans le contenu HTML
* D'un **Contrôleur**, qui va se charger de la logique derrière l'affichage des données dans la vue.

Un composant peut être constitué d'autres composants. L'intérêt d'une architecture en composants est que si jamais on souhaite étendre une fonctionnalité particulière, plutôt que de la redéfinir, on va créer un composant qui pourra être appelé partout.



Figure 10 Le Framework Angular

## 1.2 Architecture du système et environnement de test

Pour mettre en place notre système, nous avons utilisé un environnement de développement qui a assuré le bon déroulement de la phase implémentation. Cet environnement comporte des outils matériels ainsi que logiciels.

### Environnement matériel

Nous mentionnons les caractéristiques de nos ordinateurs sur lesquelles nous avons développé l’application parce qu’elles peuvent donner une idée sur les conditions du travail.

Donc l’application a été développé sur deux ordinateurs portables dont les caractéristiques principales sont :

1ère ordinateur

* Nom : Asus
* Processeur : Intel® core™ i5-6200 CPU @ 2.30 GHz 2.40 GHz
* Mémoire installée (RAM) : 8.00 Go (7.86 utilisable)
* Type de système : Système d’exploitation 64 bits, processeur x64

2ème ordinateur

* Nom : Asus
* Processeur : Intel® core™ i5-6200 CPU @ 2.30 GHz 2.40 GHz
* Mémoire installée (RAM) : 8.00 Go (7.86 utilisable)
* Type de système : Système d’exploitation 64 bits, processeur x64

### Environnent logiciel

Nous avons énuméré au cours de cette partie les différents outils utilisés tout au long de ce projet pour l’étude et la mise en place de notre application.

1. Système d’exploitation :

Nous avons utilisé comme système d’exploitation :

* Microsoft Windows 10 professionnel.

1. Outils de développement :
2. Visuel studio code

Visual Studio Code est un éditeur de code source léger mais puissant qui s'exécute sur votre bureau et est disponible pour Windows, MacOs et Linux. Il est livré avec un support intégré pour JavaScript, Type Script et Node.js et possède un riche écosystème d'extensions pour d'autres langages (tels que C ++, C #, Java, Python, PHP, Go) et des runtimes (tels que .NET et Unity).



Figure 12 Visuel Studio Code

1. Node.JS

Node.js est une plateforme logicielle libre en JavaScript orientée vers les applications réseau événementielles hautement concurrentes qui doivent pouvoir monter en charge. Elle utilise la machine virtuelle V8, la librairie libuv pour sa boucle d'évènements, et implémente sous licence MIT les spécifications CommonJS.

Parmi les modules natifs de Node.js, on retrouve http qui permet le développement de serveur HTTP. Il est donc possible de se passer de serveurs web tels que Nginx ou Apache lors du déploiement de sites et d'applications web développés avec Node.js. Concrètement, Node.js est un environnement bas niveau permettant l’exécution de JavaScript côté serveur, elle est utilisée notamment comme plateforme de serveur Web.

