**Contexte**

L’application est une solution web qui permet à l’utilisateur de rechercher des pros ou des artisans à travers un besoin.

Elle permet à l’utilisateur de pouvoir faire une demande de devis afin de les comparer.

L’utilisateur pourra donc chercher tout type de service comme des maçons, des électriciens des infirmières des cours particuliers. Lors de la saisie au prés de l’utilisateur diverse champs seront disponible afin de lui présenter les meilleurs offres.

**Schéma de compréhension**

Utilisateur

Pro

Plateforme

24/48 heur

Envoie Besoin

Envoie devis

Plateforme

Plateforme

Pro

Plusieurs Devis

Utilisateur

**Diagramme de cas d’utilisation**

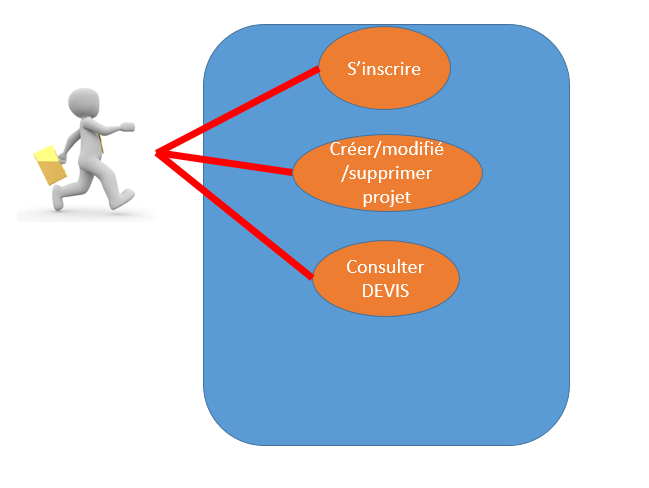
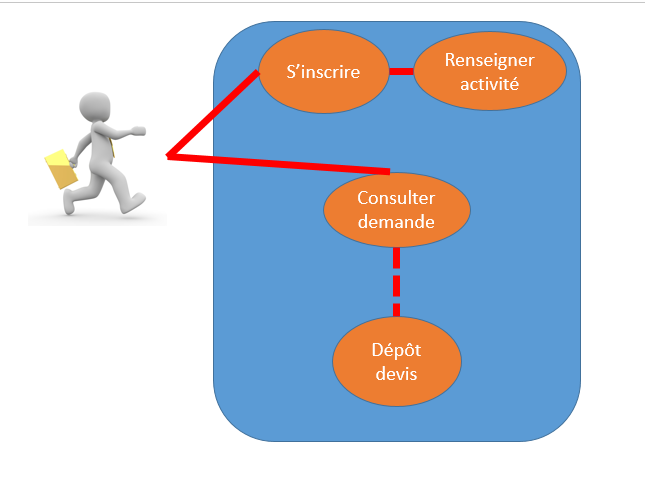


Figure 2 Diagramme cas d'utilisation

Dans ce document nous analysons les cas d’utilisation du profile Customer et le Provider avec les principales actions que chaque utilisateur effectue lors de son utilisation.

