

Dossier de synthèse   
du projet « **Annu’R** ».  
Présenté pour l’obtention du titre de

**Concepteur Développeur d’Applications**

**Christophe COUCHY**Paris, Doranco, le 30/10/2019

**SOMMAIRE**

|  |  |
| --- | --- |
| **Partie 1 : Introduction au projet** | **P.4** |
| Remerciements | **P.5** |
| Summary | **P.6** |
| Présentation | **P.6** |
| **Partie 2 : Description du projet** | **P.8** |
| Présentation du projet | **P.9** |
| * Introduction | **P.9** |
| * Mais alors, pourquoi réinventer la roue ? | **P.9** |
| Recueil des besoins de l’interface | **P.10** |
| **Partie 3 : Organisation du projet** | **p.11** |
| Le projet en agilité | **P.12** |
| * Le choix de la méthode Agile | **P.12** |
| * Définition du rôle | **P.13** |
| * Personas et actions possibles | **P.13** |
| * Backlogs | **P.14** |
| * User Stories | **P.16** |
| * Estimation des user stories | **P.17** |
| * Technologies retenues | **P.18** |
| **Partie 4 : Conception du projet** | **P.21** |
| Conception | **P.22** |
| * Architecture de l’application | **P.22** |
| * Maquettage – Structure du site | **P.23** |
| * Modélisation via UML | **P.25** |
| * Diagramme de cas d’utilisation | **P.26** |
| * Diagramme d’activité | **P.27** |
| * Diagramme de séquence | **P.29** |
| * Diagramme de classes | **P.30** |
| **Partie 5 : Développement du projet** | **P.31** |
| Le cycle de développement – Back-end | **P.32** |
| * Mise en place du projet | **P.32** |
| * Développement des entités | **P.33** |
| * Liaison à la base de données SQL | **P.34** |
| * Création d’un web service REST | **P.36** |
| Le cycle de développement – Front-end | **P.39** |
| * Présentation de la couche Angular | **P.39** |
| * Mise en place de la couche Angular | **P.40** |
| A suivre ? | **P.43** |

**Partie 1 :**

**INTRODUCTION**

**AU**

**PROJET**

**Remerciements**

Je remercie le centre de formation Doranco pour les moyens humains et l’infrastructure  
mis à disposition pour réaliser cet apprentissage.   
Ainsi que pour mettre à disposition un personnel intervenant pour animer les modules de formations.  
Enfin, merci de m’avoir permis de me former et de passer aujourd’hui cette certification qui me permettra, je le souhaite, de donner un nouveau départ à ma vie professionnelle.

**Résumé**

Il s’agit d’un projet personnel réalisé suite à l’expression d’un besoin dans mon ancienne entreprise. J’ai eu la chance de travailler au siège de cette société, qui est l’un des leaders européens en matériel de bureau, matériel scolaire et fournitures.

J’ai pu constater que du fait du nombre important de salariés, beaucoup ne se connaissaient pas les uns les autres, et il était fréquent de rechercher le nom d’une personne, ou son numéro de téléphone professionnel.  
L’intranet n’ayant pas été entretenu au fil des années, le manque d’un simple trombinoscope se faisait donc sentir.  
A l’époque des réseaux sociaux, il sera intéressant de créer un type de « Facebook local ».

Pour réaliser ce projet, je vais créer une application web JEE, avec :  
  
- une partie Back-End sous technologie Java, avec le framework Spring.  
- une partie Front-End réalisé avec le framework Angular.   
- une base de données en SQL.

La finalité d’un tel projet n’est nullement lucrative. Il s’agit plus simplement de permettre de connecter les gens entre eux.

**Summary**

It’s a personal project realized following the expression of a need in my old company.   
I was fortunate enough to work at the headquarters of this company, which is one of the European leaders in office equipment, school equipment and supplies.

I found that, because of the large number of employees, many did not know each other, and it was common to search for a person's name, or their business phone number.

The intranet has not been maintained over the years, the lack of a simple « trombinoscope » (facebook) was felt.

At the time of social networks, it will be interesting to create a type of "local Facebook".  
To realize this project, I will create a JEE web application, with :

- a Back-End part under Java technology, with the Spring framework.  
- a Front-End part realized with the Angular framework.  
- a database in SQL.

The purpose of such a project is not lucrative. It's more simply about connecting people to each other.

**Présentation**

Je suis Christophe COUCHY, 38 ans, célibataire sans enfants.  
Après une formation mi-artistique (diplômé en infographie print et 3D) mi-informatique (diplômé d’un BTS informatique de gestion, option administrateur réseau), j’ai pu obtenir « le meilleur des deux mondes » en devenant webdesigner (après avoir appris le HTML et le CSS en autodidacte).

J’ai ainsi pu, dans mon ancienne société (une entreprise de e-commerce), réalisé quelques projets servant les intérêts de la société (des refontes graphiques et fonctionnelles du site de vente e-commerce vendu aux clients partenaires).  
C’est ainsi qu’à commencer mon appétence pour le développement informatique. D’abord tourné exclusivement vers le développement FrontEnd (HTML, CSS, JS), la curiosité m’a poussé à découvrir le développement BackEnd.

J’ai donc pu profiter, quelques années plus tard, d’une rupture conventionnelle, pour me consacrer à mon projet de reconversion professionnelle.

Pour réaliser cette démarche, je me suis appuyé sur le site « défi-métier » ou j’y ai trouvé une formation conventionnée dispensée par le centre de formation « Doranco », et pour aller jusqu’au bout de mon projet de formation.

Aujourd’hui, mon objectif est bien sûr de décrocher un emploi en tant que développeur informatique, et, à plus ou moins long terme, évoluer en tant que développeur FullStack.

**Partie 2 :**

**Description du projet**

**Présentation du projet**

**Introduction**

Travaillant dans la société d’eCommerce où j’ai été employé durant 6 ans, j’ai été confronté au problème suivant : il n’y avait pas, dans cette société, de trombinoscope, ni de quelconque annuaire en ligne.  
Pour une société dont le siège emploi plus de 300 personnes, chacune ayant un bureau et un numéro de téléphone correspondant, c’était un manque pourtant évident.

Cette réflexion de ma part s’était fait à l’époque où je venais de terminer mon projet précédent de refonte du site e-commerce, à grand renfort de CSS et de JavaScript, voire de jQuery.  
L’idée de démarrer un nouveau projet était donc très motivante.

Rapidement, des idées vinrent se greffer à cette ébauche de projet : la possibilité de se connecter, de se créer un véritable profil, de discuter avec d’autres….. Très rapidement, ce simple annuaire en ligne devenait une sorte de Facebook d’entreprise….

**Mais alors, pourquoi réinventer la roue ?**

En effet, il serait facile de se demander pourquoi refaire ce qui existe déjà ?  
La première des raisons concerne bien sûr le public visé. Mon objectif n’est pas que chaque employé de la société passe ses journées sur Facebook,   
(je le fais déjà bien assez moi-même ^^)   
mais de cibler avant tout la communication entre les employés de la société uniquement.  
Cela peut inclure des moyens de communications entre les employés, mais davantage pour renforcer l’esprit de grande famille, si cher à beaucoup de sociétés.

Cependant, je me suis heurté très rapidement à un problème de taille : Impossible de réaliser un tel projet avec mes seules connaissances en programmation front end. Une couche Back End s’avérait alors nécessaire (notamment une base de données pour stocker toutes les informations telles que les noms des contacts).

**Recueil des besoins de l’interface**

L’application va se présenter comme un site internet relativement simple, accessible depuis un navigateur internet, avec un menu pour voir accès aux différentes sections du site (A propos, liste des contacts, etc….).

La navigation, possible sur desktop et mobile, sera simple, logique et intuitive.   
Dès la première page, l’utilisateur sera invité à se connecter. Il aura ainsi accès à son espace, et à sa liste de contacts.   
Chaque utilisateur aura en effet, en plus de la liste globale, accès à une liste de contacts « amis ».

L’utilisation se veut principalement depuis une machine de bureau, mais l’accès à un portail mobile sera un plus indéniable

Les besoins orientés métier :

* L’utilisateur doit pouvoir se connecter (l’accès à l’application est réservé à des utilisateurs authentifiés.
* S’il n’a pas déjà de compte, le visiteur (pas encore utilisateur donc) doit pouvoir se créer un compte. Il sera alors redirigé vers un formulaire de création de compte.
* Une fois connecté, l’utilisateur peut consulter la liste des contacts. Un simple coup d’œil à la liste des utilisateurs lui donne accès aux informations les plus basiques d’un annuaire tels que le numéro de téléphone, le nom complet, le numéro de bureau).
* Cliquer sur un contact de la liste lui donne accès à la fiche plus détaillé s’il est amis avec ce contact.
* Si l’utilisateur possède les droits suffisants, il peut ajouter, supprimer ou modifier une fiche contact.

