



Haute Ecole de Namur Liège Luxembourg—
Implantation IESN

Département économique



Analyse et développement de widgets en
Java avec Vaadin Framework dans le
cadre d'une application de gestion de
données spatiales en temps réel

Travail de fin d'étude présenté en vue de l'obtention du diplôme de bachelier en
Informatique de Gestion

YASSIN LAZRAK

Maitre de stage : Dimitri Duchateau

Promoteur : Stéphane Robert

Année académique 2016-2017

Remerciements

Je remercie tout d'abord monsieur Olivier Dubois de m'avoir permis d'intégrer son entreprise en tant que stagiaire.

Je remercie monsieur Robert Stéphane pour le suivi et ses conseils tout au long du stage et lors de la rédaction de ce travail de fin d'étude.

Je tiens à remercier tout particulièrement l'équipe R&D OSCARS pour m'avoir aidé et guidé dans les différentes tâches qui m'ont été assignées durant le stage.

Je remercie Isabelle Alboraletti pour avoir relu et corrigé les fautes d'orthographies de mon travail de fin d'étude.

Table des matières

1	INTRODUCTION	5
2	PRESENTATION DE LA SOCIETE	6
3	PRESENTATION DU PROJET	7
4	METHODES, OUTILS ET TECHNOLOGIES	8
4.1	Outils et logiciels	8
4.1.1	Schéma synthétique « Outils & logiciels »	8
4.1.2	GIT	9
4.1.3	ONEDRIVE	9
4.1.4	Netbeans	9
4.1.5	IntelliJ IDEA	10
4.1.6	SQL developper	10
4.1.7	Oracle WebLogic Server	11
4.1.8	Oracle Stream Analytics	11
4.2	Technologies	12
4.2.1	Google Web Toolkit	12
4.2.2	Maven	13
4.2.5	Langages de programmation	15
4.2.6	Java	15
4.2.7	JavaScript	17
4.2.8	SQL	17
4.2.9	JSON	18
5	ANALYSE DE L'EXISTANT	19
5.1	Architecture de GIP	19
5.2	Ecrans déjà existants	24
5.3	Base de données	25
6	UN STAGE PROGRESSIF	27
7	ANALYSE	28
7.1	Analyse des beacons	28
7.1.1	Analyse des besoins	28
7.1.2	Conception	29
7.2	Analyse des widgets	32

7.3	Cas d'utilisation	33
7.6	Interface homme-machine	53
7.7	Architecture des gplets	58
8	IMPLEMENTATION	59
8.1	Vaadin	59
8.2	Hiérarchie de fichier d'un projet Vaadin	66
9	CONCLUSION	67
10	BIBLIOGRAPHIE – WEBOGRAPHIE	68

1 Introduction

Ce travail de fin d'étude, réalisé à la fin de la troisième année du baccalauréat en informatique de gestion a pour but d'expliquer le travail et les tâches effectués durant le stage de 15 semaines en immersion au sein de l'entreprise OSCARS.

Ce document est séparé en divers chapitres :

Le premier chapitre présente la société Oscars, dans laquelle j'ai effectué mon stage.

Le deuxième chapitre présente le sujet de stage pour lequel une analyse et une implémentation de widgets m'ont été demandées par la société.

Le troisième chapitre reprend l'ensemble des définitions et explications des technologies et des instruments de développement utilisés.

Le quatrième chapitre décrit le fonctionnement des applications déjà présentes au sein de l'entreprise.

Le cinquième chapitre explique l'avancement progressif du stage, les difficultés auxquelles j'ai été confronté.

Le sixième chapitre traite de l'analyse des fonctionnalités demandées au début du stage, ce chapitre contient : les besoins des utilisateurs, les schémas UML représentatifs et la base de données nécessaire à l'implémentation des widgets.

Le septième chapitre reprend l'ensemble des croquis des écrans utilisateurs créés durant l'analyse des widgets.

Le huitième chapitre éclairci les zones d'ombre sur le framework utilisé durant le stage.

Les 2 derniers chapitres concluront les 15 semaines de stage chez Oscars par une conclusion et des critiques et suggestions.

2 Présentation de la société

OSCARS S.A est une société indépendante créée en 2007 par M. Olivier Dubois et située à Andenne, en région Namuroise. Son équipe est constituée d'une dizaine de personnes : consultants, R&D managers, analystes, développeurs et comptables. La société OSCARS est spécialisée dans les technologies Oracle, notamment dans sa composante spatiale, et dans les domaines des SIG. C'est grâce à ses expertises et ses années d'expérience que l'entreprise s'est hissée au rang d'experte au point de devenir une société reconnue dépassant nos frontières.

La société propose à ses clients des services de consultation dans le domaine Oracle Spatial, mais pas uniquement ! Elle propose aux entreprises une aide à l'optimisation et la rentabilisation des données au sein de leurs systèmes d'informations géographique.

Elle propose également une plateforme de gestion de données spatiale en temps réel, GIP.

Plus précisément, « *GIP est une plateforme d'acquisition et de traitement de données géolocalisées vous permettant de déclencher des alertes en temps réel en fonction d'événements spatiaux préalablement définis. En effet, vous pouvez définir vos propres alertes et événements spatiaux, ce qui génère des workflows associés, que vous pouvez ensuite activer ou désactiver à tout moment en fonction de vos désideratas.*

GIP est un outil générique, non-intrusif et essentiel à la prise de décision pertinente en temps réel en fonction de positions et d'événements réalisés ou non. »¹.

Pour finir, OSCARS S.A met à disposition de tous : des formations, une assistance, des conseils et des suivis de projet grâce à sa renommée et sa fiabilité auprès du géant Oracle.



Figure 1 Logo de la société OSCARS S.A

¹ <http://www.oscars-sa.eu/fr/geo-intelligent-platform>

3 Présentation du projet

L'objectif du stage consiste dans un premier temps à l'analyse et à l'ajout de nouvelles fonctionnalités tierces présentées sous forme de widgets disponible dans différents logiciels déjà existants ou non de la société OSCARS S.A.

Ces fonctionnalités sont présentées dans un tableau de bord, les widgets peuvent varier d'un client à l'autre selon les besoins et les demandes de celui-ci. Une horloge, des températures ou encore une grille des horaires de décollages d'atterrissements d'avions sont les exemples de widgets possible à implémenter. Nombre de widgets additionnels peuvent être ajoutés : graphiques, indicateur de vol, météo, etc.

Un des buts des widgets est de présenter les données récupérées dans une base de données ou d'un flux de données de manière plus lisible pour l'utilisateur, et ainsi permettre une vue globale. Mais aussi aider l'utilisateur à prendre une décision ou bien juste à titre informatif (Horloge par exemple). Certains widgets auront pour objectif d'informer l'utilisateur de manière périodique, en effet, les données récupérées seront mises à jour par intervalle ou lors de la détection d'évènements.

Une des principales spécifications de ces widgets est la configuration. En effet, ceux-ci pourront être paramétrés soit dans un configurator, soit dans le logiciel lui-même. Les widgets possèdent également un système de « drag'n'drop » qui permettra à l'utilisateur de les réordonner.

Dans un deuxième temps, il sera question d'implémentations de BEACON², toujours sous forme de widgets, précédé d'une analyse plus poussée.

Pour finir, il est important de souligner que les widgets devront être un maximum indépendant du logiciel GIP. Ceux-ci seront codés en Java et les classes correspondantes devront se trouver dans des librairies « .jar » afin d'être ajoutés dans n'importe quelle application via le pom.xml.

² Balise radio émettant un son et un clignotement déclenchés lors du survol par un avion

4 Méthodes, outils et technologies

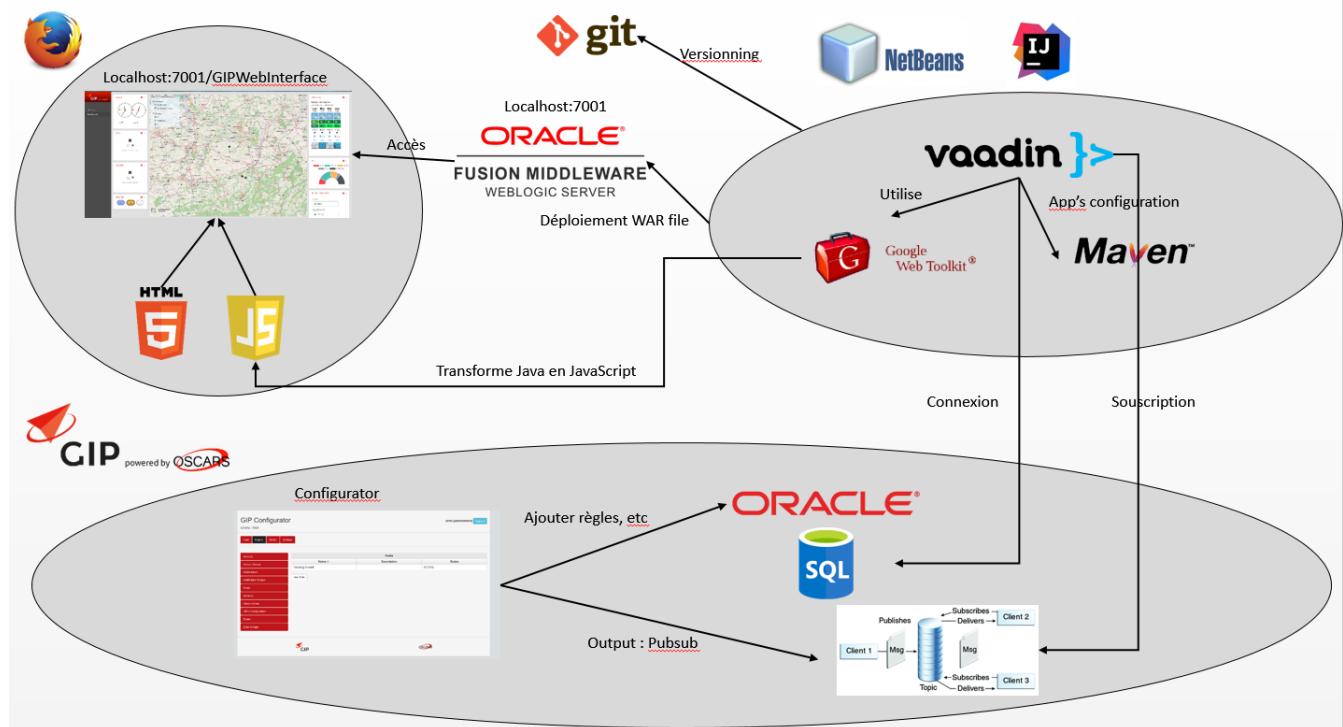
Ce chapitre concerne l'ensemble des méthodes, outils et technologies utilisées durant le stage, chaque outils et technologie sera défini. Un schéma synthétique illustrera la place de chaque outils et technologie durant l'implémentation des widgets.

4.1 Outils et logiciels

OSCAR S.A met à disposition de ses employées différentes plateformes pour le partage de données, la documentation, les rapports de bug, etc. Ces plateformes sont utilisées fréquemment dans l'entreprise.

Au vu des nombreux outils et logiciels utilisés durant le stage, un schéma synthétique englobant ceux-ci sera soumis. Ce schéma illustre la place et l'importance de chacun dans le processus du développement de l'application. Sera suivi ensuite leurs caractéristiques et une brève définition.

4.1.1 Schéma synthétique « Outils & logiciels »



4.1.2 GIT

« Git est un logiciel de gestion de versions décentralisé. C'est un logiciel libre créé par Linus Torvalds, auteur du noyau Linux, et distribué selon les termes de la licence publique générale GNU version 2. En 2016, il s'agit du logiciel de gestion de versions le plus populaire qui est utilisé par plus de douze millions de personnes. »³.



Figure 2 Logo GIT

Git est un logiciel de version, c'est-à-dire qu'il permet de stocker un ensemble de fichiers tout en conservant la chronologie de ceux-ci, en outre, il permet également de récupérer des versions antérieures concernant un fichier.

4.1.3 ONEDRIVE

« Microsoft OneDrive est un ensemble de services en ligne : stockage et applications Word, Excel, PowerPoint et OneNote, dont les fonctionnalités sont toutefois réduites par rapport aux logiciels installés sur un ordinateur. Ce service a été créé en 2007 et a porté les noms Windows Live Folders, Windows Live SkyDrive, SkyDrive et enfin son nom actuel depuis janvier 2014.»⁴.



Figure 3 Logo de Microsoft OneDrive

Onedrive a pour intérêt la grande quantité de données possible à stocker (jusqu'à 1 Terra Octets disponible dans la version Office 365), ainsi que la récupération rapide de données.

C'est aussi un choix alternatif quant à un back-up.

4.1.4 Netbeans

« NetBeans est un environnement de développement intégré (EDI), placé en open source par Sun en juin 2000 sous licence CDDL (Common Development and Distribution License) et GPLv2. En plus de Java, NetBeans permet la prise en charge native de divers langages tels le C, le C++, le JavaScript, le XML, le Groovy, le PHP et le HTML, ou d'autres (dont Python et Ruby) par l'ajout de greffons. Il offre toutes les facilités d'un IDE moderne (éditeur en couleurs, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages Web).



