

KIRSZ Arthur  
PORCHERON David  
LEVIN Warren



## PROJET D'ARCHITECTURE DES IHM

23 OCTOBRE 2014

*Conception d'une application web de gestion de contacts*

# Table des matières

Introduction	1
Spécifications métier	2
Diagramme de cas d'utilisation	3
Phase de maquettage de l'IHM	4
Diagramme de classe partie métier	6
Schéma Spring MVC	7
Difficultés rencontrées	8
Conclusion	9

# Introduction

L'objectif de ce projet est de concevoir et de développer l'IHM d'une application web de gestion de contacts, à l'aide du framework Spring MVC.

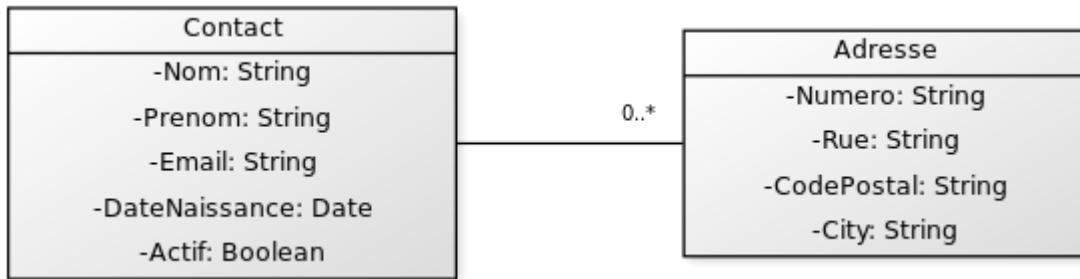
Ainsi, l'outil devra répondre à certaines spécificités, telles que:

1. CRUD sur les contacts :
  - a. Création d'un contact
  - b. Consultation d'un contact et de ses adresses (recherche par critère)
  - c. Modification des propriétés d'un contact
  - d. Suppression d'un contact
2. CRUD sur les adresses :
  - a. Création d'une adresse
  - b. Modification d'une adresse
  - c. Suppression d'une adresse

Pour développer et concevoir cette application, nous avons utilisés les outils suivants :

1. Eclipse Java EE IDE
2. Spring MVC
3. Google App Engine (pour le déploiement du projet sur la plateforme Cloud GAE)
4. GitHub (pour la gestion de source)
5. yUML (pour la conception de diagrammes UML)

## Spécifications métier



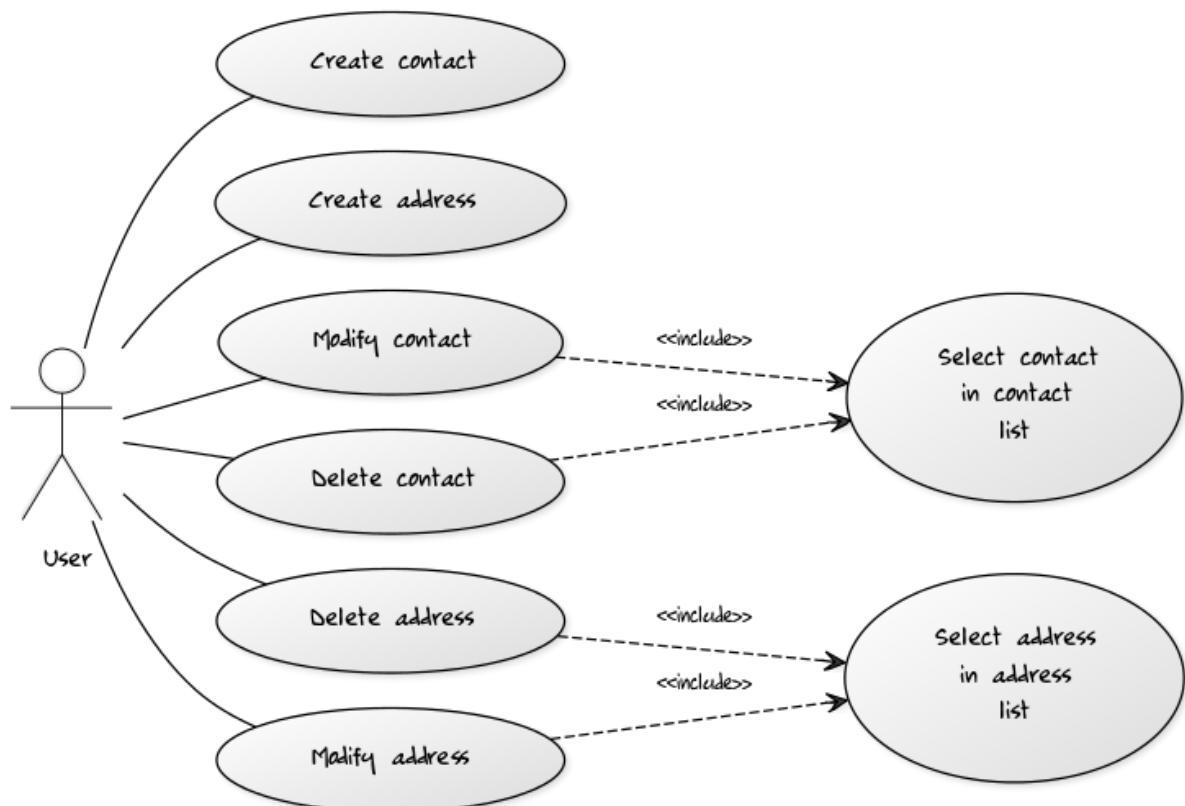
De part ce diagramme on peut affirmer que :