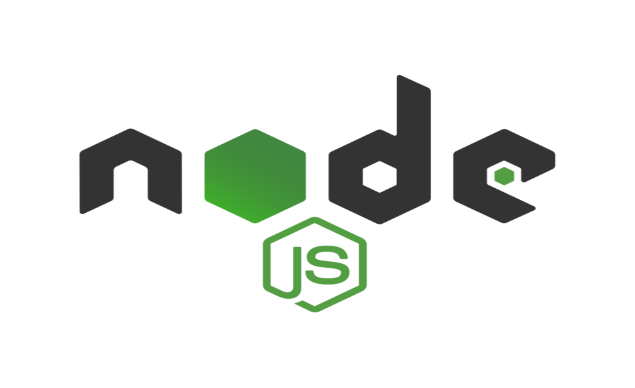


Figure 13 Plateforme Logicielle Node.JS

1. Express.js

Express.js  est un [Framework](https://fr.wikipedia.org/wiki/Framework) écrit en Java Script  pour construire des [applications web](https://fr.wikipedia.org/wiki/Applications_web) basées sur [Node.js](https://fr.wikipedia.org/wiki/Node.js). C'est de fait le Framework standard pour le développement de serveur en [Node.js](https://fr.wikipedia.org/wiki/Node.js). L'auteur original, TJ Holowaychuck, le décrit comme un serveur inspiré de [Sinatra](https://fr.wikipedia.org/wiki/Sinatra_(logiciel))  dans le sens qu'il est relativement minimaliste tout en permettant d'étendre ses fonctionnalités via des plugins.



Figure 11 Le Framework Express.JS

1. MongoDB

[MongoDB](http://www.mongodb.org/) est un SGBD[[4]](#footnote-4), comme [MySQL](http://fr.wikipedia.org/wiki/Mysql) ou PostgreSQL, mais dont le mécanisme est complètement différent. Fini le temps où il fallait créer un schéma de tables relationnelles et créer des requêtes SQL complexes. Grâce à MongoDB vous allez pouvoir stocker vos données un peu comme vous le feriez dans un fichier [JSON](http://fr.wikipedia.org/wiki/JSON)[[5]](#footnote-5). C'est à dire, une sorte de dictionnaire géant composé de clés et de valeurs. Ces données peuvent ensuite être exploitées par du [javascript](http://fr.wikipedia.org/wiki/Javascript), directement intégré dans MongoDB, mais peuvent également être exploitées par d'autre langage comme [python](http://fr.wikipedia.org/wiki/Python_%28langage%29).



Figure 14 Le SGBD MongoDB

1. Postman

Postman est une application permettant avec un navigateur web de lancer des appels d’API et de les tester.

Postman permet d’envoyer des requêtes vers l’API de site en lui ajoutant des en-têtes clés/valeurs puis il permet de formater le résultat sur les plusieurs formats tels que JSON, XML, HTML et autre.



Figure 15 L'application Postman

1. Bootstrap

Bootstrap est un Framework qui facilite et accélère le développement Front-End. Il inclue une base CSS très complète configurée à partir d’un fichier de variables, un ensemble de conventions e structure HTML et de nommage de classes des librairies Javascripts pour les fonctions les plus courantes.



Figure 16 Le Framework Bootstrap

Conclusion Générale

Ce stage a été une expérience supplémentaire dans le métier de l’informatique. Il a été très formateur, il m'a permis d'élargir mon expérience en entreprise. Les nombreuses personnes que j'ai rencontrées m'ont permis de confirmer la vision que je m'étais faite les années précédentes, qui est le travail en équipe. J'ai pu mettre en pratique mes connaissances en informatique mais aussi relationnelles.   
 J'ai aussi découvert l'organisation et le règlement de la boite qui met en avant une qualité de service et d'étude.  
Ce stage de perfectionnement de la 2 ème année est donc une très bonne occasion de confirmer son projet professionnel et de cerner toutes les facettes du métier de l’informatique.

Même si le stage n'est pas en rapport direct avec la formation acquise, je pense que c'est une bonne expérience et un bon complément à cette année.

Références

<https://www.lucidchart.com/pages/fr/exemple/uml-online>

<https://baptiste-wicht.developpez.com/tutoriels/conception/mvc/>

<https://fr.wikipedia.org/wiki/NoSQL>

<http://b3d.bdpedia.fr/docstruct.html#simple-xml-tree>

<https://fr.wikipedia.org/wiki/MongoDB>

<https://www.code4corsica.com/blog/technique/quest-ce-que-angular>

1. Système de Gestion de Base de Données [↑](#footnote-ref-1)
2. Modèle Vue Contrôleur [↑](#footnote-ref-2)
3. Modèle Vue Modèle [↑](#footnote-ref-3)
4. Système de Gestion de Base de Données [↑](#footnote-ref-4)
5. JavaScript Object Notation [↑](#footnote-ref-5)