Figure 4 Logo Netbeans

Compilé en Java, NetBeans est disponible sous Windows, Linux, Solaris (sur x86 et SPARC), Mac OS X ou sous une version indépendante des systèmes d'exploitation (requérant une machine virtuelle Java). Un environnement Java Development Kit JDK est requis pour les développements en Java.

³ <https://fr.wikipedia.org/wiki/Git>

⁴ https://fr.wikipedia.org/wiki/Microsoft_OneDrive

NetBeans constitue par ailleurs une plateforme qui permet le développement d'applications spécifiques (bibliothèque Swing (Java)). L'IDE NetBeans s'appuie sur cette plateforme. »⁵

Netbeans offre une panoplie de types de projet : JavaFX, Java, JavaEE, etc. Il est possible d'ajouter des plugins pour, par exemple, modifier l'apparence, ajouter des frameworks (Vaadin, ...), ajouter le support de nouveaux langages de programmation, etc.

De plus, Netbeans offre la possibilité d'utiliser GIT d'une manière simpliste.

4.1.5 IntelliJ IDEA

« IntelliJ IDEA est un IDE Java commercial développé par JetBrains. Il est fréquemment appelé par le simple nom d'« IntelliJ » ou « IDEA ». »⁶

IntelliJ IDEA met à disposition du développeur une panoplie de langages informatique tels que : Java, JavaScript, CoffeeScript, HTML/ XHTML/ CSS, XML/ XSL, Python, ActionScript, Ruby, Groovy, PHP, Scala ou encore Kotlin.

IntelliJ IDEA permet également de gérer un large éventail de technologies et de frameworks : Hibernate / JPA, Google Web Toolkit, Spring, AJAX, EJB, etc.

De même, il permet la gestion des connexions à différents serveurs : GlassFish, Tomcat, Jetty ou encore WebLogic.

Cependant, la gestion de connexion à des serveurs n'est disponible que dans la version Ultime.

4.1.6 SQL developer

« Oracle SQL Developer est un environnement de développement intégré (EDI) multi-plateforme, fourni gratuitement par Oracle Corporation et utilisant la technologie Java (Java Development Kit). C'est un outil graphique permettant d'interroger des bases de données Oracle à l'aide du langage SQL. Oracle SQL Developer permet le développement de A à Z d'applications en PL/SQL, la mise à disposition de feuilles de travail pour exécuter les requêtes et les scripts, une console pour l'administration de bases de données (DBA), une interface pour la génération de rapports (reporting), une solution complète de conception du modèle de données et une interface de migration permettant de migrer les bases de données d'éditeurs tiers vers Oracle.

Oracle SQL Developer supporte les produits Oracle ainsi que des plugins qui permettent de se connecter à des bases de données non Oracle. Oracle SQL Developer fonctionne avec IBM DB2, Microsoft Access, Microsoft SQL Server, MySQL, Sybase Adaptive Server, et les bases de données Teradata. »⁷.

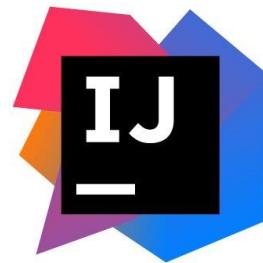


Figure 5 Logo IntelliJ IDEA

⁵ <https://fr.wikipedia.org/wiki/NetBeans>

⁶ https://fr.wikipedia.org/wiki/IntelliJ_IDEA

⁷ https://fr.wikipedia.org/wiki/Oracle_SQL_Developer

Oracle SQL developper est système de gestion de bases de données qui permet de se connecter et de récupérer une base de données distante ou locale. Il permet de faire des requêtes pour interroger celle-ci.

Il sera utile pour la récupération et la persistance des données GIP.

4.1.7 Oracle WebLogic Server

« Oracle WebLogic Server 12c, première plateforme Java d'entreprise native dans le Cloud au monde, permet de tirer pleinement parti des avantages du Cloud computing. Sa capacité unique de mutualisation permet une consolidation massive. Son architecture légère de micro conteneurs permet d'isoler les applications et autorise une portabilité totale entre vos Cloud privé et public. Son architecture haute disponibilité à plusieurs centres de données protège les applications des interruptions. Les innovations des développeurs avec la prise en charge totale de Java EE 7 et Java SE8 optimisent la productivité des équipes DevOps. En outre, vous continuez à bénéficier d'un accès à la même plateforme sur site et dans le Cloud, via Oracle Java Cloud Service, qui s'appuie sur Oracle WebLogic Server. »⁸.

Le serveur sera utilisé pour l'hébergement des librairies, des fichiers war⁹ des applications et des sources de données utilisées dans ces derniers.

4.1.8 Oracle Stream Analytics



⁸ <https://www.oracle.com/fr/middleware/weblogic/index.html>

⁹

4.2 Technologies

4.2.1 Google Web Toolkit

« *Google Web Toolkit (GWT) est un ensemble d'outils logiciels développé par Google, permettant de créer et maintenir des applications web dynamiques mettant en œuvre JavaScript, en utilisant le langage et les outils Java. C'est un logiciel libre distribué selon les termes de la licence Apache 2.0.*



Figure 7 Logo Google Web Toolkit

GWT met l'accent sur des solutions efficaces et réutilisables aux problèmes rencontrés habituellement par le développement AJAX : difficulté du débogage JavaScript, gestion des appels asynchrones, problèmes de compatibilité entre navigateurs, gestion de l'historique et des favoris, etc. GWT est un framework qui laisse la liberté au développeur en ne lui imposant pas une structure trop rigide. Comme son nom l'indique, il s'agit d'une boîte à outils qui offre des solutions permettant de développer plus facilement des solutions web/AJAX, en profitant des outils et compétences Java existants, et en faisant abstraction de la complexité habituellement liée à ce genre de technologies.»¹⁰.

En bref, lorsque l'application développée en Java sera prête à être déployée, le compilateur GWT va traduire le code Java en langage JavaScript qui sera lisible sur tous les navigateurs. Le JavaScript généré par GWT pourra permettre l'interaction avec l'utilisateur ainsi que la manipulation du DOM HTML de sorte que l'interface soit dynamique.

¹⁰ https://fr.wikipedia.org/wiki/Google_Web_Toolkit

4.2.2 Maven

« Apache Maven est un outil pour la gestion et l'automatisation de production des projets logiciels Java en général et Java EE en particulier. L'objectif recherché est comparable au système make sous Unix : produire un logiciel à partir de ses sources, en optimisant les tâches réalisées à cette fin et en garantissant le bon ordre de fabrication.



Figure 8 Logo Maven

Il est semblable à l'outil Ant, mais fournit des moyens de configuration plus simples, eux aussi basés sur le format XML. Maven est géré par l'organisation Apache Software Foundation. Précédemment Maven était une branche de l'organisation Jakarta Project.

Maven utilise un paradigme connu sous le nom de Project Object Model (POM) afin de décrire un projet logiciel, ses dépendances avec des modules externes et l'ordre à suivre pour sa production. Il est livré avec un grand nombre de tâches pré-définies, comme la compilation de code Java ou encore sa modularisation.

Un élément clé et relativement spécifique de Maven est son aptitude à fonctionner en réseau. Une des motivations historiques de cet outil est de fournir un moyen de synchroniser des projets indépendants : publication standardisée d'information, distribution automatique de modules jar. Ainsi en version de base, Maven peut dynamiquement télécharger du matériel sur des dépôts logiciels connus. Il propose ainsi la synchronisation transparente de modules nécessaires. »¹¹.

Maven, qui signifie « Accumulator of knowledge » a pour objectifs :

- Rendre le processus de build plus facile
- Fournir une uniformisation des systèmes de build
- Fournir une qualité quant aux informations d'un projet
- Fournir les meilleures façons de développer
- Permettre une transparence dans la migration de certaines caractéristiques.

Les librairies maven se trouve dans le répertoire local « .m2 ».

Pour installer les widgets dans le répertoire local il faudra ouvrir une console windows dans le fichier « /target » de la librairie et entrer la commande suivante :

```
mvn install:install-file -Dfile= « * » -DgroupId= « * » -DartifactId= « * » -Dversion=1.0 -  
Dpackaging=jar
```

Où les * seront remplacés par les paramètres adéquats préétablis dans le fichier « pom.xml »

¹¹ https://fr.wikipedia.org/wiki/Apache_Maven

En bref, MAVEN est un outil pour la gestion et l'automatisation de production de projets logiciels en Java en général et Java EE en particulier.

4.2.3 Vaadin



Figure 9 Logo du framework Vaadin

Vaadin est un framework de développement d'applications Java web conçu pour créer et maintenir des interfaces web utilisateurs. Vaadin supporte 2 modèles de programmation : le modèle « server-side » et le modèle « client-side ». Le côté serveur permet aux développeurs d'oublier totalement la partie web et de programmer des interfaces web comme une application desktop avec différents outils Java comme SWT, GWT, AWT ou encore Swing. Vaadin nous permet de se concentrer sur la partie logique de l'application, c'est pour cela que Vaadin s'occupe de l'interface utilisateur dans le navigateur ainsi que de la communication AJAX entre le serveur et le client. Il est inutile de connaître les technologies Javascript ou HTML avec Vaadin framework.

Vaadin sera expliqué en détails dans le chapitre concernant l'implémentation.

4.2.4 Publish - Subscribe

Publish – Subscribe est un mécanisme de publication et de souscription de messages dans lequel les émetteurs (publisher) n'envoient pas les messages à des destinataires (subscriber) directement mais dans un topic. Les destinataires se souscrivent à un topic les intéressants sans savoir s'il y a des émetteurs et inversement, les émetteurs ne savent pas s'il y a ou non des destinataires qui ont souscrit à ce topic.



Figure 10 Pattern Publish - subscribe

Les destinataires ayant souscrit à un topic reçoivent les messages dès que le topic à quelque chose à envoyer. Il peut y avoir plusieurs topics par catégorie / sujet.

4.2.5 Langages de programmation

Les langages de programmation les plus utilisés sont Java et SQL. Cependant l'entreprise n'hésite pas à s'adapter selon les besoins. Les logiciels sont codés très souvent à l'aide du framework Vaadin qui permet de créer des interfaces web très intuitive à l'aide de HTML / CSS / JavaScript.

4.2.6 Java

Java est un langage de programmation informatique orienté objet développé par Sun Microsystems apparu pour la première fois en 1995. Une de ses forces est son excellente portabilité à travers les systèmes d'exploitation.

Java possède plusieurs caractéristiques :

- Orienté objet, tout est objet en Java.
- Langage interprété
- Portable
- Multithread
- Dynamique



Figure 11 Logo Java

La huitième version de Java, sortie en 2014, permet plus de facilité aux développeurs, en effet, depuis la nouvelle version de Java il est maintenant possible d'utiliser des expressions lambda, ce qui facilite grandement la chose quant à l'implémentation de classe anonyme.

Java 8 intègre plusieurs nouveautés intéressantes : les expressions lambda, les méthodes default dans les interfaces, les pipelines et les streams, Nashorn JavaScript Engine, une nouvelle API de date, les opérations parallèle, etc.

Java 8 a été la version utilisée lors de l'implémentation des widgets durant le stage.

Voici un exemple de code (repris de poc¹²) qui utilise un stream et des expressions lambda qui démontrent l'efficacité de ceux-ci par rapport aux anciennes versions de Java.

Les images ci-dessous illustrent le code qui a pour but d'afficher et de trier, par âge, les personnes qui ont plus de 21 ans.

Une liste de personnes est mise à disposition, chaque personne possède un âge et un prénom.

```
List<Person> people = Arrays.asList(new Person("Dan", 23), new Person("Laura", 22), new Person("Billy", 50), new Person("George", 21));

// JAVA 8
List<Person> nameSortedByAge = people.stream()
    .filter(x -> x.getAge() > 21)
    .sorted(comparing(Person::getAge))
    .collect(Collectors.toList());

System.out.println("--> Name sorted by age and > 21");
System.out.println("\nJAVA 8 : " + nameSortedByAge);
```

¹² Proof of concept

```

//JAVA < 7

List<Person> peopleSorted = new ArrayList<>();

for (Person person : people) {
    if (person.getAge() > 21) {
        peopleSorted.add(person);
    }
}

Collections.sort(peopleSorted, new Comparator<Person>() {
    @Override
    public int compare(Person t, Person t1) {
        return t.getAge().compareTo(t1.getAge());
    }
});

System.out.println("\nJAVA 7 : " + peopleSorted.toString() + "\n");

--> Name sorted by age and > 21

JAVA 8 : [Person{name=Laura, age=22}, Person{name=Dan, age=23}, Person{name=Billy, age=50}]

JAVA 7 : [Person{name=Laura, age=22}, Person{name=Dan, age=23}, Person{name=Billy, age=50}]

```

Voici quelques explications concernant l'implémentation du code.

Un stream est toujours construit à partir d'une source de données et n'altère jamais celle-ci. Dans le cas ci-dessus, il est créé à partir d'une liste de personnes.