- Un contact peut avoir aucune ou plusieurs adresses
- Une adresse est décorrélée d'un contact (si on supprime un contact, ses adresses sont toujours visibles)

Point de détail en plus :

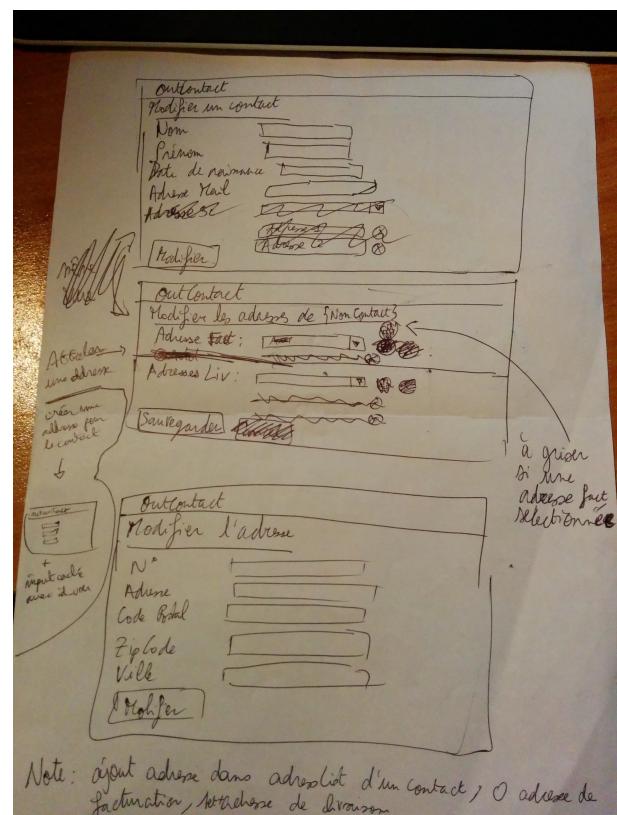
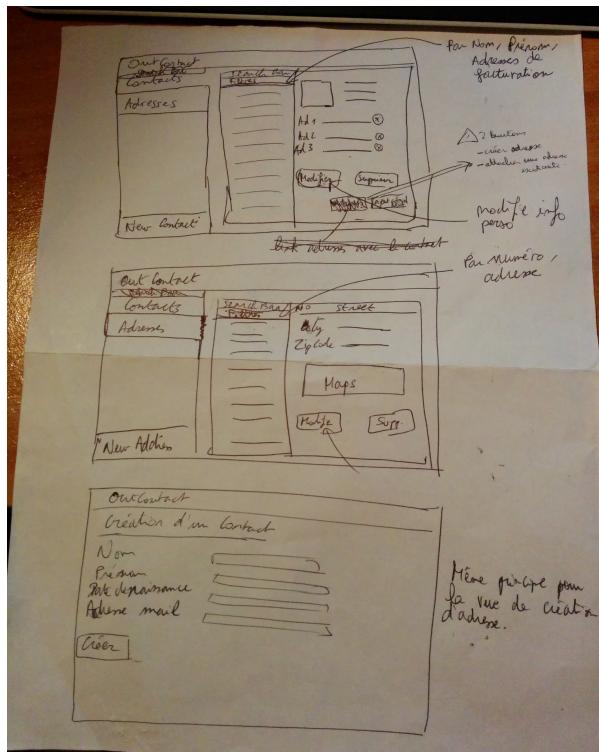
- Un contact peut avoir plusieurs adresses (une adresse de facturation et plusieurs adresses de livraison)