**Partie 3 :**

**Organisation  
du   
projet**

**Le projet en Agilité**

**Le choix de la méthode Agile**

De par mon expérience en milieu professionnel, et notamment en société de prestation de services, j’ai pu me rendre compte à quel point la méthodologie Agile est prédominante.  
Cette [méthode](https://www.soluti.fr/methode-travail-agile-scrum) consiste à développer un projet au fil de courtes étapes successives, les itérations ou sprints. On détermine à l’avance la durée de chaque itération, ainsi que ses objectifs concrets et les tâches à réaliser pour les atteindre.

À la fin de chacune des itérations, **un produit utilisable est livré au client**, qui valide les fonctionnalités répondant à ses attentes. Le même processus est reconduit jusqu’à ce que le produit corresponde en tous points à ses besoins.

Par rapport aux méthodologies en V classiques, cette méthode est plus orientée co-conception. Au fil des sprints, le client donne des feedbacks fréquemment, et a une visibilité sur l’avancement du projet.  
Le projet en est également plus flexible car grâce à ces feedbacks, le client peut préciser sa demande et réorienter le projet.  
On évite de se « perdre » en chemin, et de risquer de livrer un produit final ne correspondant pas aux attentes. Cela évite donc aussi la frustration du client.

Sur le panel de méthode Agiles existant, je vais m’appuyer sur la méthodologie SCRUM pour ce projet. D’une part car c’est celle que j’ai apprise durant ma formation, mais aussi parce que c’est celle que j’ai vu utilisé en entreprise.

La méthode Scrum permet d’améliorer la productivité des équipes en définissant clairement les rôles de chacun au cours du projet.  
On identifie 3 rôles :

* Le *product owner*, généralement le client. Il définit à quels besoins business doit répondre le produit et quelles sont les fonctionnalités attendues.
* Le *scrum master*, ou responsable projet. Il veille à la bonne mise en œuvre de la méthode, du respect des objectifs fixés à chaque *sprint* et vérifie que rien n’entrave l’avancement du projet.
* L’équipe de développement Agile, qui se charge de la réalisation du *sprint* et des objectifs associés.

**Définition du rôle**

Ceci étant dit, le fait d’être seul dans la réalisation de ce projet de développement, me permettra de jouer tour à tour les rôles de PO (Product Owner), Scrum Master et de l’équipe de développement.   
Pour l’occasion, il faudra essayer de cibler les actions, les tâches et responsabilités de chacun des rôles.

La méthode Agile recommande de se fixer des objectifs à court terme. Le projet est donc divisé en plusieurs sous-projets. Une fois l'objectif atteint, on passe au suivant jusqu'à l'accomplissement de l'objectif final. Cette approche est plus flexible. Puisqu'il est impossible de tout prévoir et de tout anticiper, elle laisse la place aux imprévus et aux changements.

**Personas et actions possibles**

La méthode AGILE préconise la création de « personas », ce sont des acteurs virtuels chargés de représenter des personnes ou systèmes afin à décrire des interactions. En tant que Product Owner, je procède à la création de mes personas : seulement trois suffiront à décrire les situations.

* Visiteurs (non connectés)
* Utilisateurs (connecté)
* Administrateurs (possède tous les droits)

|  |  |  |  |
| --- | --- | --- | --- |
|  | Administrateur | Visiteur | Utilisateur |
| Créer un compte |  | OUI |  |
| Se connecter | OUI |  | OUI |
| Se déconnecter | OUI |  | OUI |
| Consulter liste contacts | OUI |  | OUI |
| Ajouter /modifier/ supprimer une fiche contact | OUI |  |  |
| Ajouter des contacts « amis » | OUI |  | OUI |
| Consulter fiche « details contact » | OUI |  | OUI |

**Backlogs**

Le backlog de produit est la liste des fonctionnalités attendues d'un produit. Plus exactement, au-delà de cet aspect fonctionnel, il contient tous les éléments qui vont nécessiter du travail pour l'équipe.  
Ici, nous allons réaliser le backlog en premier afin de facilité l’écrite des User Stories par la suite.

**CONNEXION/CREATION DE COMPTE**

En tant que user, je peux créer un compte.  
En tant que user, je ne peux rien faire avant de m’être connecté.  
En tant que user connecté, je peux me connecter à l’application.  
Tester que la connexion fonctionne et que la sécurité empêche toute intrusion non autorisée.

**DECONNEXION**

En étant connecté, je peux me déconnecter, ce qui me renvoi à la page de login.

**SOCIABILITE**

En tant que user connecté, je peux apercevoir une liste des contacts inscrit (mais pas leur fiche complète).  
En tant que user connecté, j’ai la possibilité de demander une mise en relation avec d’autres personnes.  
Si ma demande est accepté, je peux voir la fiche complète du contact , et également discuter en direct (type Messenger).  
Tester que la visualisation du compte complet n’est visible qu’après validation des deux parties.

**LIEN INTER-ENTREPRISE**

L’application est par défaut liée au réseau local de l’entreprise, et de fait, non accessible en dehors de l’entreprise.  
Cependant, dans le cas où les deux parties le désire, il est possible que deux sociétés distinctes partagent le même réseau commun.  
Il est important pour la sécurité que cette fonction ne soit possible qu’après validation des deux (ou plus) sociétés.

**ATTRACTIVITE**

En tant que user connecté, je peux retrouver la vie d’entreprise via cette application (soit une utilisation similaire aux intranet, en plus moderne).  
L’application fait également office de trombinoscope. Il est possible d’imaginer une liaison avec un produit type skype, et un client mail afin de centraliser toute la communication entreprise autour du même logiciel.

**User Stories**

L’objectif de cette activité est d’ordonnancer les stories afin de les prendre en charge selon une estimation de priorité

**Exemple de premier Sprint**











**Estimation des User Stories**

Cette étape sert à estimer la complexité d’une User Story. De al complexité va dépendre le temps estimé pour la réalisation.

Une User Story peut être d’une grande technicité et avoir besoin d’intégrer beaucoup de mécanismes pointus nécessitant un travail conséquent de documentation. Il faut aussi prendre en compte le niveau de compétences techniques de l’équipe de développement.

Par expérience, je peux dire que l’évaluation des User Stories n’est pas une chose aisée pour un développeur débutant/junior car il est difficile d’évaluer la réelle complexité de celles-ci. Seule l’expérience, la répétition me permettra d’affiner les évaluations vers plus de justesse.

J’ai été habitué à utiliser un jeu de carte basé sur la suite Fibonacci pour estimer mes User Stories :



**Technologies retenues**

|  |  |
| --- | --- |
| **Partie Back-End** | |
| https://www.weblife.fr/wp-content/uploads/2012/05/java-logo-640x393.jpg | J’ai opté pour ce langage car c’est le premier que j’ai découvert lors de mon année à Doranco, mais pas uniquement…. C’est un langage populaire, très bien documenté, et de nombreux forums et d’articles y sont consacrés.  De plus, c’est un langage connu pour être stable, et assez fortement typé. |
| Why Spring Boot is the Preferred Java Framework? | Spring est un framework qui facilite grandement le développement d’applications en Java.  Il repose grandement sur le principe d’inversion de contrôle, et sur l’AOP (Aspect Oriented Programing). Enfin, il est associé à la notion de conteneur léger, ce qui en fait un framework plus rapide que des concurrents comme EJB. |

|  |  |
| --- | --- |
| **Environnement de développement (IDE)** | |
| Description de l'image IntelliJ IDEA Logo.svg. | IntelliJ est un environnement de développement intégré (Integrated Development Environment - IDE) de technologie [Java](https://fr.wikipedia.org/wiki/Java_(technique)) destiné au développement de logiciels informatiques. De par sa simplicité et auto-complétion poussée, c’est l’IDE qui a ma préférence. |