Voici les opérations que le stream effectue sur la liste

- Filter() : Filter est une méthode qui accepte un prédictat permettant de filtrer les éléments du stream. C'est une opération intermédiaire, ce qui signifie que l'on peut appeler d'autres méthodes du stream par la suite.
- Sorted() : La méthode sorted est une opération stream intermédiaire qui retourne un stream trier par ordre croissant à moins qu'on lui passe un Comparator. Dans ce cas-ci, elle trie par ordre croissant les personnes selon leurs âges.
- Collect() : Cette méthode permet de collecter les données récupérées sous différentes forme (liste, hashmap, concaténation de string, etc).

Les opérations terminales servent à terminer un stream, c'est le cas notamment pour la méthode « forEach » qui permet de parcourir une à une toutes les données présentent dans le stream.

Aucune méthode ne peut être appelée après les opérations terminales tandis que les opérations intermédiaire nous permettent de continuer des opérations sur le stream (jusqu'à un terminal).

4.2.7 JavaScript

« *JavaScript est un langage de programmation de scripts principalement employé dans les pages web interactives mais aussi pour les serveurs avec l'utilisation (par exemple) de Node.JS. C'est un langage orienté objet à prototype, c'est-à-dire que les bases du langage et ses principales interfaces sont fournies par des objets qui ne sont pas des instances de classes, mais qui sont chacun équipés de constructeurs permettant de créer leurs propriétés, et notamment une propriété de prototypage qui permet d'en créer des objets héritiers personnalisés. En outre, les fonctions sont des objets de première classe.*



Figure 12 Logo JavaScript

JavaScript a été créé en 1995 par Brendan Eich. Il a été standardisé sous le nom d'ECMAScript en juin 1997 par Ecma International dans le standard ECMA-262. Le standard ECMA-262 en est actuellement à sa 7^e édition. JavaScript n'est depuis qu'une implémentation d'ECMAScript, celle mise en œuvre par la fondation Mozilla. L'implémentation d'ECMAScript par Microsoft se nomme JScript, tandis que celle d'Adobe Systems se nomme ActionScript. »¹³.

Avant l'apparition du JavaScript les sites web étaient constitués de pages statiques contenant du HTML et du CSS et leur apparence ne changeait pas. Le JavaScript a permis une dynamisation des pages web grâce à sa faculté d'interagir avec le DOM et ainsi permettre le DHTML (Dynamic HTML).

JavaScript est une action exécutée par le navigateur web, pouvant modifier l'apparence de la page web.

JavaScript est désormais incontournable dans le monde du web, il est partout, au point de se démocratiser au sein de divers environnements comme Node.JS, les applications pour smartphones / tablettes, les applications Windows 8 ou encore les logiciels multiplateforme.

4.2.8 SQL

« *SQL (sigle de Structured Query Language, en français langage de requête structurée) est un langage informatique normalisé servant à exploiter des bases de données relationnelles. La partie langage de manipulation des données de SQL permet de rechercher, d'ajouter, de modifier ou de supprimer des données dans les bases de données relationnelles.*



Figure 13 Logo SQL

Outre le langage de manipulation des données, la partie langage de définition des données permet de créer et de modifier l'organisation des données dans la base de données. La partie

¹³ <https://fr.wikipedia.org/wiki/JavaScript>

langage de contrôle de transaction permet de commencer et de terminer des transactions, et la partie langage de contrôle des données permet d'autoriser ou d'interdire l'accès à certaines données à certaines personnes.

Créé en 1974, normalisé depuis 1986, le langage est reconnu par la grande majorité des systèmes de gestion de bases de données relationnelles (abrégé SGBDR) du marché.

SQL fait partie de la même famille que les langages SEQUEL (dont il est le descendant), QUEL (intégré à Ingres) ou QBE (Zloof). »¹⁴.

SQL permet de créer des scripts pour interroger une base de données de manière spécifique.

La plateforme GIP utilise une base de données utilisant du SQL et plus particulièrement la composante spatiale d'Oracle.

4.2.9 JSON

« JSON (pronunciation : /ʒi.sɔn/ (fr) ou /dʒeɪ.sɔn/ (fr)¹, ou /'dʒeɪ.sən/ (en)²), ou JavaScript Object Notation, est un format de données textuelles dérivé de la notation des objets du langage JavaScript. Il permet de représenter de l'information structurée comme le permet XML par exemple. Créé par Douglas Crockford entre 2002 et 2005, il est décrit par la RFC 7159 de l'IETF. »¹⁵.

Les fichiers JSON représentent un ensemble de données avec des paires de clé - valeur, ces éléments sont représentés sous forme de tableau, d'objet ou bien de type : booléen, nombre (int, double, float, ...), null, objet, tableau ou encore chaînes de caractères.

JSON est indépendant des langages de programmation, c'est pourquoi JSON est intégré dans 55 langages de programmation différents. Le format JSON permet de communiquer des informations entre applications dans un format commun.

```
{
  "firstName": "John",
  "lastName": "Smith",
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021"
  },
  "phoneNumber": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "fax",
      "number": "646 555-4567"
    }
  ]
}
```

Figure 14 Exemple du format JSON

¹⁴ https://fr.wikipedia.org/wiki/Structured_Query_Language

¹⁵ https://fr.wikipedia.org/wiki/JavaScript_Object_Notation

5 Analyse de l'existant

Ce chapitre rassemble l'ensemble des informations à propos de la plateforme GIP misent à disposition durant le stage.

Ce chapitre reprend l'architecture de la plateforme GIP, une description détaillée de celle-ci ainsi que de ses composants, les différentes règles de détection présentent dans GIP, les écrans utilisateurs déjà existants et la base de données associée à GIP.

5.1 Architecture de GIP

Petit rappel, comme décrit dans le chapitre « Présentation de la société », GIP est une plateforme d'acquisition de données géolocalisées en temps réel, son rôle consiste à aller chercher des données brutes auprès d'une source, de faire des calculs sur celles-ci et ensuite de les dispatcher selon différents types de canaux de transmissions (SMS, base de données, email, etc).

GIP une plateforme très modulable et robuste, cependant, cette modularité implique une architecture complexe et difficilement compréhensible. Il est donc indispensable de bien connaître son fonctionnement pour y insérer des fonctionnalités supplémentaires, notamment des widgets.

Le schéma ci-dessous illustre l'architecture de la plateforme d'acquisition GIP dans sa forme la plus simpliste.

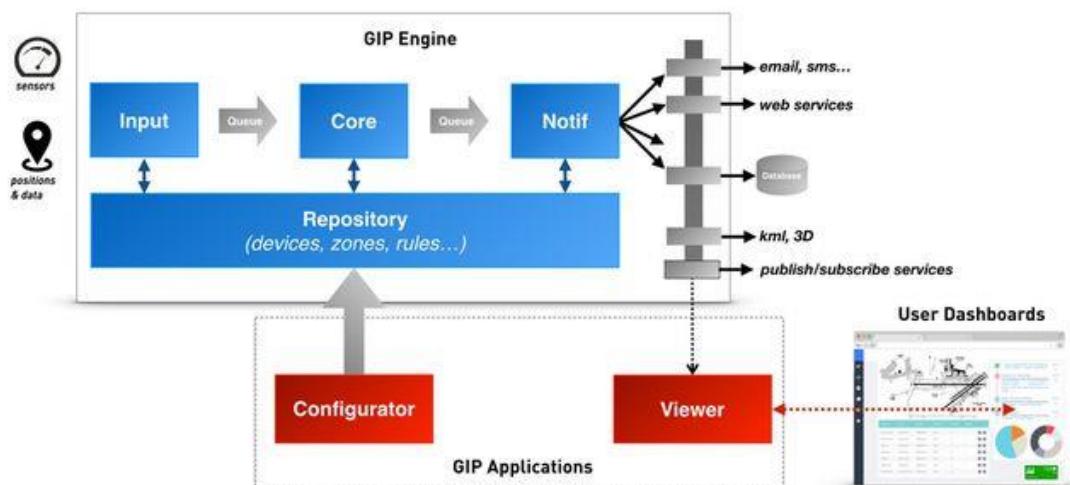


Figure 15 Schéma simplifié du fonctionnement de GIP

5.1.1 GIP Engine

Input

La partie « input » représente un point d'entrée de toutes les données brutes envoyées par des devices. Ceux-ci sont des objets qui ont pour but d'enregistrer ou calculer des mesures qui seront ensuite envoyées.

Les données qui sont reçues dans l'application broker vont être filtrées et renvoyées par la suite au prochain module, GIP Core.

Core

Gip Core est le module intelligent de la plateform GIP. C'est lui qui contient les règles qui vont déclencher les alertes qui vont être transmis et interpréter par le module GIP Notification.

Les notifications peuvent être de différents type, notamment sms, email, web services, publish – subscribe, etc.

Elles constituent une source de données réceptive par des applications.

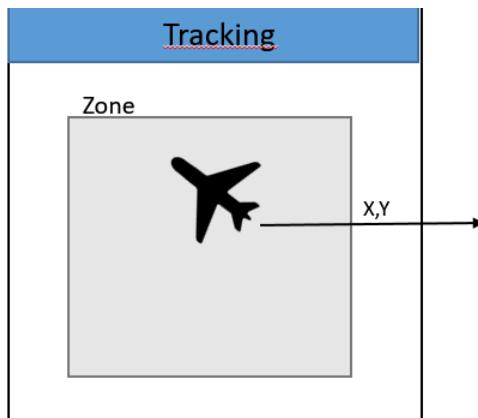
Règles de détection

Les évènements peuvent être détectés selon différents critères énumérés ci-dessous.

Tracking

La règle de détection « Tracking » permet de déclencher un évènement lorsque l'on reçoit des données d'une entité.

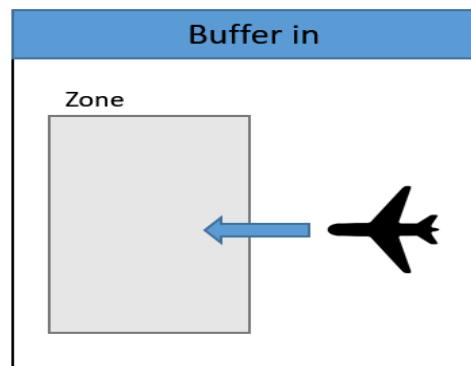
C'est grâce à cette règle que le giplet « flight information » peut fonctionner.



Buffer in

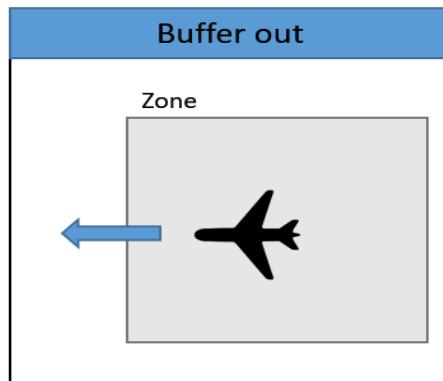
La règle de détection « Buffer in » permet de déclencher un évènement lorsqu'une entité entre dans une zone d'intérêt.

Cette règle est présente dans le giplet « beacon », effectivement, les beacons seront déclenchés lorsqu'un avion entrera en interaction avec la zone correspondante de celui-ci.



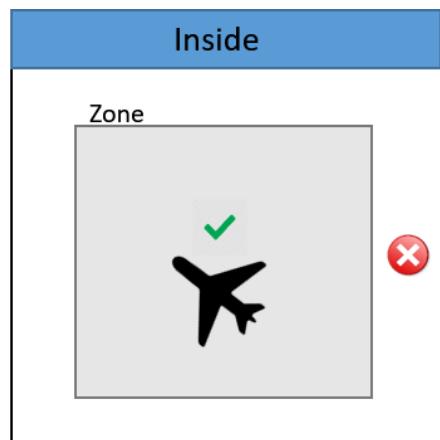
Buffer out

La règle de détection « Buffer out » permet de déclencher un évènement lorsqu'une entité sort d'une zone d'intérêt.



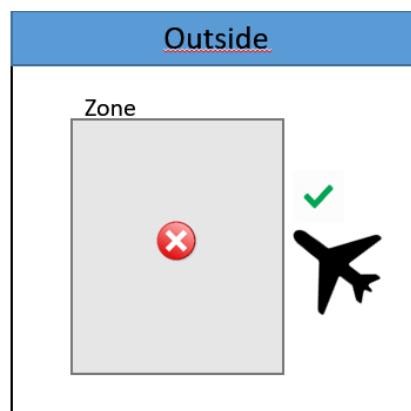
Inside

La règle de détection « Inside » permet de savoir si une entité se trouve actuellement dans une zone.



Outside

La règle de détection « Outside » permet de savoir si une entité se trouve actuellement en dehors d'une zone.



Greater

La règle de détection « Greater » permet de détecter un évènement lorsqu'une valeur scalaire est supérieur à un paramètre au préalablement choisi.

Lower

La règle de détection « Lower » permet de détecter un évènement lorsqu'une valeur scalaire est inférieure à un paramètre au préalablement choisi.