**Diagramme de paquetage.**

**Diagramme de classes.**

**Diagramme de séquence.**

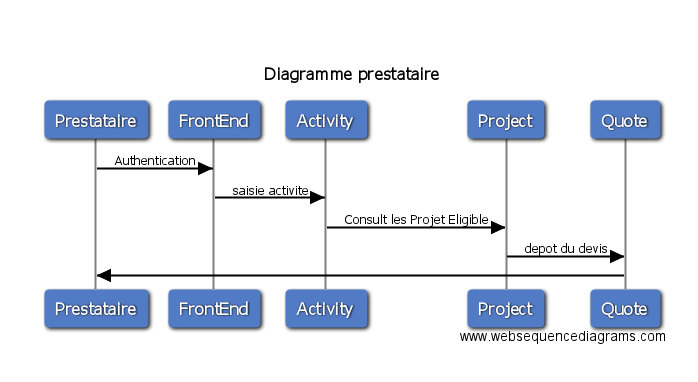
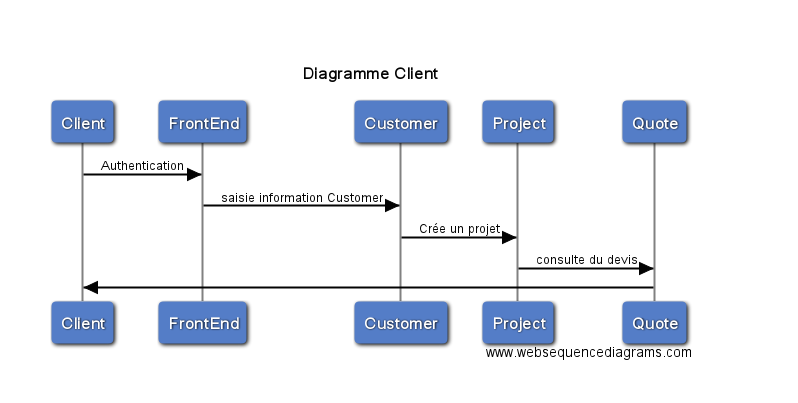
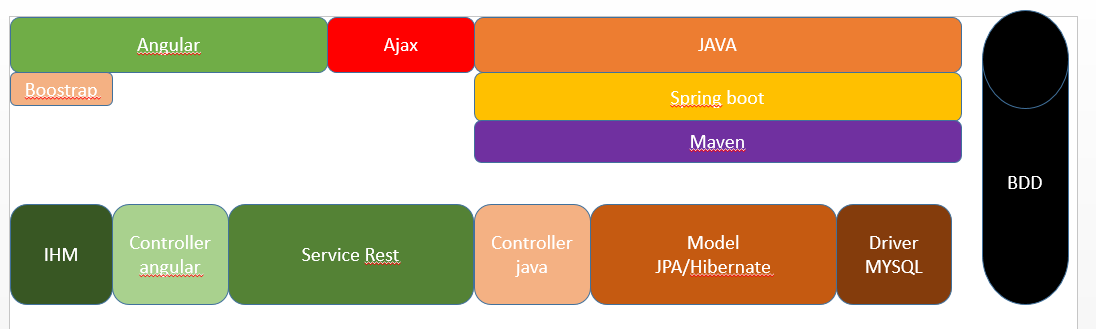


Figure 1 Diagramme de Séquence

Le diagramme de séquence permet de montrer les interactions d'objets dans le cadre d'un scénario d'un Diagramme des cas d'utilisation. Dans un souci de simplification, on représente l'acteur principal à gauche du diagramme, et les acteurs secondaires éventuels à droite du système. Le but étant de décrire comment se déroulent les actions entre les acteurs ou objets.

**Architecture de l’application**

JHIPSTER



Model

Controller

VUE

Figure 3 Architecture application

Dans cette figue, nous schématisons l’architecture de jhipster avec les principaux Framework et l’utilisation des langages de programmation.

**Design pattern**

**Singletons** : Le cœur de Spring est composé de Spring Core : un conteneur qui implémente le motif de conception IoC (Inversion of Control). Ce conteneur prend en charge la création, la gestion du cycle de vie et les dépendances des objets qu'il gère. La définition de ces objets est faite dans la déclaration du contexte de Spring dans un fichier de configuration XML ou partiellement réalisée en utilisant des annotations.

**MVC** : model vue controller

**Factory** : La fabrique permet de créer un objet dont le type dépend du contexte : cet objet fait partie d'un ensemble de sous-classes. L'objet retourné par la fabrique est donc toujours du type de la classe mère mais grâce au polymorphisme les traitements exécutés sont ceux de l'instance créée. Ce motif de conception est utilisé lorsqu'à l'exécution il est nécessaire de déterminer dynamiquement quel objet d'un ensemble de sous-classes doit être instancié.

Il est utilisable lorsque :

Le client ne peut déterminer le type d'objet à créer qu'à l'exécution

Il y a une volonté de centraliser la création des objets

**Story Board**

**Voir dossier STORY BOARD en HTML ainsi que le PDF**

**Sketch**

**Voir vidéo sktech**