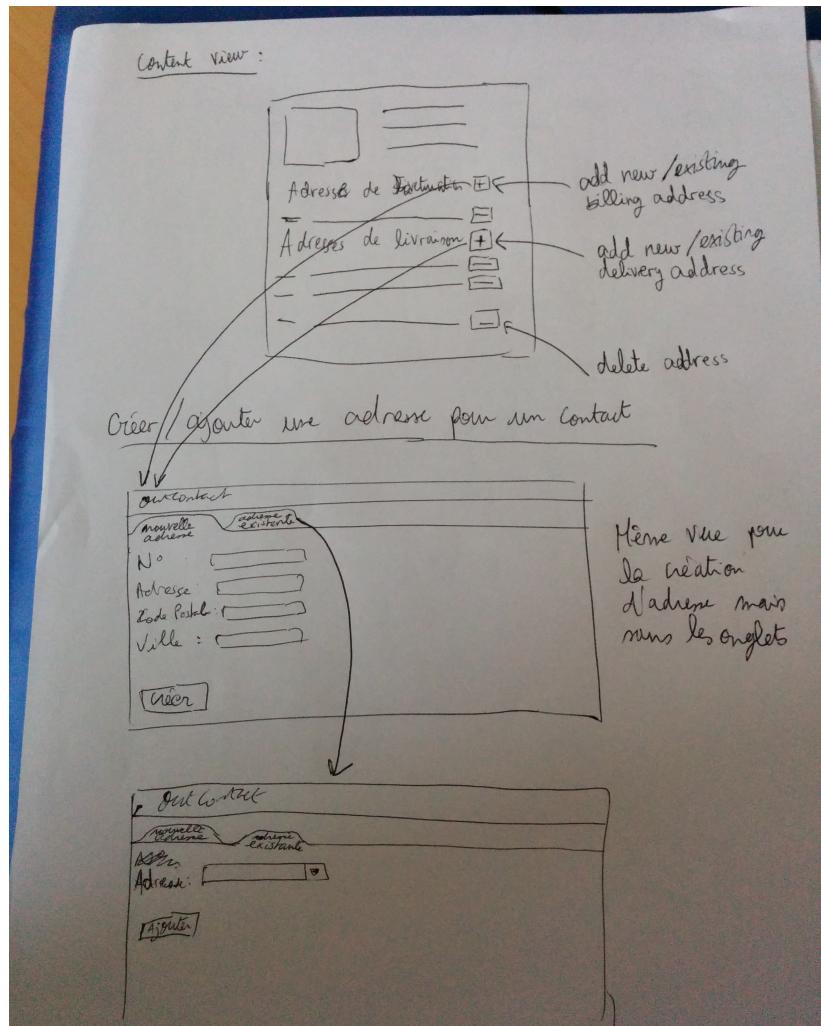
## Diagramme de cas d'utilisation



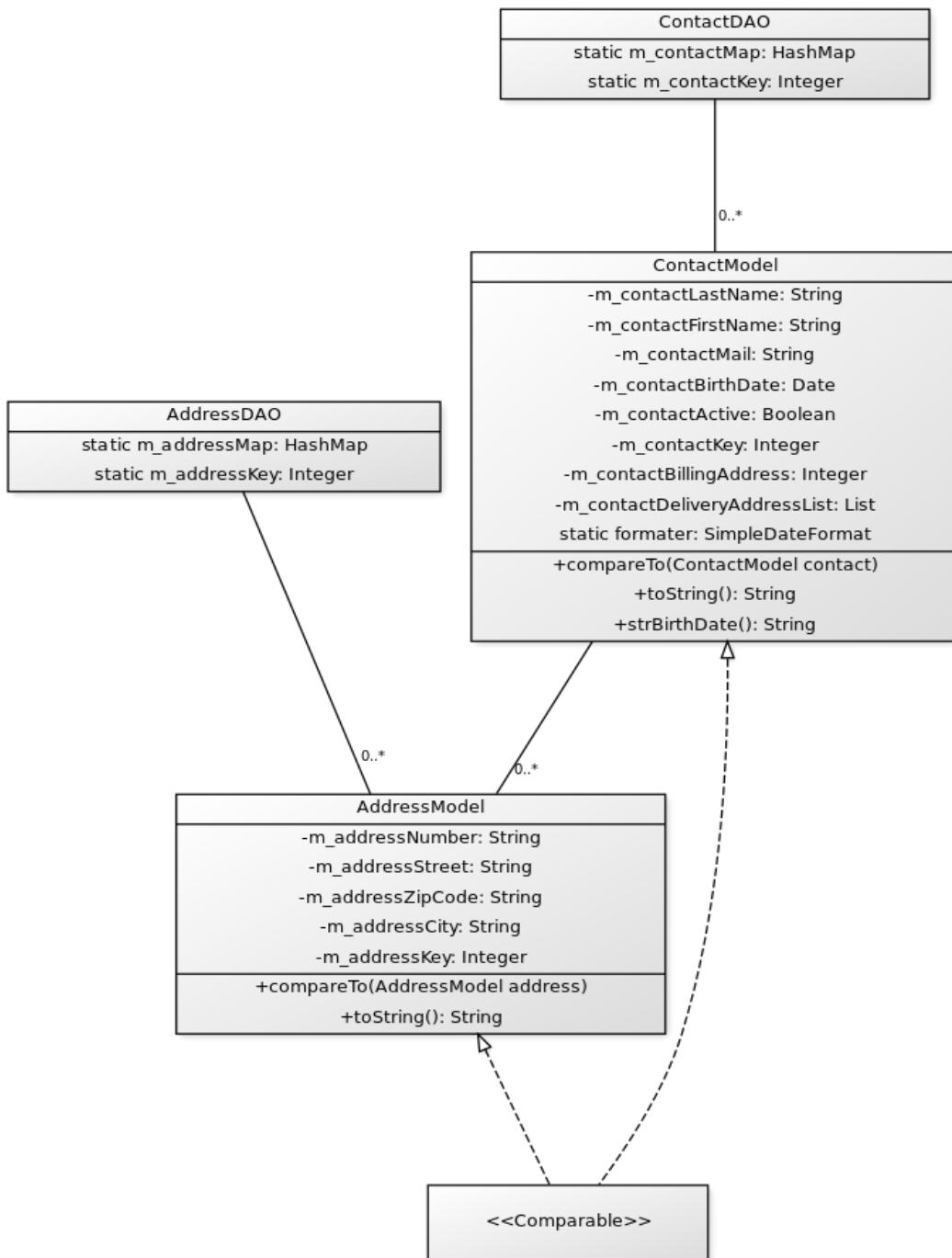
## Phase de maquettage de l'IHM

Cette phase de conception est très importante car elle a permis de nous clarifier les idées sur l'IHM que nous voulions concevoir. Même si ces schémas ne représentent pas la réalité de l'application actuelle, ce sont des étapes importantes de la réalisation de l'application.