Range Inclusive

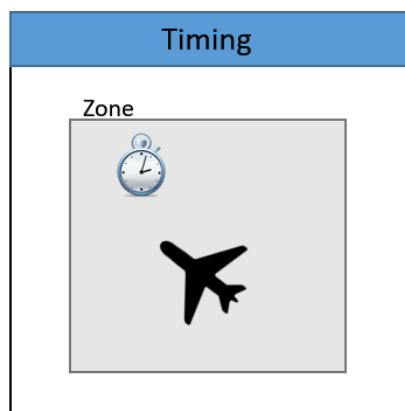
La règle de détection « Range inclusive » permet de détecter un évènement lorsqu'une valeur se trouve entre une borne supérieure et une borne inférieure fixées.

C'est notamment grâce aux 2 règles vues précédemment que l'on pourra déclencher des évènements si l'on veut être alerté lorsqu'un avion dépasse X km / h, nous alerter si l'avion ne va pas assez vite (prévention de problème) ou encore si l'on veut qu'une entité reste dans une fourchette de vitesse.

Timing

La règle timing calcule la différence de temps entre l'entrée et la sortie d'une entité dans une zone.

Cette règle permettra, dans le futur, d'optimiser et d'organiser au mieux les entités présentes. Par exemple, cela permettra d'optimiser le temps qu'un avion passe sur la piste.



Repository

Le repository stocke les paramètres de configuration des modules (=base de données). Les différents modules de GIP se basent sur une configuration déjà présente en DB.

5.1.2 GIP Application

Configurator

Le configurator est une interface web qui permet de configurer toutes les composantes interne de GIP. Il permet donc de configurer la partie « repository » de la plateforme.

Par exemple, le configurator permet de détecter des évènements et lever des notifications selon des règles préconfigurées.

The screenshot shows the GIP Configurator interface. On the left, a sidebar lists various configuration categories: Devices, Device Groups, Notifications, Notification Groups, Rules, Services, Subscriptions, URLs Configuration, Zones, and Zone Groups. The 'Rules' category is selected. The main panel displays a table of rules:

Name	Description	Status
Beacon	Tracking Aircraft	ACTIVE

Below the table, several input fields are shown for configuring a new rule:

- Name: BEACON
- Display Name: Beacon
- Description: (empty text area)
- Service: Raspberry Service
- Detection Type: Buffer In
- Notification Group: Beacon
- Status: ACTIVE

At the bottom, there is a table titled "Attribute value" with two rows:

Name	Value
SPATIAL_CONTEXT *	2D
ZONES_GROUP *	BEACON

Buttons at the bottom include "Save" (green), "Cancel", and "Delete" (red).

L'image ci-dessus illustre la configuration d'une règle qui permet de détecter un évènement lorsque qu'un avion entre en contact avec une zone (règle de détection « Buffer in »).

Viewer

Le viewer représente n'importe quelle application situé client qui pourra se connecter à une source de données et afficher les notifications, évènements reçu grâce à la plateforme GIP au sein d'une interface web.

Le viewer est considéré comme une un portail géographique qui prend des data sources produites par le moteur GIP et interprète les données reçues de GIP notif.

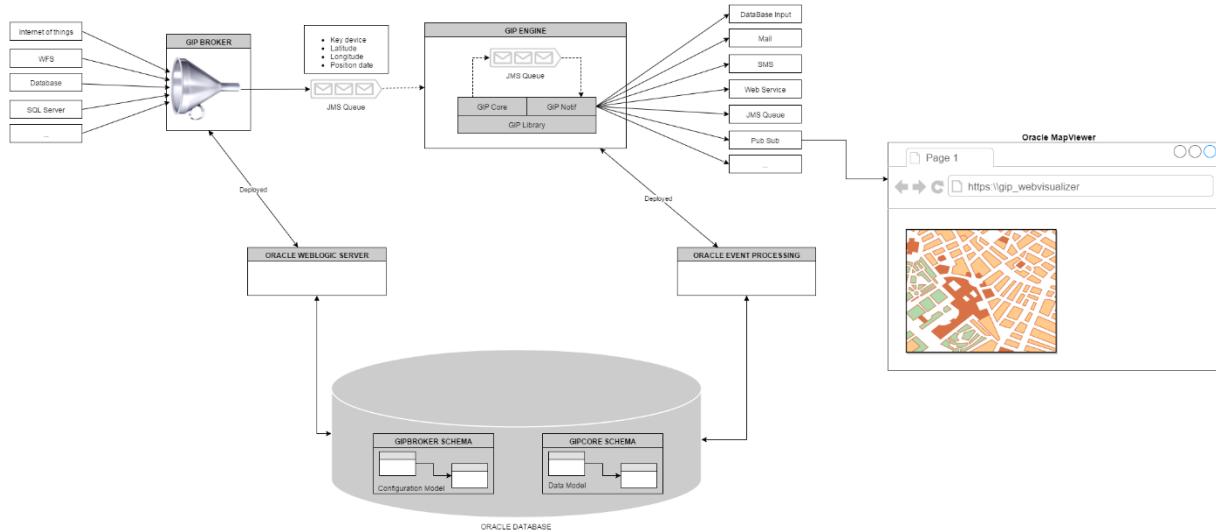


Figure 16 Schéma complet du fonctionnement de GIP

5.2 Ecrans déjà existants

5.2.1 Page d'accueil



Figure 17 Page d'accueil déjà existante de l'application GIP

5.2.2 Onglet « Dashboard »

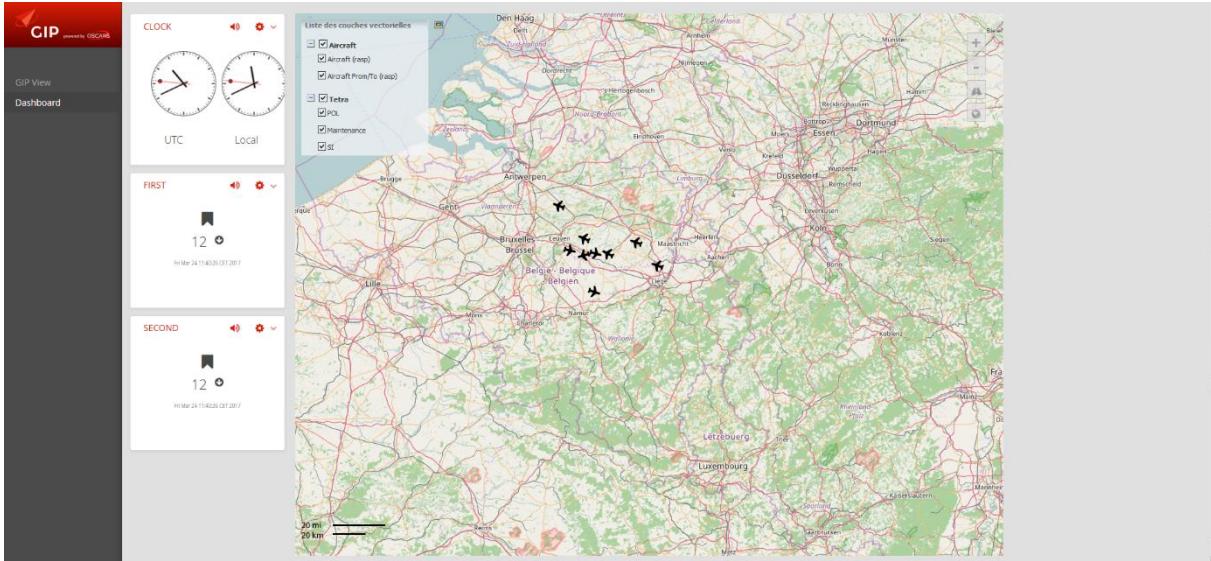


Figure 18 Dashboard de l'application GIP déjà existant

5.3 Base de données

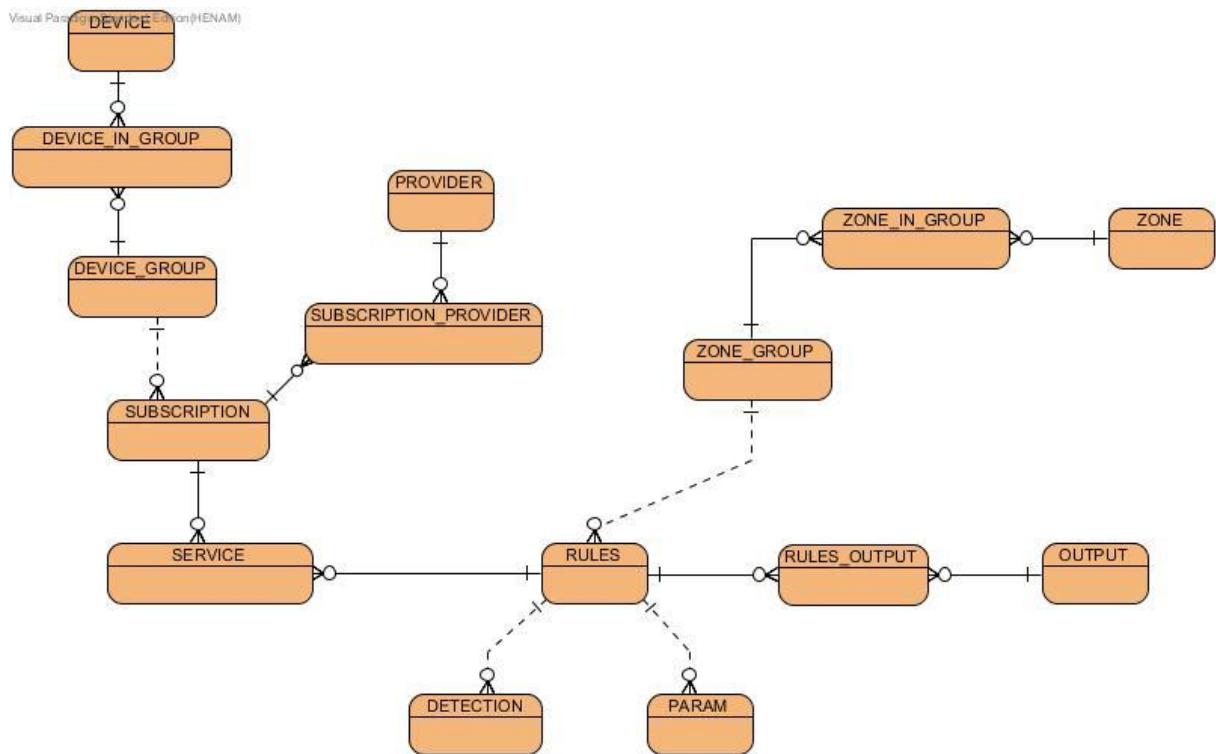


Figure 19 Base de données simplifiée

Pour mieux comprendre le modèle de données ci-dessus imaginons qu'il existe 3 parties distinctes, une partie « input » qui correspond au broker et s'occupe de la source des données et effectue un premier filtre, une partie « output » qui représente la sortie des données selon une configuration et s'occupe de faire un deuxième filtrage et le service qui serait l'union entre les 2 parties.

On peut voir le service comme une boite aux lettres, on veut s'abonner à plusieurs presse (=souscription à un provider) : le soir, la libre, l'echo, etc mais ne recevoir que certains type de journaux, on ne veut pas les magazines ou les publicités (=device group).

Il n'y a que par la boite aux lettres que l'on peut recevoir les journaux.

Lorsque que l'on relève la boite aux lettres on effectue un deuxième filtre (=rules), on ne veut que les journaux qui sont relatif à notre région (=zone) et qui parle d'économie.

Une fois que tous les critères sont respectés on peut commencer à lire.

La table output est un canal de sortie des informations précédemment mâchées et travaillées. Plusieurs canaux sont disponibles depuis la plateforme GIP.

6 Un stage progressif

Durant le stage, l'apprentissage des technologies utilisées par l'entreprise s'est fait progressivement. L'avancement progressif s'est accentué en fonction des technologies de plus en plus complexes utilisées.

Tout d'abord, la familiarisation avec le framework « Vaadin » s'est opérée dès le début du stage, lorsqu'il était question de faire fonctionner et de modifier par la suite le code source d'une application Vaadin récupérer sur github.

Les premières modifications apportées au code source du projet ont été précédées de documentation sur le framework en question. Ces modifications portaient sur la réalisation de graphique en temps réel et de reporting en utilisant une librairie externe « ChartJS ».

Cette librairie met à disposition un large choix d'implémentation de graphique sous différentes formes telles que le graphique en barre, le graphique en tarte, le graphique en « donut », le graphique en ligne, le graphique en bulles, le graphique en radar, etc.

Deuxièmement, la récupération du tableau de bord « GIP » déjà existant. Etant donné que celui-ci accède à une base de données il a été essentiel de pouvoir comprendre et reproduire ces mêmes comportements. L'architecture d'accès à la base de données est semblable à celle du design pattern « DAO ».

Vient ensuite l'analyse et l'implémentation des widgets. Plusieurs widgets ont été implémentés de manière simple, notamment le cas pour le widget météo. Cependant, cette implémentation résulte d'une analyse remplie de questions (choix de la source de données, format, affichage, etc).