|  |  |
| --- | --- |
| **Partie Front-End** | |
| Résultat de recherche d'images pour "logo angular" | Angular est un framework basé sur le langage open source TypeScript.  En plus d’être très utilisé aujourd’hui dans le monde de l’entreprise, Angular est porté par Google, ce qui lui assure une certaine pérennité.  Son fonctionnement sous forme de component m’a semblé parfaitement adapté à mon apprentissage. |
| Résultat de recherche d'images pour "logo typescript" | TypeScript est un langage de programmation libre et open source développé par Microsoft On parle de « sur-ensemble typé » de Javascript  TypeScript se démarque par rapport à JavaScript avec son typage statique et optionnel, un système de classes et d'interfaces. |
| Résultat de recherche d'images pour "angular CLI logo" | AngularCli est un outil pratique pour fournir une  « Command Line Interface » pour interagir par des lignes de commandes sur diverses opérations de création de vue, de services, etc… |
| Résultat de recherche d'images pour "logo node js" | NodeJS est une plate-forme logicielle libre en JavaScript. Fréquemment utilisé comme plateforme de serveur Web, je l’ai utilisé ici afin d’installer Angular CLI. |
| Résultat de recherche d'images pour "logo bootstrap" | Bootstrap est un framework CSS qui permet notamment de réaliser facilement des design responsive. Je l’utilise principalement pour créer rapidement des designs attractifs. |

|  |  |
| --- | --- |
| **Outils de versionning** | |
| Résultat de recherche d'images pour "logo github" | GitHub est une plateforme open source de gestion de versions et de collaboration destinée aux développeurs de logiciels. Je l’utilise pour sauvegarder mon travail sur un serveur, et également, dans une optique de maitrise de cet outil. En effet, il est très répandu en milieu professionnel. |

|  |  |
| --- | --- |
| **Stockage des données** | |
| Résultat de recherche d'images pour "logo wamp" | WampServer (anciennement WAMP5) est une plateforme de développement Web de type WAMP, permettant de faire fonctionner localement (sans se connecter à un serveur externe) des scripts PHP.  Dans le cas présent, je m’en sers afin d’utiliser MySQL. |
| Résultat de recherche d'images pour "logo mysql" | J’utilise MySQL comme SGBD (Système de Gestion de Base de Données). Il fait partie des SGBD les plus utilisés au monde[3](https://fr.wikipedia.org/wiki/MySQL#cite_note-3), par le grand public et les professionnels, pour sa stabilité et aussi sans doute sa gratuité !! |

**Partie 4 :**

**Conception  
du   
projet**

**Conception**

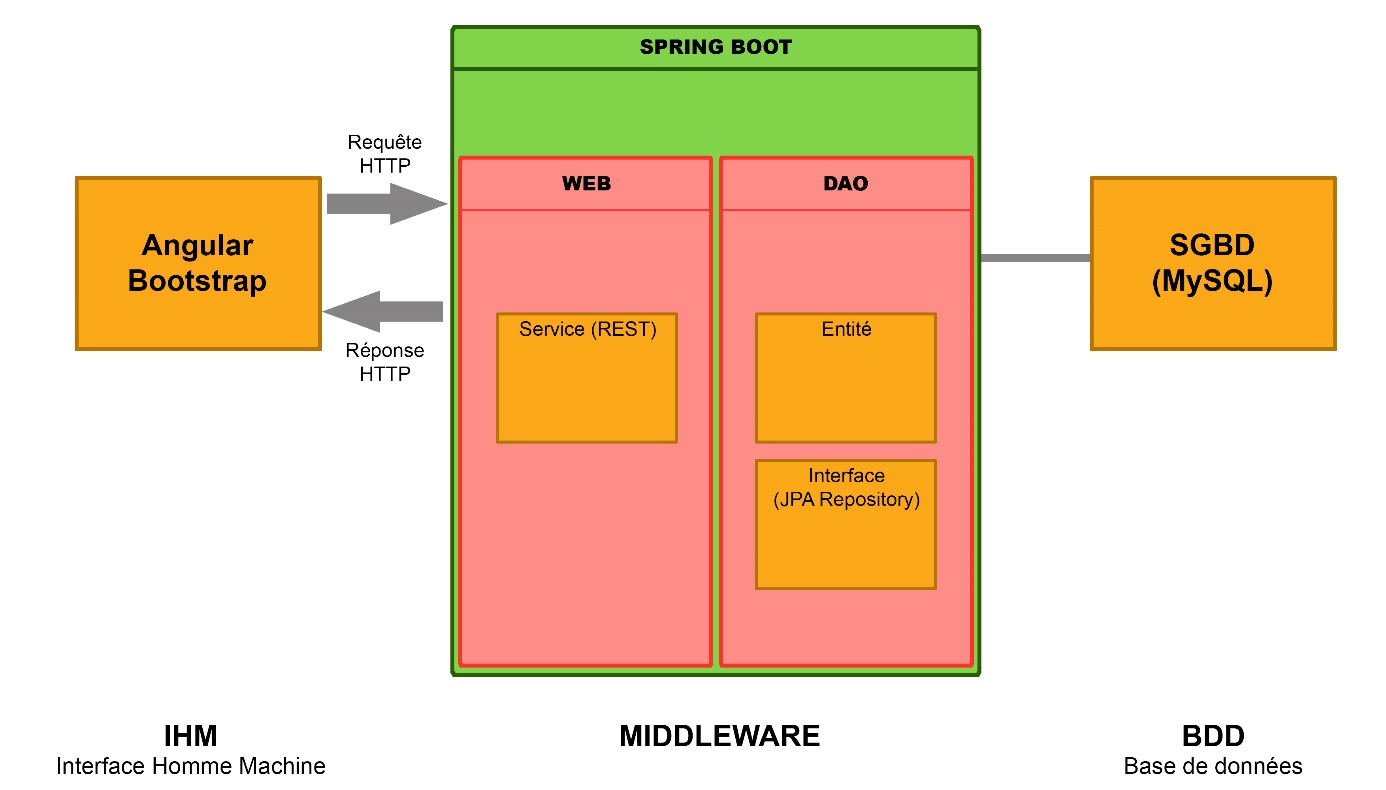
**Architecture de l’application**

Cette application repose sur une architecture de type Service SOA (Servcices Oriented Architecture) et va se composer de trois parties :

* **MiddleWare** : Un BackEnd réalisé en Java, et avec le framework Spring Boot. Ce framework prendra en compte l’inversion de contrôle.  
  Cette partie sera un Web Service Restful.
* **IHM** : Un FrontEnd réalisé en Angular (donc avec du HTML et du TypeScript). On utilisera notamment Bootstrap pour la mise en page et rendre le tout responsive.
* **BDD**: La base de données SQL, ici réalisé avec MySQL.

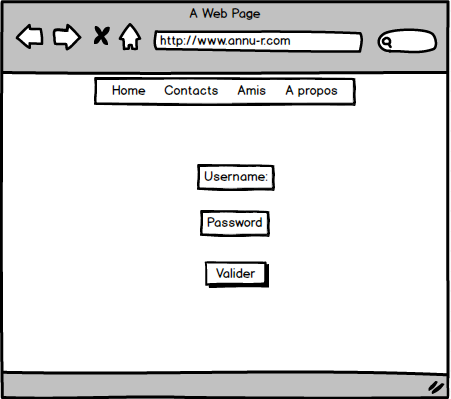
Comme on peut le voir sur le schéma, on va créer une entité qui représentera les contacts affichés dans l’annuaire.  
On utilisera une interface qui utilise JPA Repository (Java Persistance API) afin d’accéder facilement à la base de données.  
La communication entre la partie Front et la partie Back se fera via un micro service Restful. Il s’agira donc de données en JSON qui seront échangés entre les couches de l’application.

Le mécanisme pour accéder aux données est réalisé à l’aide du service DAO (design pattern Data Access Object).

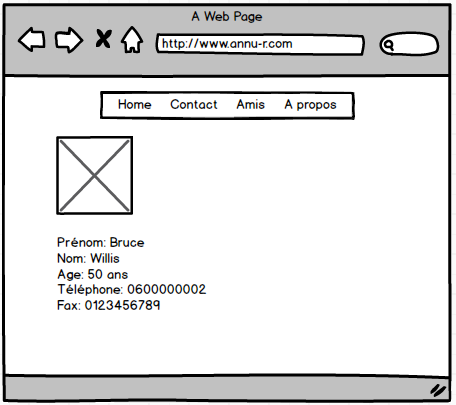


**Maquettage – Structure du site**

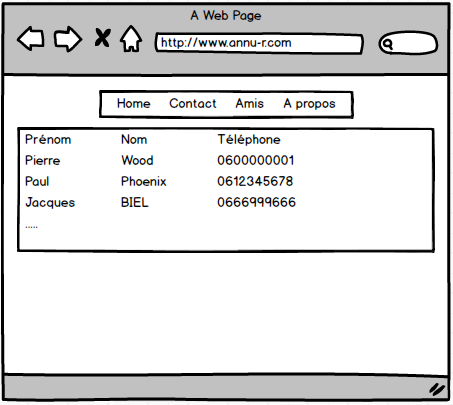
La page de login :



