|  |  |
| --- | --- |
| IMIE | Projet Assurance |

Projet Assurance

IMIE

|  |  |
| --- | --- |
| Référence : Spécifications fonctionnelles | Version : V01 du 21/03/2015 |
| Auteurs : Groupe CDPNL3 n°3 | Statut : Non Validé |

|  |  |  |
| --- | --- | --- |
|  | **Nom** | **Fonction** |
| **Rédaction** | Groupe CDPNL3 n°3 | Chef de Projet |
| **Validation MOA** | IMIE | Directeur de Projet |
| **Validation MOE** | Groupe CDPNL3 n°3 | Chef de projet |

|  |  |  |
| --- | --- | --- |
| Version | Objet de l’évolution | Date d’émission |
| 1.0 | Première version | 21/03/2015 |

# Table des matières

[Objectif des spécifications f](#h.1fob9te)onctionnelles

[Présentation générale du produit](#h.3znysh7)

[Le produit et son concept](#h.2et92p0)

[Exigences principales](#h.tyjcwt)

[Fonctions](#h.3dy6vkm)

[Chaîne fonctionnelle](#h.1t3h5sf)

[Constituants principaux](#h.4d34og8)

[Documents](#h.17dp8vu)

[Liste des documents de sortie](#h.3rdcrjn)

[Liste des documents d’entrée](#h.26in1rg)

## Objectif des spécifications fonctionnelles

La spécification fonctionnelle est la description détaillée des fonctions d'un logiciel en vue de sa réalisation.

Tout ce qui n’est pas écrit ne sera pas réalisé.

## Présentation générale du produit

Dans les sous-paragraphes on énumérera :

- les besoins fondamentaux qui nécessitent la création du produit,

- les fonctions attendues, dans les grandes lignes, sans présager dans un premier temps de la solution retenue.

*Nota : dans le cas où un CDC (Cahier des Charges) a été rédigé, il comporte déjà certaines de ces informations. Le CDC sera ajouté dans la documentation de référence. La chaîne fonctionnelle ainsi que l’arborescence produit correspondant à la solution proposée est déduite de ces informations.*

## Le produit et son concept

Les équipes du service développement doivent produire et fournir un outil de gestion back office à destination de l'utilisateur permettant :

* La création de devis. On peut y trouver les devis :
* Voitures
* Habitation
* Pour la création d’un devis, un wizard nous offrira plusieurs options :
* Il doit être possible de reprendre un devis en cours (on enregistre l’état entre chaque changement d’écran)
* Il est possible d’annuler un devis en cours de rédaction (donc suppression des étapes enregistrées)
* Les différentes listes (véhicules, marques, etc...) seront fournies par des micro services
* A la fin d’un devis, on l’enregistre dans la base de l’application et si tout se passe bien on publie un résumé dans un micro service (nom, prenom, identifiant, type, garantie, prix)

## Exigences principales

* On doit gérer les rôles (pour l’instant seulement 2 (USER et ADMIN))

Ecran d’administration

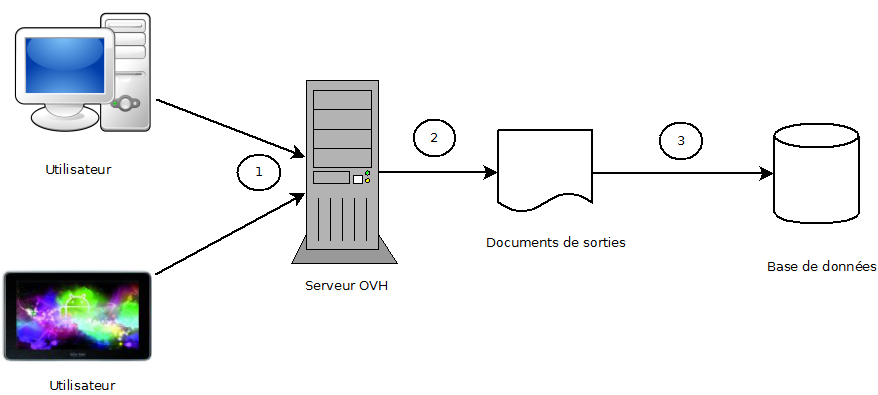
On doit pouvoir voir au travers d’une liste les utilisateurs qui sont en train de réaliser un devis, cette liste doit mentionner l’utilisateur, le nom ainsi que le type du devis et l’étape dans laquelle il se trouve. Cette liste étant en temps réelle, lorsqu’un utilisateur commence un devis ou en finis un, la liste doit se mettre à jour sans rechargement de page.

Bonus secondaire, au clic sur un nom de la liste, on doit pouvoir envoyer un message en temps réel à cet utilisateur, ceci sous la forme d’une fenêtre non modale en bas de l’écran de l’utilisateur.

Afin de réaliser la liste des admins, vous avez le droit d’utiliser un autre Framework UI (Angular ou autre par exemple). Ce Framework doit s’intégrer dans l’environnement des applications (sécurité, etc…)

La communication en temps réelle entre un system back office et front office peut se faire via le bus de Vert.x (il existe un connecteur SockJs entre du JavaScript et un bus Vert.X).

## Chaîne fonctionnelle



1. L'utilisateur rentre les informations des nouveaux documents à partir des formulaires web.
2. Le serveur va traiter les informations et vérifier si elles sont correctes. Si elles le sont, on va créer les documents grâce aux réponses du formulaire.
3. Les fichiers créés vont être stockés en base de données.

## Constituants principaux

Utilisation de Framework :

* JSP pour créer dynamiquement du code HTML, ou tout autre type de page web.
* Application frontal :
* Tomcat
* Spring (MVC/IOC/Data)
* Micro Service :
* Vert.X (qui sera lié à un MySql)

Base de données :

* SGBD : MySQL
* Interface : PhPMyAdmin

Langages utilisés :

* Java
* JavaScript
* SQL

## Utilisateurs

Cinématique d’authentification :

* L’utilisateur accès à l’application via une page de login.
* Submit vers Spring Security
* Authentification en utilisant un micro service et récupération de l’utilisateur ainsi qu’un token JWT généré par le micro service.
* Persistence en session du token généré pour l’utilisateur

Vert.X refusera tout appels hormis l’authentification si aucun token valide n’est passé à m’appels. Les appels entre Spring et Vert.X se feront via des appels Rest (ON reste ouvert à d’autres propositions). Les utilisateurs n’ayant aucun compte accèderont à un formulaire de création pour ensuite se connecter à l’application web.

## Documents

## Liste des documents de sortie

* Devis véhicules
* Devis habitation
* Liste des utilisateurs qui sont en train de réaliser un devis

## Liste des documents d’entrée

* Wizard de création de devis véhicules
* Wizard de création de devis habitation
* Formulaire création utilisateur