L'implémentation du widget « Flight information » qui récupère l'ensemble des données en temps réel d'un avion lors d'un clic a demandé d'une part, une compréhension du pattern « publish-subscribe » et d'autre part une compréhension de la configuration globale de l'application GIP ainsi que ses composants, notamment pour le configurator.

Enfin, l'analyse et l'implémentation du widget beacon s'est montré particulièrement plus complexes. En effet, le premier problème qui s'est posé est l'incohérence dis de « multi-persistiance », la persistiance permet la configuration d'un accès à une base données. L'application GIP et la librairie du widget beacon ne peuvent pas avoir 2 persistiance.xml différente car le framework JPA ne peut pas les distinguer. La solution trouvée consiste à utiliser des « prepardeStatement » pour l'accès à la base de données dans la librairie beacon.

En conclusion, les technologies utilisées durant le stage sont toutes aussi variées que complexes et leur implémentation demande réflexion poussée.

7 Analyse

Durant mon stage, 2 analyses m'ont été soumises, l'une se penche sur l'implémentation de widgets sur un tableau de bord et l'autre se veut plus précise quant à l'implémentation du widget « beacon ». Ce chapitre va donc être scindé en 2 parties, d'une part pour l'analyse des beacons en particulier et d'autre part pour l'analyse et l'implémentation des widgets.

Même si l'analyse du widget « beacon » ne diffère pas beaucoup des autres widgets, il est indispensable de savoir comment fonctionne un beacon marker et comment les données seront traitées et reçues.

Une partie importante du stage a été consacrée à l'analyse et l'implémentation de ces beacons.

7.1 Analyse des beacons

7.1.1 Analyse des besoins

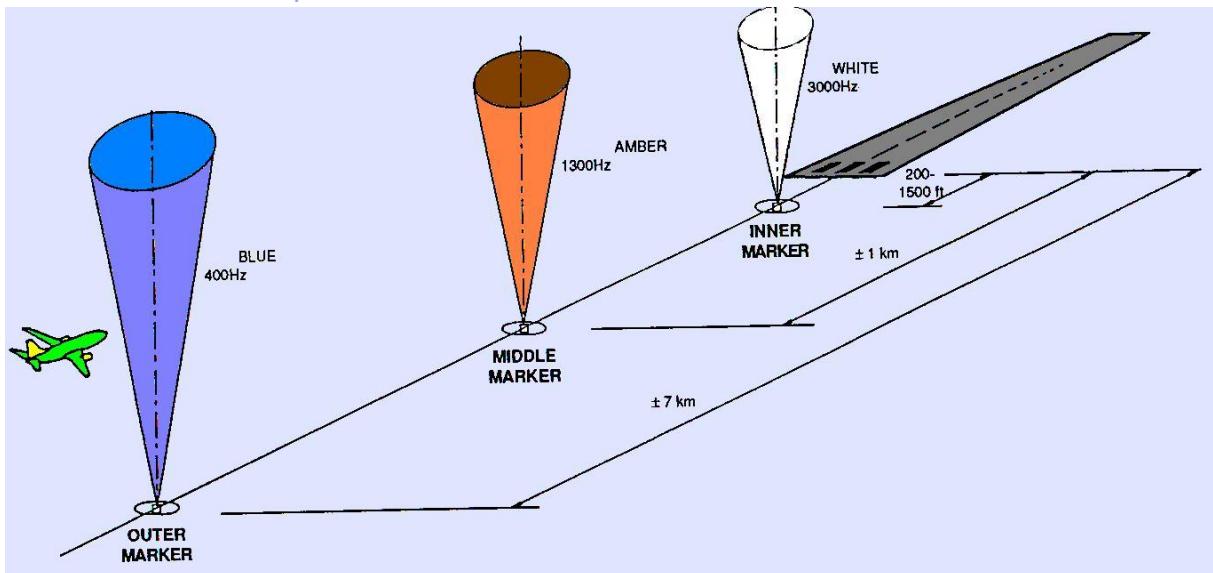


Figure 20 Représentation des beacons

Qu'est-ce qu'un beacon ?

« La radioborne Marker est une radiobalise VHF utilisée dans l'aéronautique permettant au pilote de déterminer à quelle distance est le début de la piste. Les radiobornes Marker de repérage fonctionnent sur la fréquence assignée de 75 mhz et envoient un faisceau étroit radioélectrique en direction verticale. Quand un avion vole au-dessus d'une radioborne Marker, le signal radioborne active l'instrument d'indicateur de radioborne Marker puis l'alarme clignote sur le tableau de bord avec une alarme sonore. »¹⁶

Pour faire simple, un beacon est une radiobalise qui émet des faisceaux verticaux dans le but de disposer des informations sur un avion lorsque celui-ci passe outre et ainsi procéder à des vérifications pendant la phase d'atterrissement ou de décollage.

¹⁶ https://fr.wikipedia.org/wiki/Radioborne_Marker

Les markers sont au nombre de trois :



Figure 16 Voyant Outer

- Outer marker, représenté en bleu, il est le plus loin situé par rapport à la piste, 8km.



Figure 17 Voyant Middle

- Middle marker, représenté en jaune, il est celui qui se trouve entre l'Outer et l'Inner, à 1 km de la piste.



Figure 18 Voyant Inner

- Inner marker, représenté en blanc, il est celui qui se trouve le plus proche de la piste, à 100m.

Le survol de l'un d'entre eux déclenche un allumage des voyants sur le tableau de bord de l'application web ainsi qu'un son caractéristique.

7.1.2 Conception

Pour pouvoir créer le giplet « beacon marker » il faut tout d'abord faire quelques configurations. Celles-ci se font via le configurator et se constituent de plusieurs étapes.

Le diagramme d'activités ci-dessous illustre les différentes étapes à suivre pour accuser une bonne réception des données dans notre application web à travers pubsub.

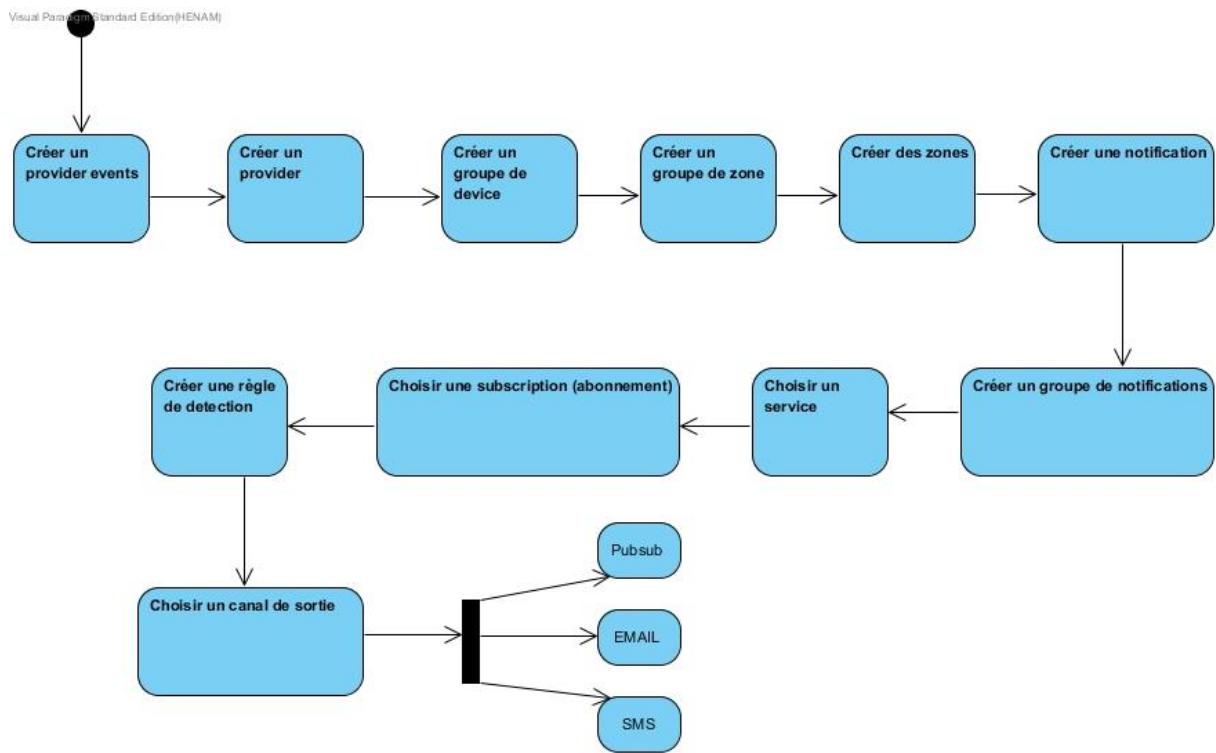


Figure 21 Configuration générique d'une sortie pubsub

Lors du développement, les beacons seront remplacés par des communes de Belgique au vu du manque des données, en effet, il est impossible de recevoir des données précises concernant les avions quant à la localisation de l'entreprise. Les avions captés ont une altitude de minimum 10.000 mètres alors que le beacon le plus loin de la piste se déclenche lorsque la distance entre celui-ci et l'avion atteint le seuil d'altitude de 500m.

Les tests ont donc été réalisés avec des données factices.

Lorsque cette distance de 500m est atteinte une alerte est déclenchée et envoyé sous forme de JSON via un canal pubsub. Une fois que l'application –déjà souscrite à ce pubsub- reçoit la notification, elle va comparer les données reçues et les corrélées avec les choix de l'utilisateur, si les données match alors le beacon correspondant à la notification clignotera et émettra un son l'identifiant.

Le tableau ci-contre correspond aux étapes de configuration d'une sortie pubsub. A chaque étape correspond une brève explication.

Etapes	
Créer un provider events	Définition des données envoyées
Créer un provider	Choisir une source de données
Créer un groupe de device	Les avions sont le centre d'inétrêts
Créer un groupe de zone	Rassembler les beacons par piste
Créer des zones	Créer l'ensemble des beacons
Créer une notification	Une notification pubsub beacon
Créer un groupe de notification	Notification à partir de la notification créée précédemment
Choisir un service	Le service du raspberry PI
Choisir une subscription	Souscription au raspberry PI
Créer une règle	Déclenche la règle lorsqu'un avion entre dans la zone du beacon (corrélation entre la position de l'avion et la GEOM2D du beacon)
Choisir un canal de sortie	Dans ce cas-ci, une sortie pubsub

Les zones et les groupes de zone sont reprises dans des fichiers .shp et ensuite convertit en SQL grâce à map-builder et qgis qui sont des logiciels de cartographie.

7.2 Analyse des widgets

OSCARS S.A veut mettre à disposition de ses clients des mini-outils de gestion, communément appelé « giplet », pour condenser et synthétiser toute l'information sur une seule page. Les informations présentes peuvent être des indicateurs de plusieurs types, notamment économique, financier, commercial, ressource humaine ou encore informatif.

En premier lieu, il faut savoir que les données traitées par GIP sont d'une part, reçues par la base de données et d'autre part, reçues en temps réel par des devices. Les données reçues sont encodées dans un format JSON.

Ensuite, la technologie utilisée pour recevoir ces messages JSON en temps réel dans l'application Vaadin n'est autre que le pattern « Publish – Subscribe », défini dans le chapitre « Méthodes, outils et technologies ».

Enfin, chaque utilisateur acquerra des droits. Pour cela, les utilisateurs sont créés auparavant par un administrateur et enregistrés dans la base de données. Les différents types de droits donnent aux utilisateurs accès ou non à des fonctionnalités, configuration de l'application.

Par exemple, l'ajout, la paramétrisation et la suppression visuelle d'un giplet s'effectuent dans l'application web « GIP » par un client tandis que rendre disponible un giplet dans une application web et la suppression dans celle-ci se font par un administrateur.

L'analyse se porte sur plusieurs giplets, tels que la météo, des informations de vols, des beacons, ou encore des graphiques.

Chaque giplet est composé d'une configuration unique qui permet de modifier son contenu.

Dans la suite de l'analyse, chaque configuration de giplet est suivie d'un scénario permettant de comprendre comment ceux-ci sont configurables et de pré conditions nécessaires.

En outre, il est important de préciser qu'il peut coexister 2 types de graphique parmi les giplets. Le premier type de graphique se caractérise par son critère de données reçues en temps réel et le deuxième type se singularise par la présence de données déjà disponible.

La configuration des graphiques met en évidence 2 éventualités.

La première est la suivante : le client connaît les données et sait précisément quel type de graphique il souhaite avoir sur son tableau de bord. Il est donc du devoir du développeur de programmer ces graphiques auparavant et de les proposer en tant que giplet dans une liste. Toutefois, le graphique ne sera pas paramétrable si ce n'est que le type (tarte, donut, ligne, barre, etc).