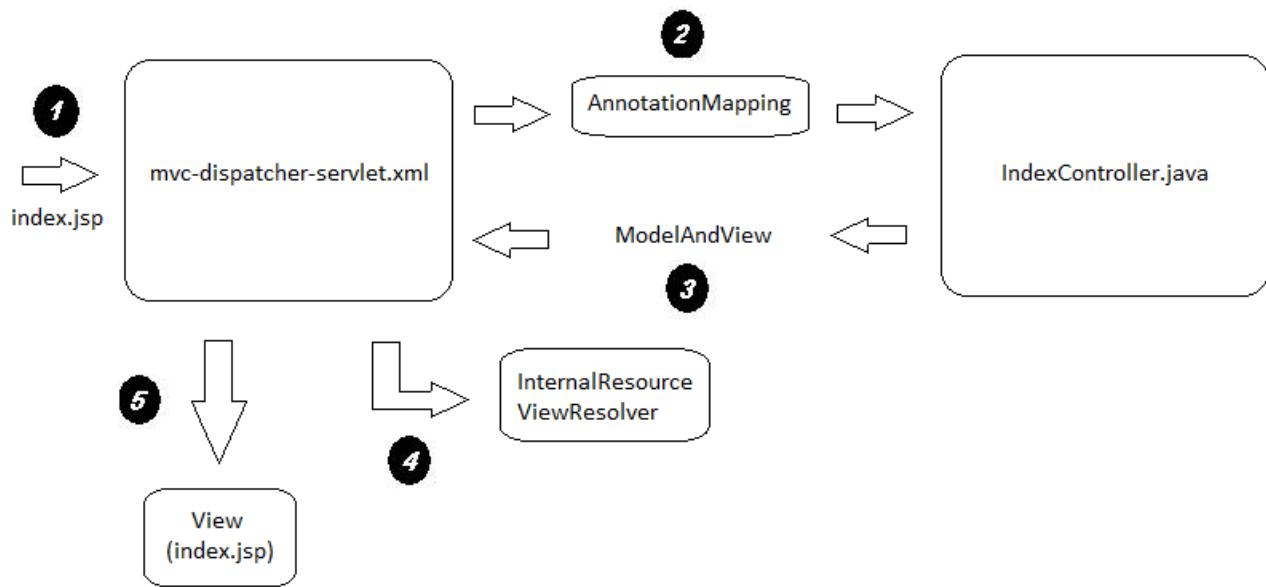




## Diagramme de classe partie métier



## Schéma Spring MVC



Ce schéma décrit l'exécution de l'application, lorsque l'utilisateur sélectionne un contact dans la liste :

1. La page index.jsp va consulter le fichier mvc-dispatcher-servlet.xml pour connaitre l'emplacement des contrôleurs à appeler.
2. Grâce aux annotations Spring (@Controller et @RequestMapping), le contrôleur IndexController est appelé.
3. La méthode initIndexForm du contrôleur IndexController est exécutée et retourne un objet de type ModelAndView contenant le string « viewName » (qui correspond au nom de la page jsp mappée, ici « index ») et le model contenant l'ensemble des éléments à afficher sur la page en question.
4. Le fichier mvc-dispatcher-servlet.xml consulte son « View Resolver » lui permettant de trouver la vue dont le nom logique est « index ». Ici le type de View Resolver choisi est InternalResourceViewResolver.
5. Enfin, le fichier mvc-dispatcher-servlet.xml transmet la requête à la vue associée, ici la page jsp contenue dans /war/view/index.jsp

## Difficultés rencontrées

La première difficulté fut de mettre en place un environnement de développement « propre » et cohérent pour chaque membres du projet. Ainsi, il a été nécessaire de passer par une phase d'initialisation d'environnement pour chacun des membres (avec plus ou moins de difficultés pour certains).

Puis vint la phase la plus difficile : créer un projet Google Maven. Dans un premier temps, nous avions l'idée d'utiliser Maven pour gérer plus facilement les dépendances Spring/JSTL, mais au vue des nombreux essais non fructueux et frustrants de création de projet nous avons décidé de créer un projet Google et de lui rajouter les .jar nécessaire à la bonne exécution de Spring MVC. Enfin nous pouvions coder notre première jsp et notre premier contrôleur.

Également, une difficulté fut de prendre en main Spring et de concevoir l'IHM que nous avions prévu lors de la phase de maquettage du projet. Les différences majeures entre chaque version de Spring et la clarté de la documentation ne nous ont pas favorisé l'apprentissage de ce framework.

Enfin, un travail a été fait sur l'ergonomie de l'application suite au module d'ergonomie de la majeure IHM. Une réflexion a été faite sur l'architecture de l'information, la création d'un système qui soit adapté aux tâches que recherche l'utilisateur. L'apprentissage de la librairie Bootstrap nous a permis de découvrir un certain nombre de techniques pour faire un site en responsive design, et sur l'utilisation d'un système de grille CSS pour créer une navigation unifiée sur les différents terminaux.

## Conclusion

Ce projet nous aura permis de prendre en main plusieurs outils de développements largement utilisés dans le monde du travail comme Eclipse, Spring ou Maven (même si nous ne l'avons pas intégrer au final). Le déploiement du projet sur GitHub nous a permis de mieux gérer la mise en commun des sources afin d'avoir à tout moment la dernière version de l'application. De plus, le fait d'avoir plusieurs projets dans GitHub nous permet d'avoir un portfolio intéressant de projets, ce qui représente une vraie plus-value lors des entretiens.

Enfin, nous avons pu nous challenger lors de la phase de développement en nous attribuant certaines tâches et en les définissant par périodes et par priorités, pour mieux piloter notre projet.

Quelques évolutions sont envisageables pour améliorer le projet :

- Offrir les paramètres passés dans l'url
- Ajouter d'autres champs à la table contact (contactUrlPicture, contactSex, ...)
- Faire évoluer l'application en mode SPA