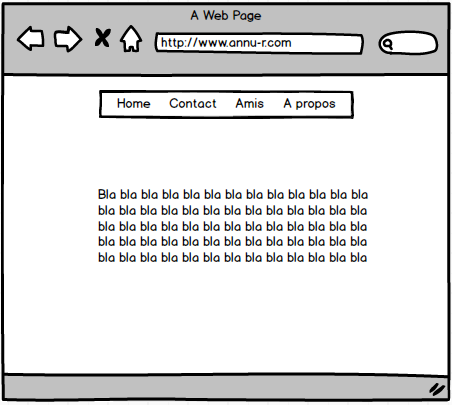
Page Détails de profil :



La liste des contacts (la liste d’amis aura la même forme):



La page « A propos »



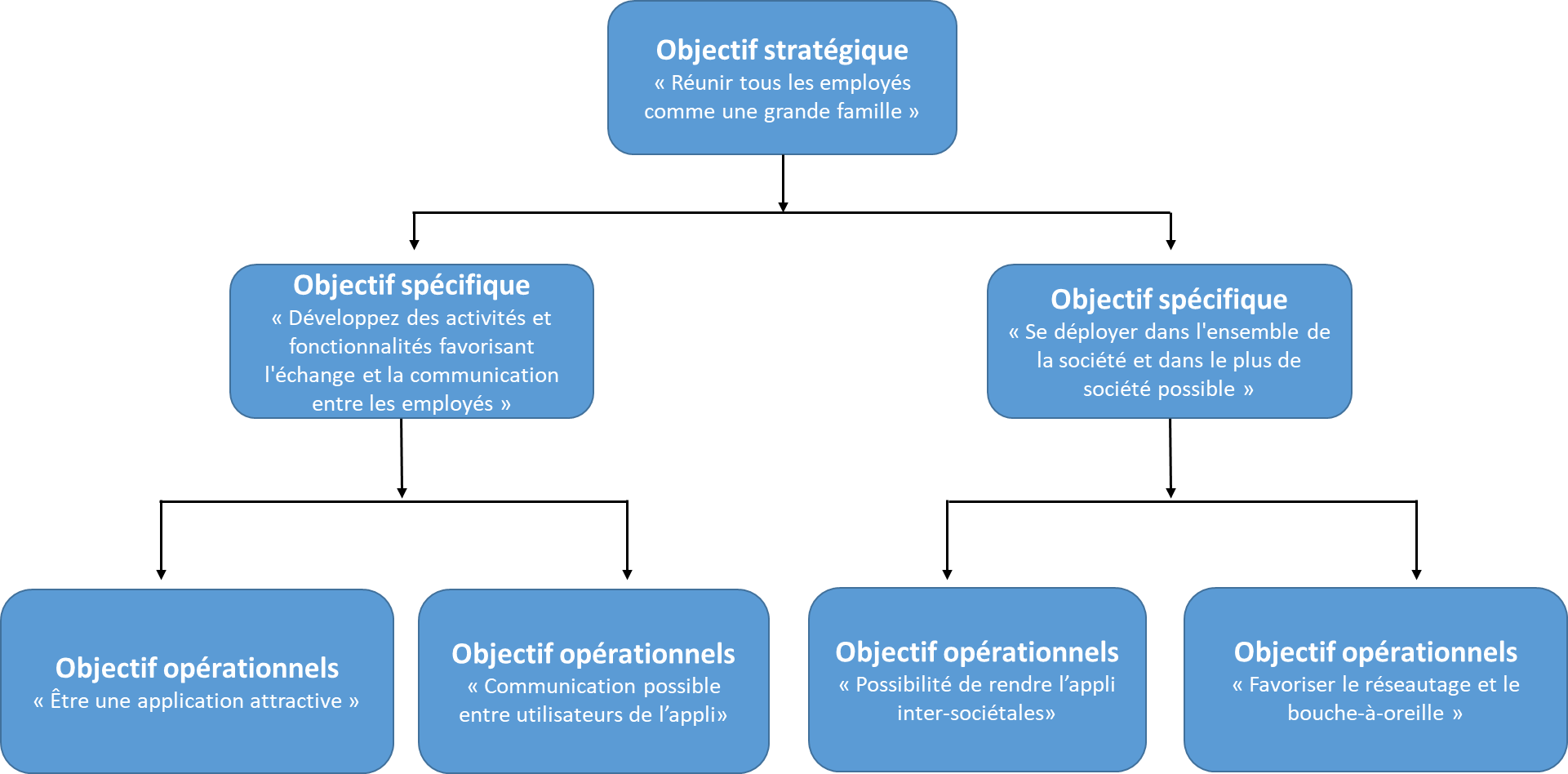
**Modélisation via UML**

Les besoins orientés métier :

* L’utilisateur doit pouvoir se connecter avec un login et un mot de passe
* S’il n’a pas de compte, il peut s’en créer un
* Une fois connecté, il est possible d’ajouter des contacts à la base
* Il est possible de rectifier (modifier) des données de contacts
* Il est possible de supprimer un contact
* (À voir) Il est possible d’ajouter des contacts dans sa liste de contacts pour échanger discussions privées.

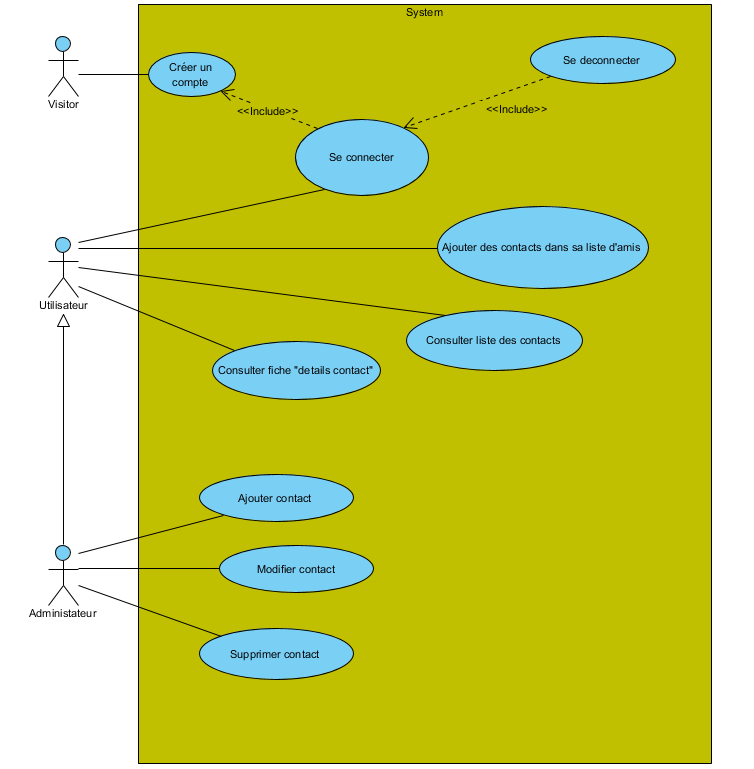
**L’arbre à objectifs**

En premier lieu, je vais utiliser un arbre à objectif afin d’offrir une vue simple des objectifs du projet :



**Diagramme de cas d’utilisation**

Ce type de visuels met en scène l’interaction fonctionnelle des acteurs avec le système.  
Ici, j’ai schématisé les différentes actions possibles par les simples visiteurs (soit, non connectés), les Utilisateurs connectés et les administrateurs.

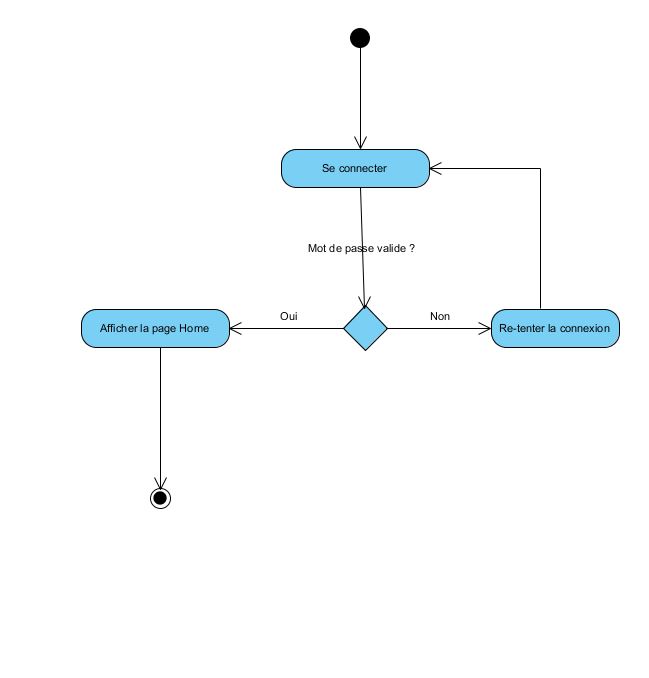


**Diagramme d’activité**

Le diagramme d’activité permet de visualiser le déroulement d’un processus et met à plat une logique d’évènements.