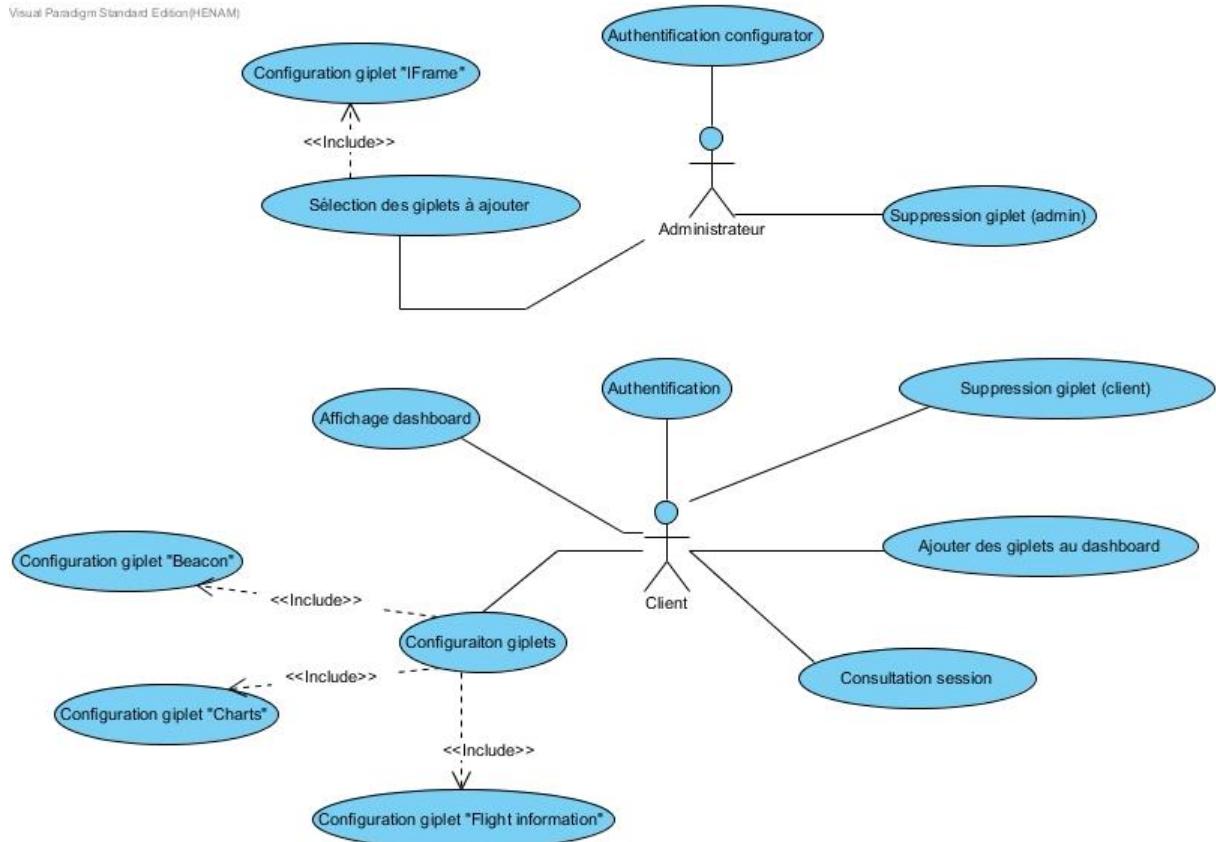
La deuxième éventualité confère au client des graphiques paramétrables : source de données, couleur, nom des axes, animations, etc. Ce qui laisse aux clients une liberté de choix quant à ses graphiques, il n'est pas fermé sur ces choix et créer des graphiques avec les données qu'il juge utile.

Chacune des éventualités amène son lot de complication et / ou de contraintes. Notamment pour les graphiques « pré programmer », il en relève une contrainte et ne laisse que peu de choix à l'utilisateur quant à la liberté des données. Néanmoins cette éventualité permet une simplification du code et de l'architecture de l'application.

Pour conclure, les données devraient être enregistrées dans une base de données pour pouvoir réutiliser les données ultérieurement. Notamment pour les statistiques.

7.3 Cas d'utilisation

Le diagramme des cas d'utilisation représente l'ensemble des fonctionnalités analysées nécessitant une implémentation.



Acteurs

1. Client

Personne lambda de l'entreprise.

2. Administrateur

Personne responsable de certaine fonctionnalités présente dans l'application.

7.3.1 Authentification

Lorsqu'un utilisateur désire utiliser l'application il doit se rendre sur l'url correspondant fourni.

Avant son utilisation l'utilisateur doit pouvoir s'authentifier auprès de l'application web, si le login et le mot de passe sont corrects, l'utilisateur accèdera à la page d'accueil. Sinon, un message d'erreur survient pour lui renseigner les champs qui ne sont pas correct.

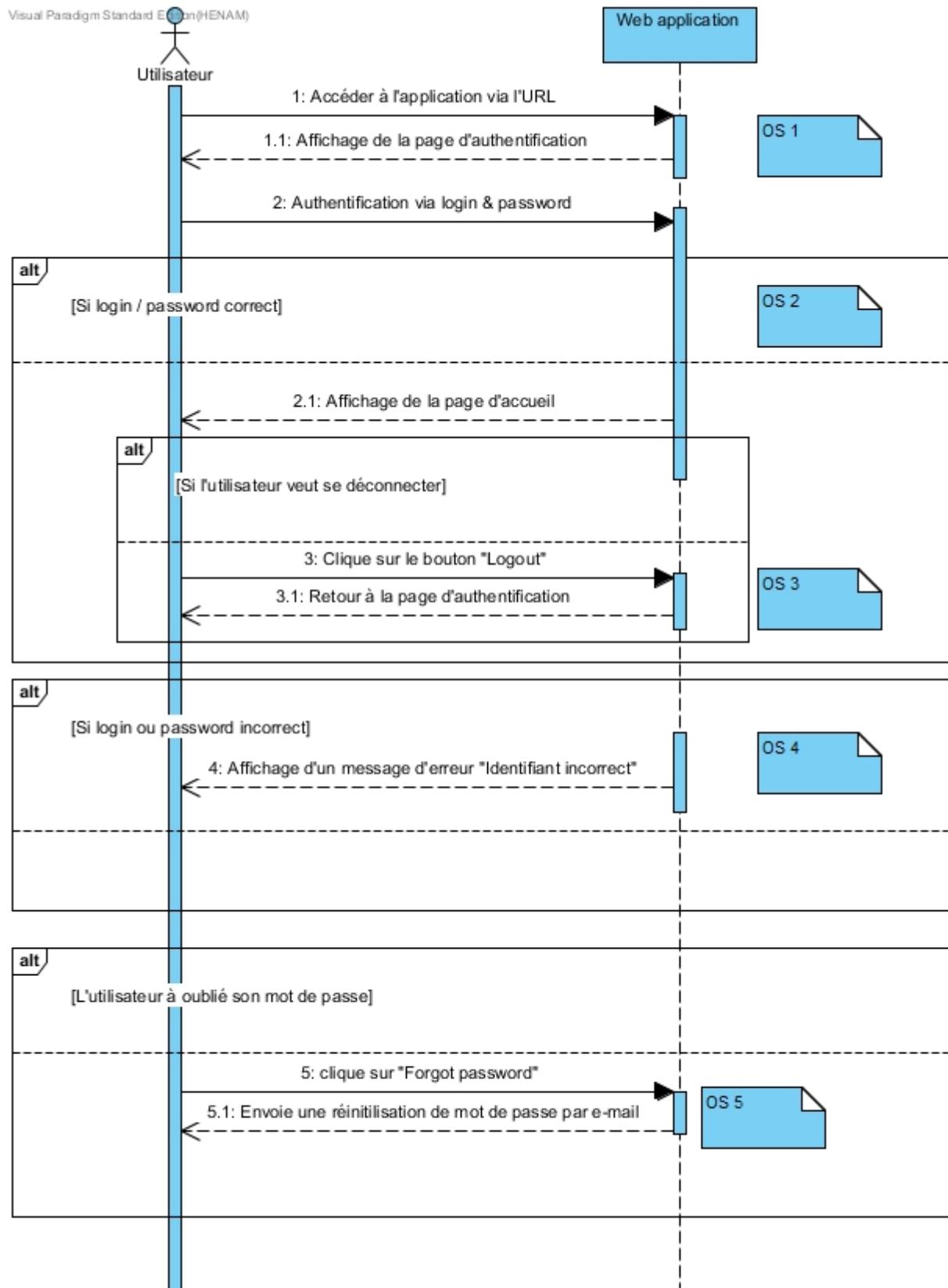
L'utilisateur à la possibilité (s'il a oublié son mot de passe) de le réinitialiser en cliquant sur « forgot password », un e-mail lui sera envoyé pour la réinitialisation.

Pré conditions

L'utilisateur est enregistré dans la base de données avant de se connecter pour la première fois à l'application web

Scénario	
Utilisateur	Web application
1. L'utilisateur accède à l'application web via un url fourni auparavant	2. Une page d'authentification s'affiche à l'écran
3. L'utilisateur rentre ses credentials	
	4. Si le login et le mot de passe sont corrects alors l'utilisateur est redirigé vers la page d'accueil de l'application
	4. Si le login et le mot de passe ne sont pas corrects alors la page d'authentification est rafraîchie avec les données erronées
	5. Si l'utilisateur se trouve sur n'importe quelle page de l'application et qu'il veut se déconnecter, il lui suffit d'appuyer sur l'icône en bas de page.
	6. Redirection vers l'écran de login
7. Si l'utilisateur a oublié son mot de passe ou son login alors il peut appuyer sur « forgot password »	8. Un courrier électronique de réinitialisation de mot de passe est envoyé à l'adresse mail de l'utilisateur

Le diagramme de séquence suivant représente une interaction logique entre l'application web et l'utilisateur lorsque celui-ci veut s'authentifier.



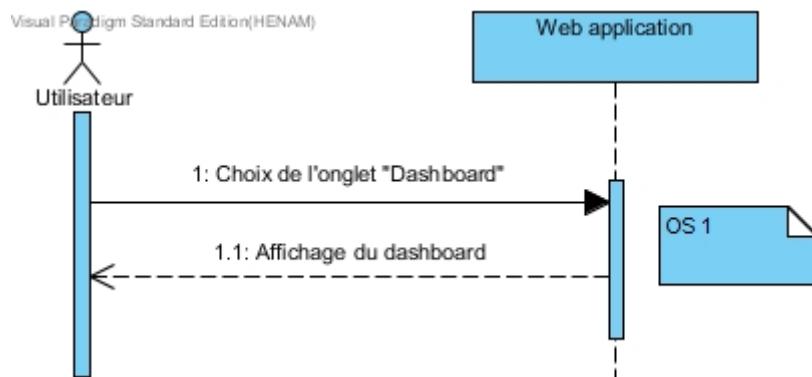
7.3.2 Affichage dashboard

L'utilisateur aura pour choix de sélectionner l'option de menu qui affiche son tableau de bord.

Pré conditions	
L'utilisateur doit s'être authentifié	

Scénario	
Utilisateur	Web application
1. Choisi l'option de menu « Dashboard »	2. Redirige l'utilisateur vers la page web qui correspond au tableau de bord

Le diagramme de séquence suivant représente une interaction logique entre l'application web et l'utilisateur lorsque celui-ci désire accéder à la page web contenant son tableau de bord.



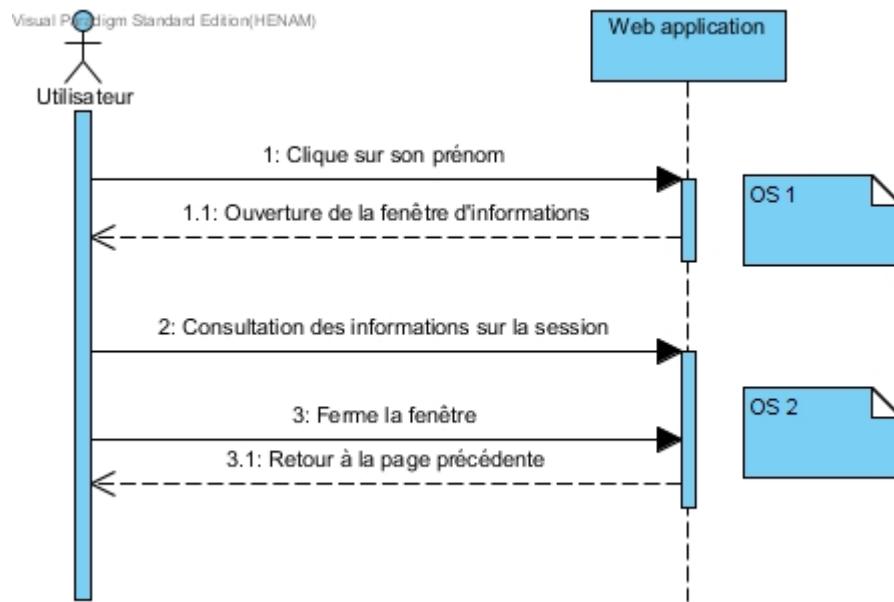
7.3.3 Consultation session

Lorsque l'utilisateur s'est authentifié correctement il aura la possibilité d'accéder aux informations stockées en base de données le concernant. Notamment son nom, prénom, e-mail, date de naissance, adresse, fonction dans l'entreprise, rôles détenus, etc.