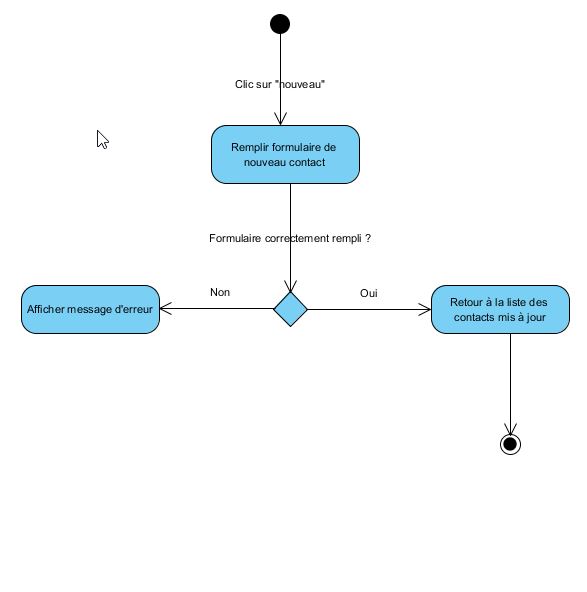
**Diagramme d’activité « authentification »**

Ce diagramme décrit le processus logique imaginé lors de l’authentification dans l’application.

****

**Diagramme d’activité « nouveau contact »**

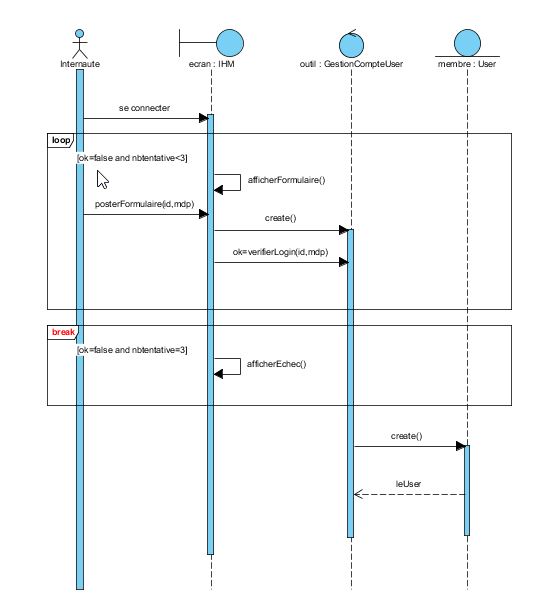
Ce diagramme décrit le processus logique imaginé lors de la création de nouveau contacts dans l’application.

****

**Diagramme de séquence**

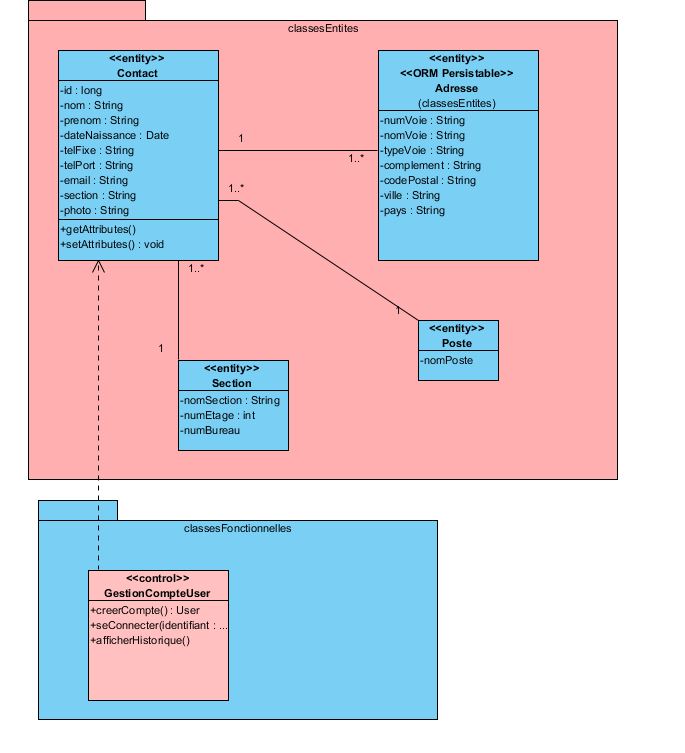
Le diagramme de séquence me permet de réaliser une représentation graphique des interactions entre les acteurs et le système selon un ordre chronologique et de vérifier la logique imaginée pour la séquence d’authentification.

Ici, Ce diagramme de séquence est chargé de montrer les mécanismes mis en place lors de tentative de connexion.

****

**Diagramme de classes**

Le diagramme de classe est une étape fondamentale dans la conception orientée objet, ce sont les visuels permettant une représentation des éléments qui composent un système et leurs relations.

****

**Partie 5 :**

**Le développement**

**du projet**

**Le cycle de développement – Back-End**

**Mise en place du projet**

Je crée mon nouveau projet en choisissant un projet « spring Initializr » dans IntelliJ. De fait, il va utiliser Maven, un outil de construction de projets, qui permet de faciliter et d’automatiser certaines tâches, et notamment, de gérer des dépendances vis-à-vis des bibliothèques nécessaires au projet.

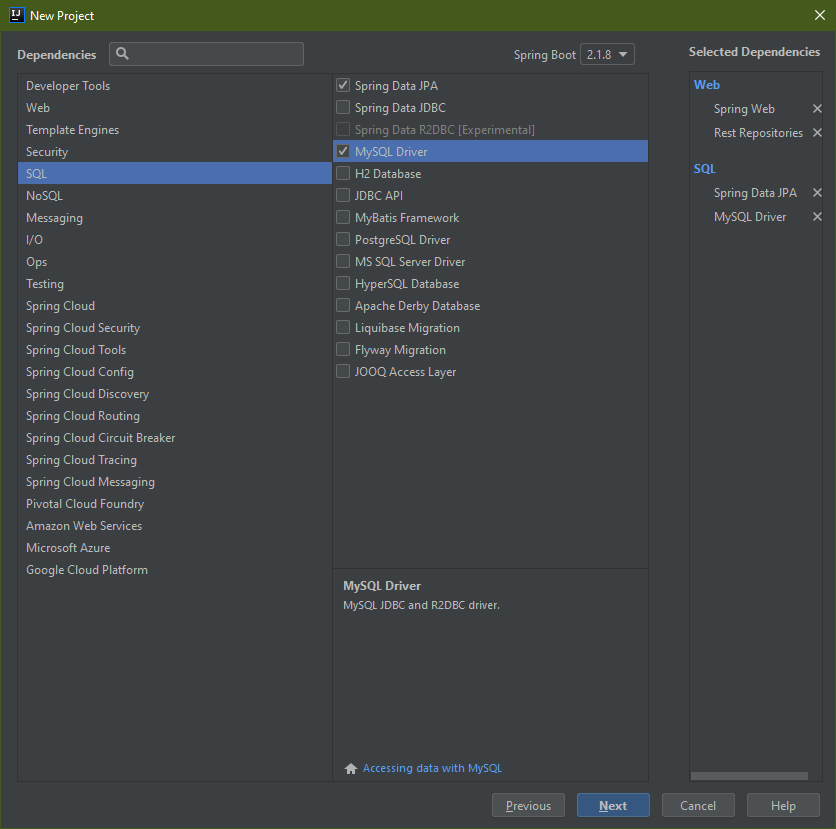
**Dépendances installés :**

- **Spring Data JPA** : On va utiliser une base de données relationnelle donc la Java Persistence API (abrégé en JPA).

- **MySQL Driver**  : On va utiliser cette dépendance pour avoir les pilotes MySQL

- **Spring Web**  : Comme on va créer une application web, on va utiliser la dépendance web (pour api REST)

- **Rest Repository**  : Pour exposer une API RESTfull rapidement (on utilisera l’annotation @RestController).

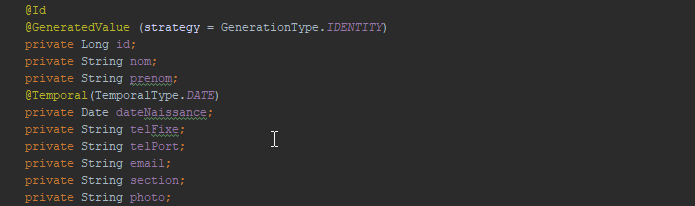


**Développement des entités**

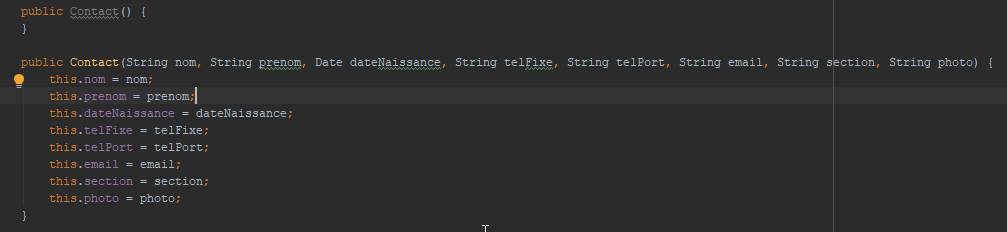
Je vais commencer par créer mes entités. Etant donné que le but de l’application qui sera développé ici, est de faire un annuaire de contact, je vais commencer par créer mon entité « contact ».  
Les entités sont rendus « serialisable », c’est-à-dire que je vais la rendre persistante pour stockage (ici, le but est le stockage en base de données SQL).



Comme c’est une entité JPA (Java Persistance API), on fait le mapping objet relationnel.On utilise pour cela l’annotation @Entity.  
Elle nous informe que cet objet sera transformé en entité.



L’annotation @Id indique le premier attribut suivant (private Long id) est la clé primaire  
@GeneratedValue sert à générer une séquence dans la base de donnée pour une incrémentation automatique.



Une fois les attributs crées, je peux générer les constructeurs, avec et sans paramètres.  
Dans la foulée, je génère également les getters et setters.

Je vais utiliser le principe du « couplage faible » pour ne pas avoir beaucoup de dépendance entre les classes de mon application. L’idée est de rendre la maintenance de cette dernière facile.  
Je vais pour cela créer une interface dao.  
DAO (Data Access Object) est un pattern qui permet de faire le lien entre la couche d'accès aux données et la couche métier d'une application.

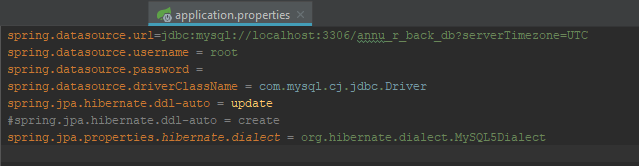
Pour cela, Spring offre l’interface générique JpaRepository qui facilite le travail du mapping objet relationnel.   
C’est parce qu’il s’agit d’une interface générique qu’il faut spécifier les entités :



Ici, l’entité que l’on veut gérer est « contact », et l’id (soit la clé primaire) est ici de type « Long ».

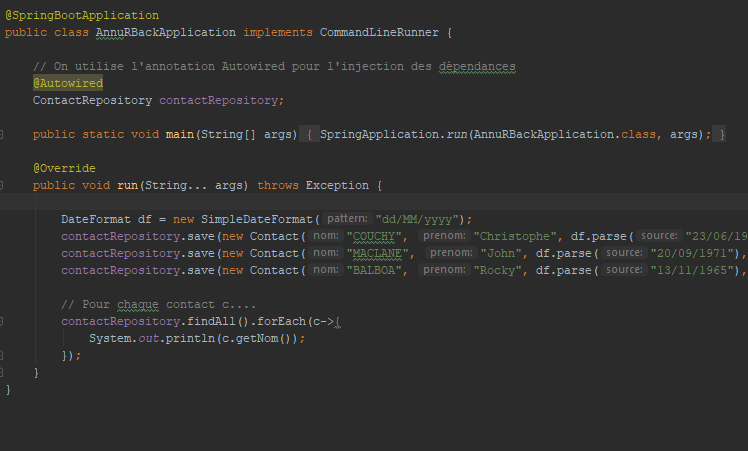
**Liaison à la base de données SQL**

Afin de lier mon application à une base de données, je dois la configurer, et grâce à Spring, cela se fait via le fichier « application.properties ».



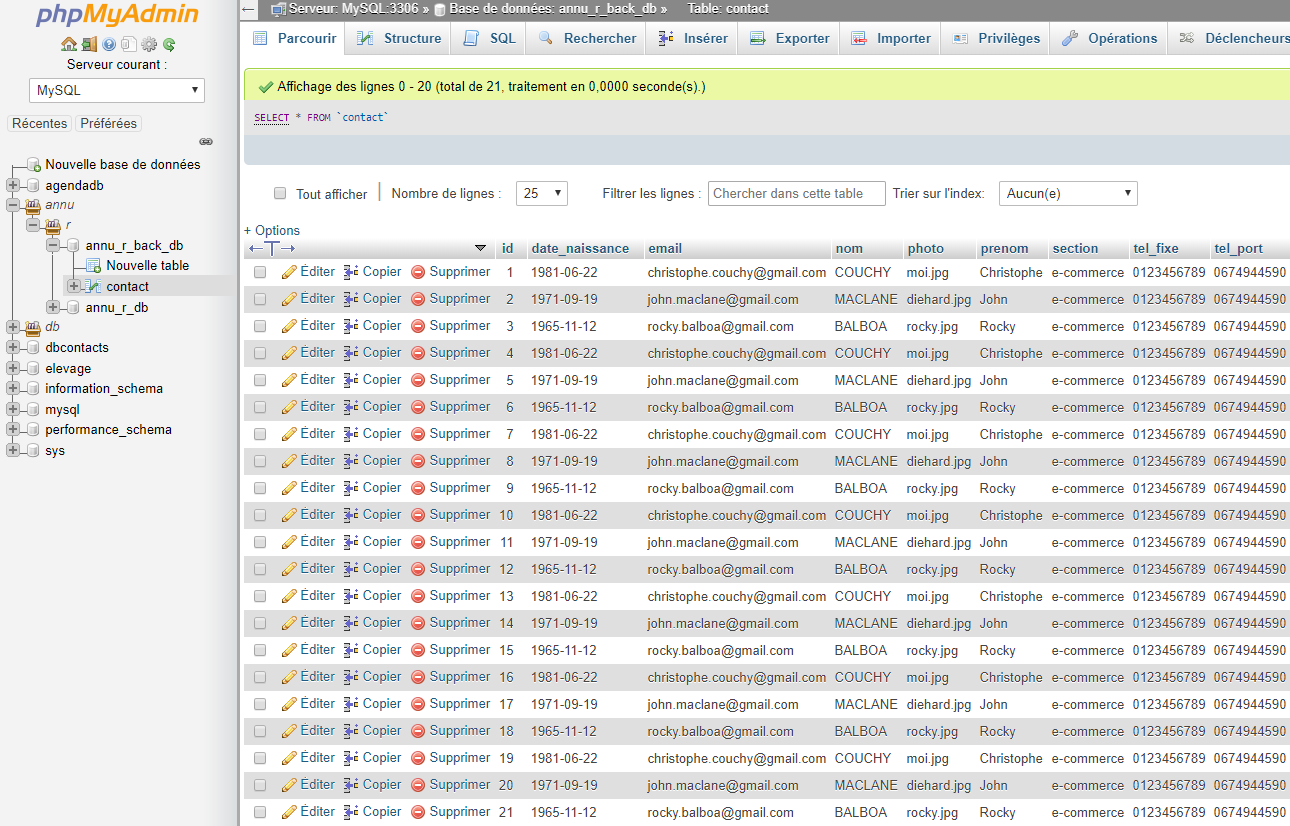
Ici, j’ai créé en amont une base SQL vide, avec MySQL, que j’ai nommé « annu-R-back\_db ».  
Durant la phase de développement, je laisse la valeur « spring.jap.hibernate.ddl-auto » sur update et non sur create.  
La différence est que sur update, à chaque fois que je compile mon code, les quelques contacts que j’ai paramétré s’ajoute en plus à la base de données. C’est-à-dire que si j’en ai injecté 3, j’en aurai 6 en recompilant, puis 9 ainsi de suite.

Pour tester ceci, Il est possible d’implémenter l’interface « CommandLineRunner ». Cela me permet d’injecter des données écrit « en dur » dans la base.



Exemple du fichier « main » de l’application qui implémente l’interface « CommandLineRunner ».  
De fait de cette interface, il faut redéfinir la fonction « run ». On va tester l’interface « contactRepository » pour laquelle on utilise l’annotation @Autowired, pour faire l’injection des dépendances.  
Accessoirement, on utilise DateFormat pour saisir une date et la « parser » au format Date.