Pré conditions	
L'utilisateur doit s'être authentifié	

Scénario	
Utilisateur	Web application
1. L'utilisateur clique sur son nom / prénom	2. Ouverture de la fenêtre contenant toutes les informations concernant l'utilisateur de la session
3. Après avoir pris connaissance des informations, l'utilisateur peut fermer cette fenêtre	
	4. Retour à la page précédemment visitée

Le diagramme de séquence suivant représente une interaction logique entre l'application web et l'utilisateur lorsque celui-ci désire consulter les informations le concernant.



7.3.4 Sélection des giplets à ajouter

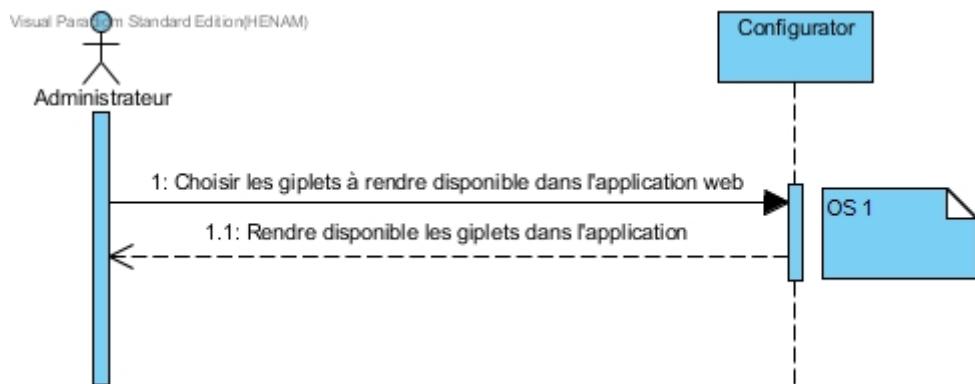
Pré conditions

Les giplets sont disponibles dans des librairies java
L'utilisateur doit s'être authentifié

Scénario

Administrateur	Configurator
1. Choisir les giplets à rendre disponibles dans l'application web	
	2. Les giplets sont disponibles dans l'application

Le diagramme de séquence suivant représente une interaction logique entre l'application web et l'administrateur lorsque celui-ci désire rendre disponible des giplets dans l'application web.



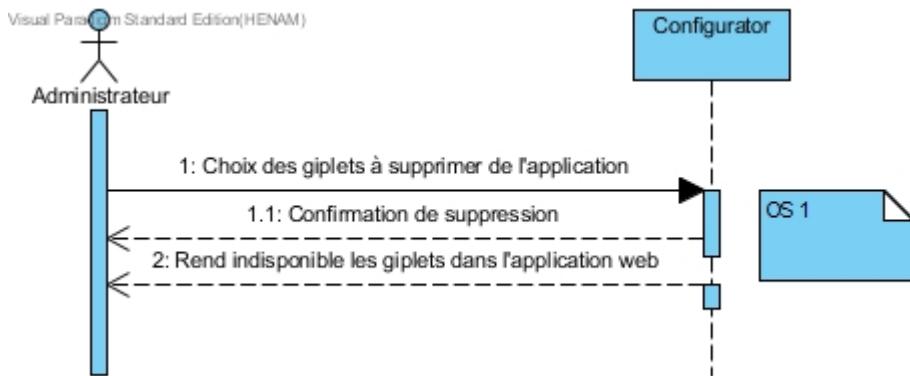
7.3.5 Suppression giplets (admin)

Pré conditions

- Le giplet doit être présent dans la configuration de l'application (il ne peut pas supprimer un giplet qui n'est pas disponible)
- L'administrateur doit s'être authentifié avant d'accéder au configurator

Scénario	
Administrateur	Configurator
1. L'administrateur choisit les giplets qu'il désire ne plus rendre disponible dans l'application web	
2. Confirme son choix	3. Confirmation de suppression
	Les giplets supprimés ne seront désormais plus disponible à ajouter dans l'application web via la page web « giplets »

Le diagramme de séquence suivant représente une interaction logique entre l'application web et l'administrateur lorsque celui-ci désire rendre indisponible des giplets dans l'application web.



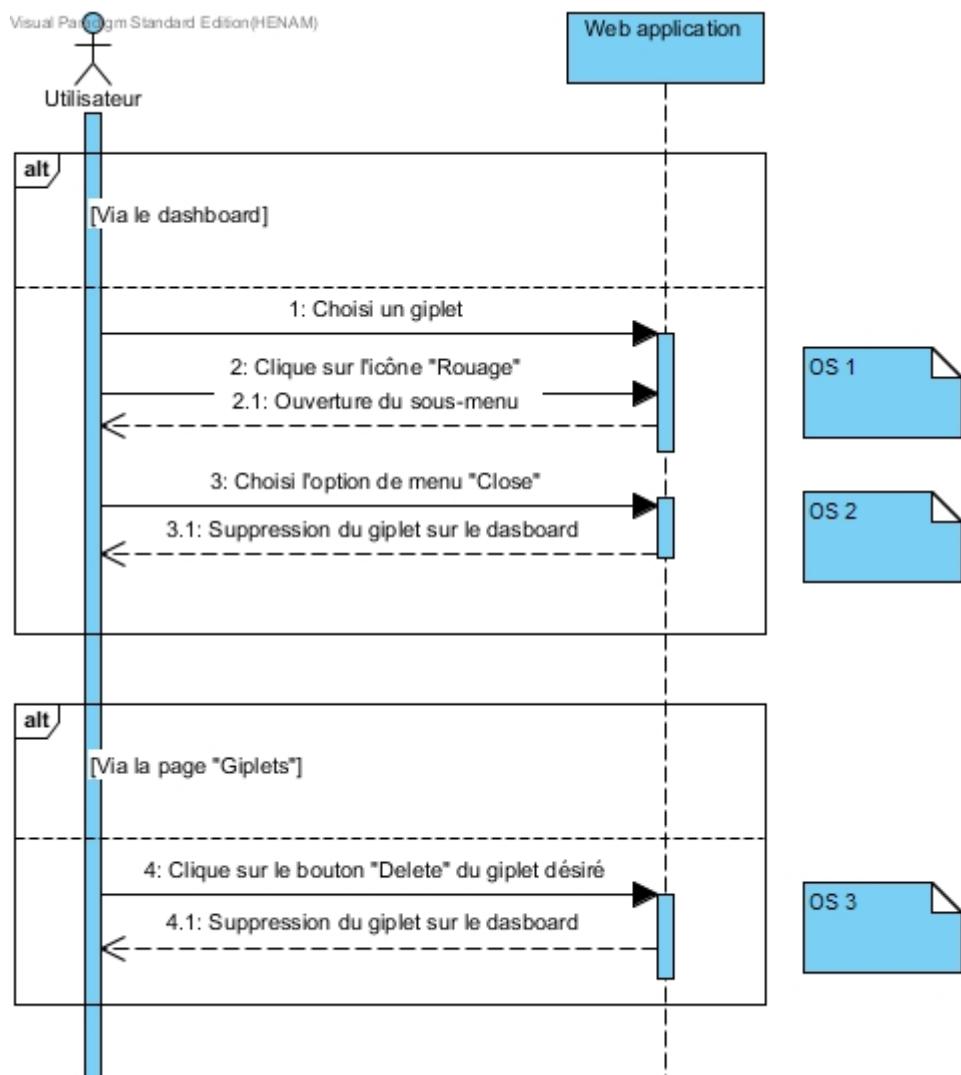
7.3.6 Suppression giplets (client)

Pré conditions

- Le giplet doit être présent sur le tableau de bord (il ne peut pas supprimer un giplet inexistant)
- L'utilisateur doit s'être authentifié

Scénario	
Utilisateur	Web application
1. Si l'utilisateur désire supprimer un giplet via le tableau de bord, il choisit le giplet qu'il veut supprimer, il clique sur le rouage et choisit l'option de menu « close »	2. Disparition du giplet sur le tableau de bord
3. Si l'utilisateur désire supprimer un giplet via la page « giplets », il choisit le giplet qu'il veut supprimer de son tableau de bord et clique sur « delete »	4. Disparition du giplet sur le tableau de bord

Le diagramme de séquence suivant représente une interaction logique entre l'application web et l'utilisateur lorsque celui-ci désire supprimer des giplets de son tableau de bord.

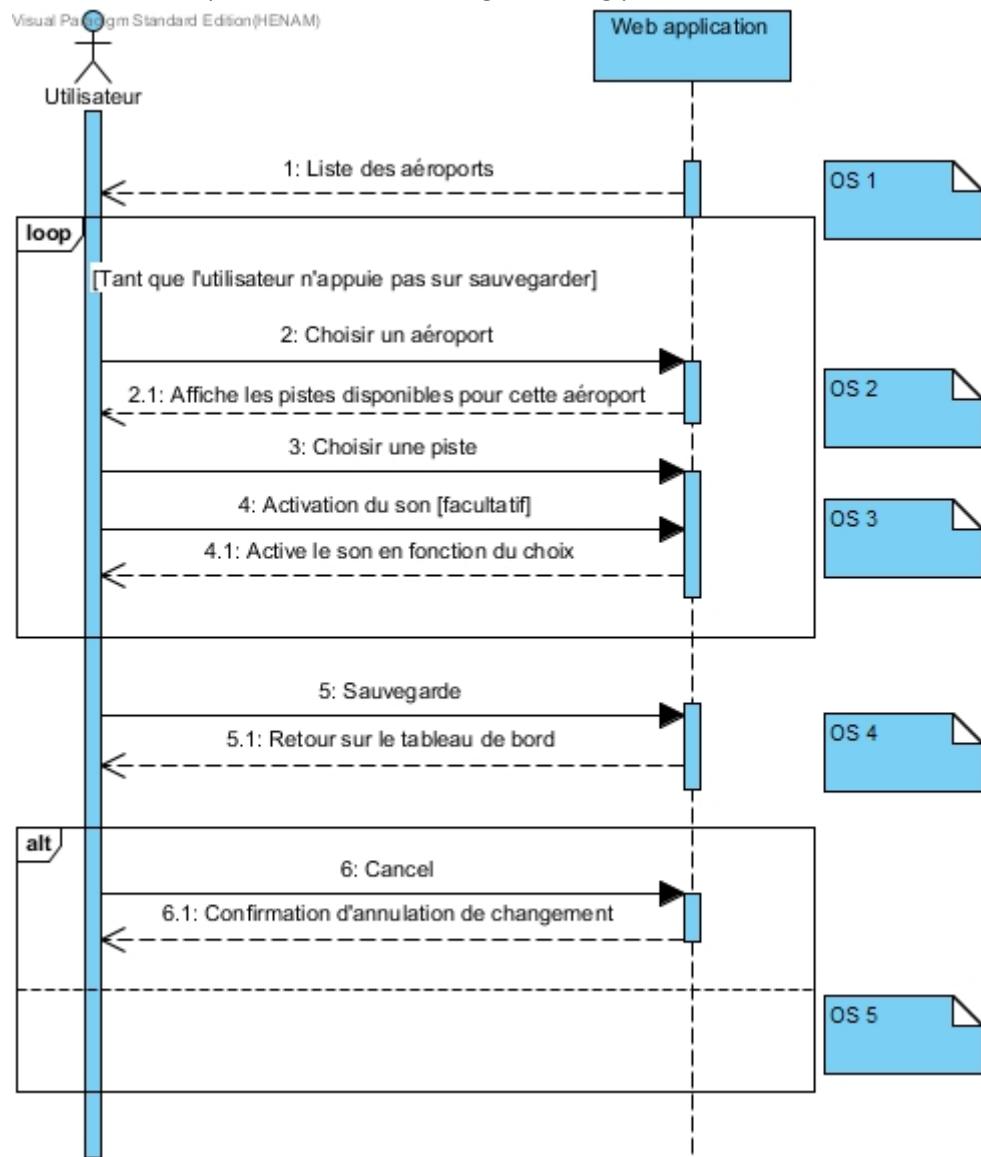


7.3.7 Configuration giplet « beacon »

L'utilisateur à la possibilité de configurer le giplet beacon directement depuis le tableau de bord.

Pré conditions	
Utilisateur	Scénario
	Web application
	<ul style="list-style-type: none"> • L'utilisateur doit s'être authentifié • Le giplet doit être présent sur le tableau de bord
2. Tant que l'utilisateur n'a pas appuyé sur le bouton sauvegarder	1. L'application web propose une liste de tous les aéroports disponibles
Il choisit l'aéroport souhaité	Affichage des pistes disponibles pour cet aéroport
Il choisit la piste	
Il active ou désactive le son des beacons	
	3. Une fois que l'utilisateur a appuyé sur le bouton sauvegarder, l'application enregistre les modifications et redirige l'utilisateur vers le tableau de bord
4. Si l'utilisateur appuie sur le bouton cancel	5. Fenêtre de dialogue pour confirmer l'annulation des modifications

Le diagramme de séquence suivant représente une interaction logique entre l'application web et l'utilisateur lorsque celui-ci désire configurer son giplet « beacon ».