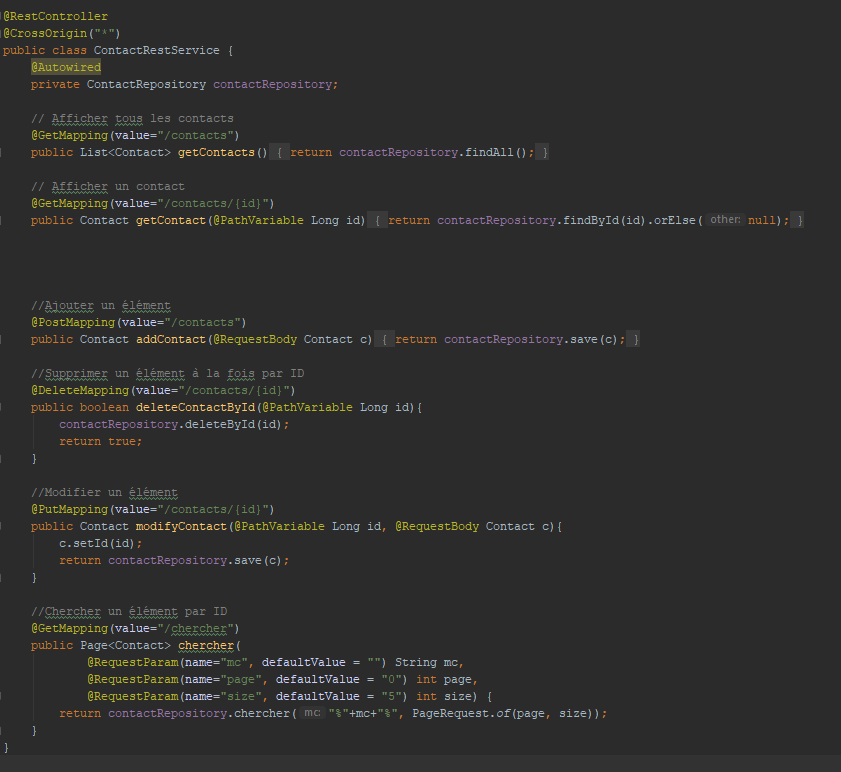
Après cela, il est possible de procéder à un premier test, et le résultat est visible sur la base données SQL :



**Création d’un web service REST**

Un service web est un moyen de mettre en contact deux machines via un réseau.

Un service compatible REST, ou « RESTful », est une interface de programmation d'application qui fait appel à des requêtes HTTP pour obtenir (GET), placer (PUT), publier (POST) et supprimer (DELETE) des données.   
Comme c’est un service REST, on va utiliser l’annotation REST Controller. C’est ici que l’on va écrire les méthodes pour, par exemple, afficher tous les contacts, et d’autres.



A noter que si l’on veut tester le service en question, il est aisé de tester l’affichage des contacts via le navigateur internet :

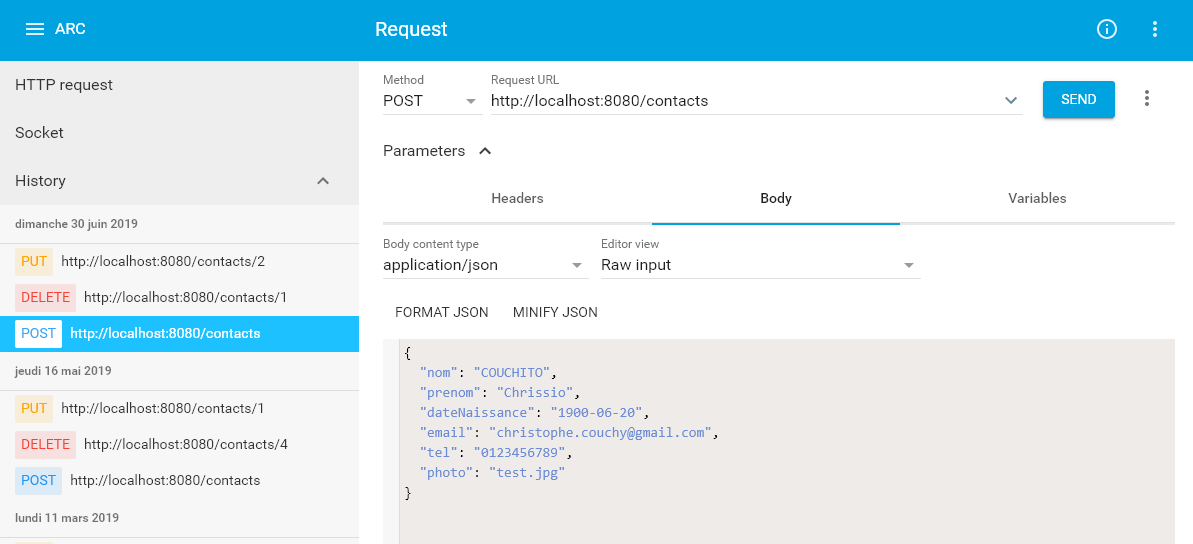


Le WebServices est chargé de répondre au client et le protocole HTTP accompagne la réponse avec des codes d'état pour spécifier une des cinq catégories de réponse (informations, succès, redirection, erreur client et erreur serveur).

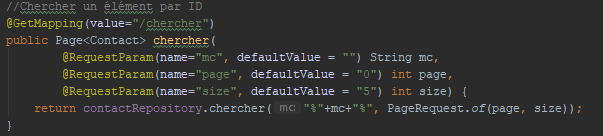
* 200 : succès de la requête ;
* 301 et 302 : redirection, respectivement permanente et temporaire ;
* 401 : utilisateur non authentifié ;
* 403 : accès refusé ;
* 404 : page non trouvée ;
* 500 et 503 : erreur serveur ;
* 504 : le serveur n'a pas répondu.

Cela permet d’informer le client sur le déroulement de la requête lors de la consommation du WebServices.

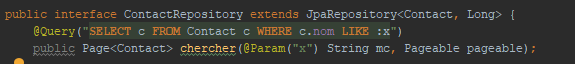
Si l’on veut tester l’ajout, où la modification d’un contact, il sera nécessaire de passer par un client REST. De nombreux clients existent en ce sens. Pour ma part, j’ai utilisé ARC (Advanced REST Client).  
Cet outil est une extension du navigateur Google Chrome.



**La méthode « rechercher »**



Pour cette méthode que l’on utilise dans le ContactRESTService, on va avoir besoin de la déclarer dans notre interface ContactRepository.

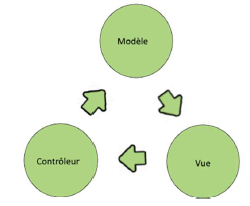


Cette méthode fait des recherches par mot-clé, et retourne des pages, c’est pourquoi on a mot-clé (mc) et « pageable » en paramètres.  
On utilise également une requête SQL, qui va chercher dans la base de données le mot-clé passé en paramètre de la fonction.

**Le cycle de développement - Front-End**

**Présentation de la couche Angular**

Angular répond au design Pattern MVC (Model-Vue-Contrôleur). il s'agit d'un patron de conception qui permet d'avoir une stricte séparation entre la vue (ce que l'utilisateur voit), le modèle (les données) et le contrôleur (les actions possibles).



MVC est utilisé dans de nombreux langages de programmation pour apporter une  
structure / architecture à une application. C'est le nom donné à une manière d’organiser son code et une façon d’appliquer le principe de séparation des responsabilités, en l’occurrence celles du traitement de l’information et de sa mise en forme.

En voici les composants :

**- Modèle :**

Structure de données représentant une entité de l’application, généralement  
transmise en JSON depuis l'IHM vers le middleware et inversement.

**- Vue :**

La vue est rendue par la combinaison des fichiers HTML/SCSS. Les données sont issues  
du Modèle pour mettre la Vue à jour et afficher les bonnes informations dans le fichier

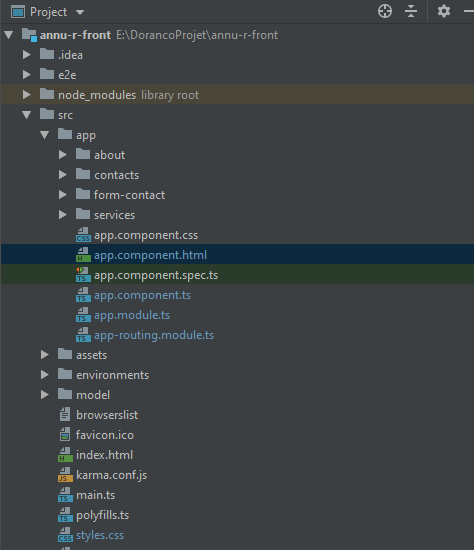
HTML.

**- Contrôleur :**

Comme son nom l’indique, cette couche contrôle des données. Les contrôleurs  
permettent de communiquer avec la Vue. C'est via le contrôleur que je peux mettre  
les données à jour de façon dynamique entre le serveur et le client. Le contrôleur  
s’appuie sur une classe service qui me sert à récupérer ou envoyer des données.

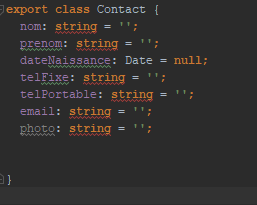
**Mise en place de la couche Angular**

A l’aide d’Angular CLI, il est aisé de mettre en place un environnement de travail Angular.  
Je passe ici sur les commandes techniques de génération de nouveau projet pour indiquer l’arborescence du projet fonctionnel :

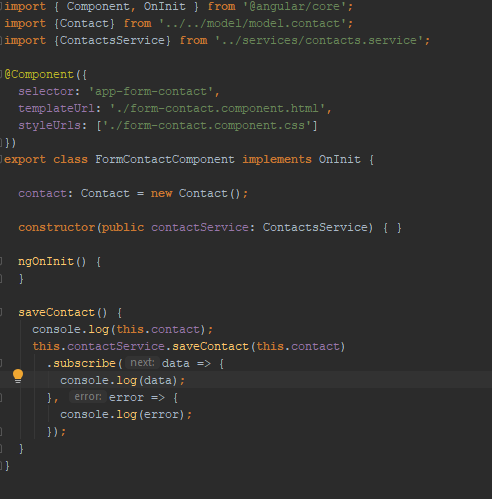


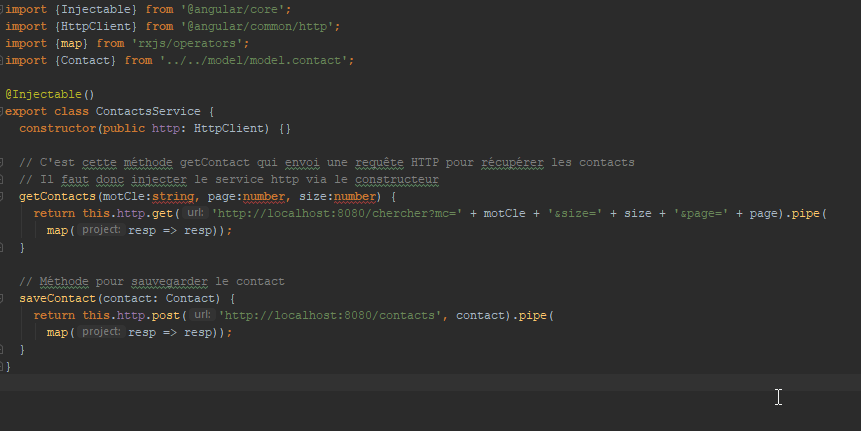
(A noter que ce n’est pas l’arborescence de l’application terminée… A cette date, le code du projet n’est pas encore terminé. Cependant, les premières fonctions basiques sont fonctionnelles et la logique métier est présente).

De la même manière que dans la couche Back en Java, il m’est possible de créer une classe Contact qui sera réutilisé dans le code Angular



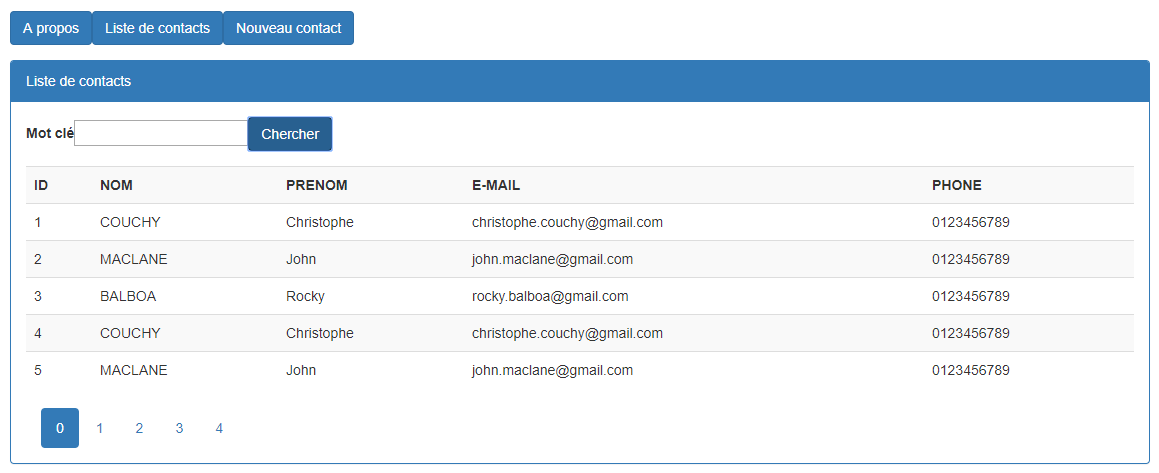
Comme ici, dans le componant correspondant au formulaire de création de nouveau contact.



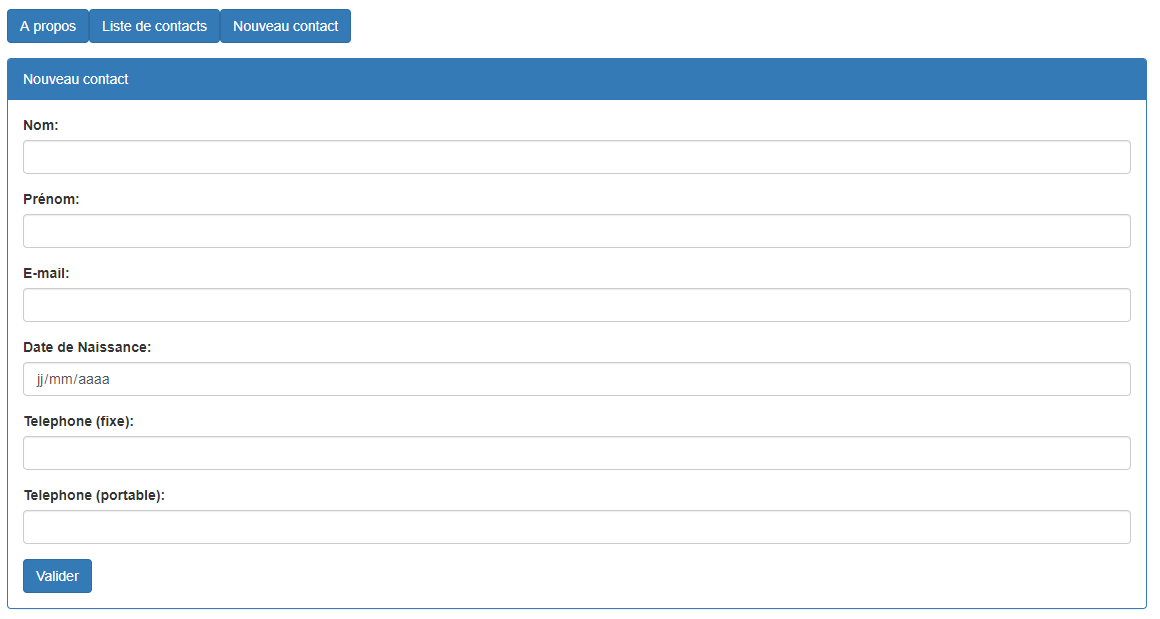


Dans le service crée sous Angular, on récupère les données envoyé en JSON par la couche Java.  
De base, il y a nécessairement des problèmes de Cross Origin, mais Spring permet de régler ces problèmes très facilement grâce à l’annotation @CrossOrigin( ) dans le Web Service REST.

Les contacts crées dans la base de données sont bien récupérés dans la couche Front. La fonction recherche est fonctionelle.



Ainsi que le formulaire de nouveau contact :



**A suivre… ?**

Par expérience, je sais qu’il y a souvent des retards de livraisons en développement… C’est ici le cas.  
Les causes seraient multiples, mais j’ai néanmoins, comme cela aurai été le cas en entreprise, fait en sorte d’avoir un produit montrable dans le délai imparti.

Il s’agit donc actuellement d’une V1, assorti d’un programme fonctionnel, mais pas encore finalisé.  
J’ai personnellement déjà appris beaucoup durant la réalisation de ce projet, et ai bien l’intention de continuer encore à le peaufiner.

A ce jour du 01/10/2019, je vais continuer mon développement dans le but de pouvoir présenter une application finalisé.