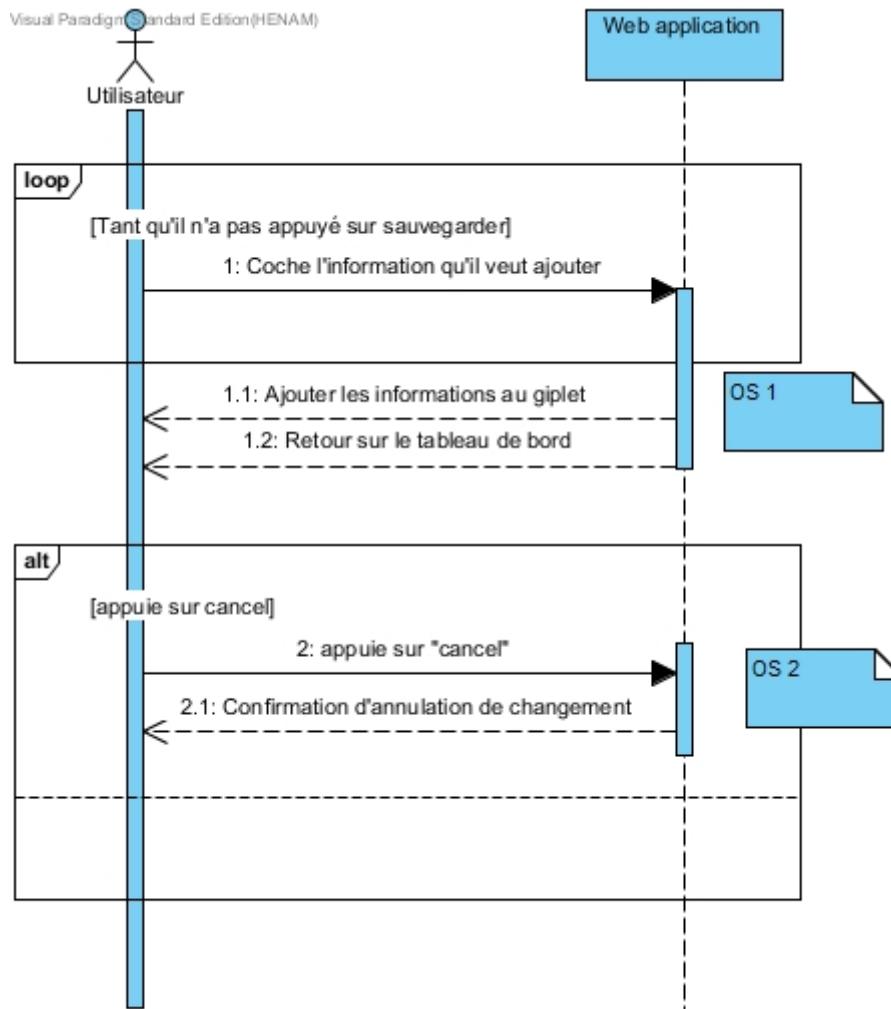
7.3.8 Configuration giplet « Flight information »

L'utilisateur à la possibilité de configurer son giplet « Flight information » directement depuis le tableau de bord.

Pré conditions	
<ul style="list-style-type: none">• L'utilisateur doit s'être authentifié• Le giplet doit être présent sur le tableau de bord• L'utilisateur doit se trouver sur la page de configuration du giplet	

Scénario	
Utilisateur	Web application
1. Tant que l'utilisateur n'a pas appuyer sur sauvegarder	
Il choisit les informations qu'il désire obtenir dans son giplet	
	2. Une fois que l'utilisateur à sauvegarder, ajout des informations au giplet
	3. Fermeture de la configuration et retour sur le tableau de bord
4. Si l'utilisateur appuie sur le bouton cancel	
	5. Confirmation d'abandon des modifications
	6. Si l'utilisateur confirme alors il est redirigé vers le tableau de bord sans modifications du giplet
	6. Sinon il reste sur la configuration du giplet

Le diagramme de séquence suivant représente une interaction logique entre l'application web et l'utilisateur lorsque celui-ci désire configurer le giplet « Flight information ».



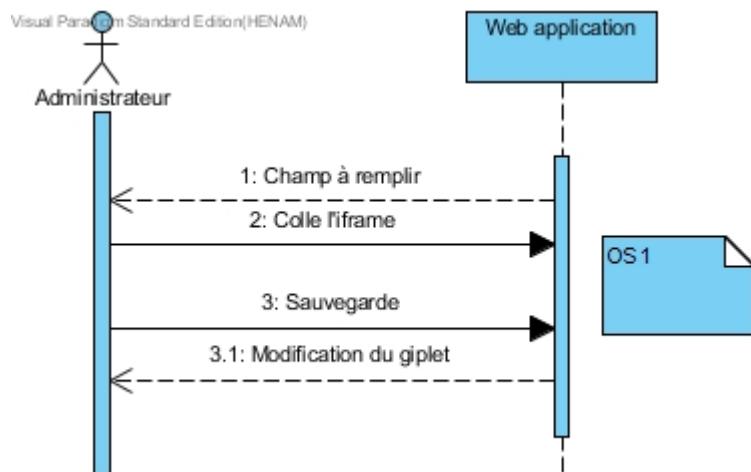
7.3.9 Configuration giplet « IFrame »

Seul un administrateur à la possibilité de configurer son giplet « IFrame » directement depuis le tableau de bord.

Pré conditions	
<ul style="list-style-type: none"> L'utilisateur doit s'être authentifié en tant qu'administrateur Le giplet doit être présent sur le tableau de bord Il doit se trouver sur la page de configuration du giplet 	

Scénario	
Administrateur	Web application
2. Il choisit un lien « iframe » et l'insère	1. Champ libre à remplir
	3. Après la sauvegarde par l'utilisateur, la fenêtre se ferme et le giplet est mis à jour

Le diagramme de séquence suivant représente une interaction logique entre l'application web et l'administrateur lorsque celui-ci désire configurer le giplet « IFrame».



7.3.10 Configuration giplet « charts »

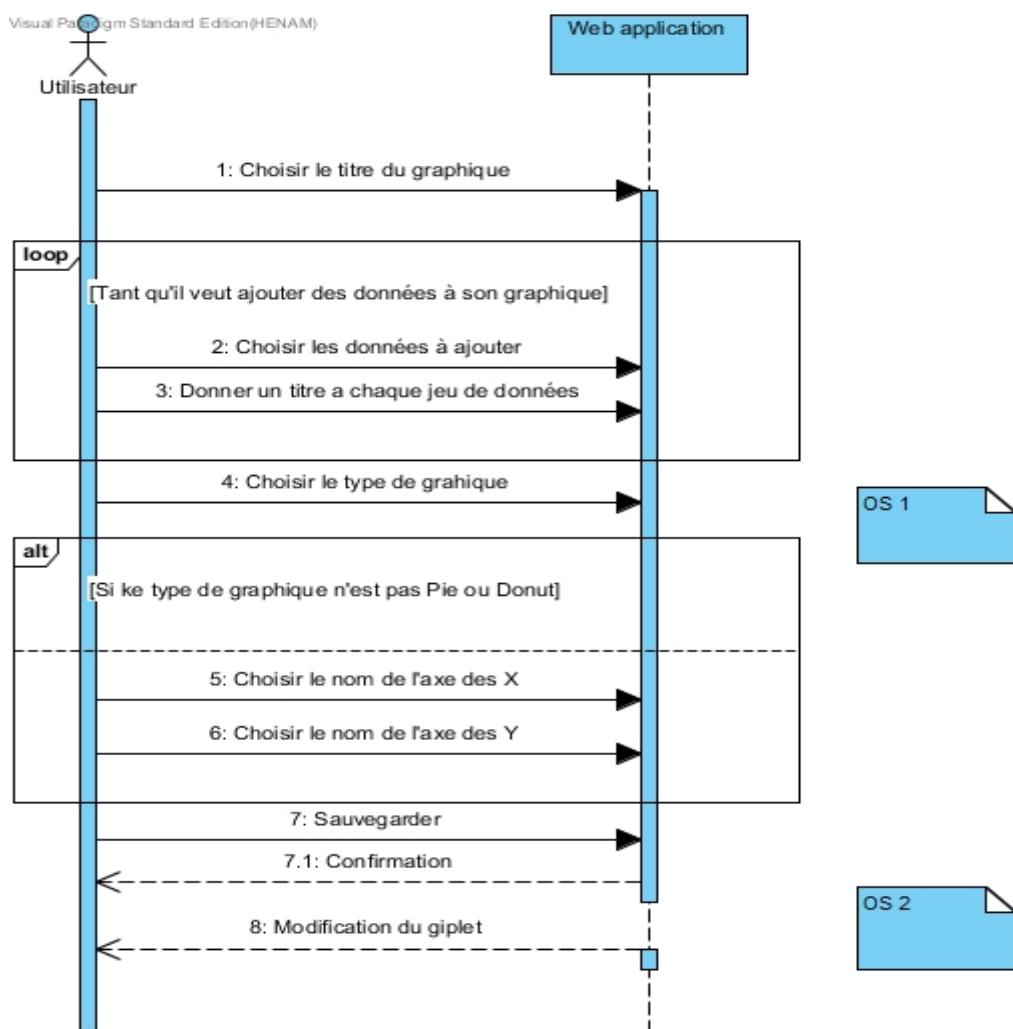
L'utilisateur à la possibilité de configurer son giplet graphique différemment selon la caractéristique du graphique (real time ou reporting).

Pré conditions

- Les données disponibles dans le graphique proviennent d'une base de données ou d'un canal publish-subscribe
- L'utilisateur doit s'être authentifié
- L'utilisateur doit se trouver dans la fenêtre de configuration du giplet
- La source de données doit être pré enregistrée

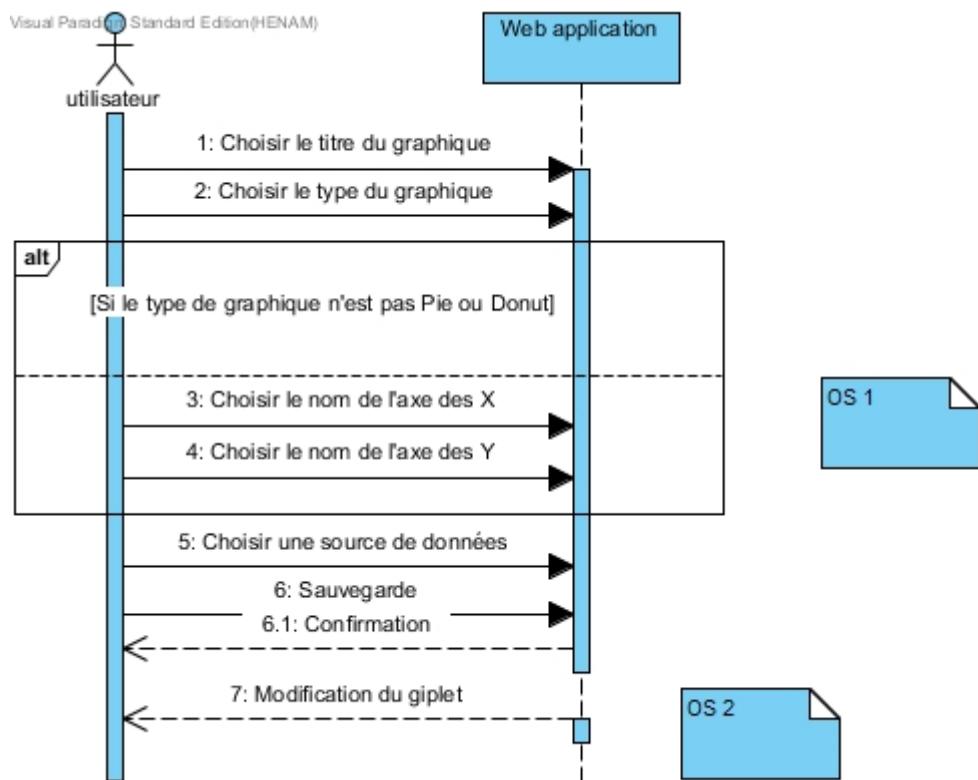
Dans le cas d'un graphique « reporting »

Scénario	
Utilisateur	Application web
1. L'utilisateur choisit le titre du graphique	
Tant que l'utilisateur veut ajouter des données	
2. Il choisit les données à ajouter	
3. Il donne un nom à son jeu de données	
4. Il choisit le type de graphique	
Si le type de graphique n'est pas Tarte ou donut	
5. Il choisit le nom de l'axe des X	
6. Il choisit le nom de l'axe des Y	
7. Sauvegarder ses modifications	
	7.1 Confirmation
	8. Modification du giplet sur le tableau de bord



Dans le cas d'un graphique « real-time »

Scénario	
Utilisateur	Web application
1. L'utilisateur choisit le titre du graphique	
2. Il choisit le type de graphique	
Si le type de graphique choisi n'est pas Tarte ou Donut	
3. Choisir le nom de l'axe des X	
4. Choisir le nom de l'axe des Y	
5. Choisir une source de données pour les graphiques	
6. L'utilisateur sauvegarde ses modifications	
	6.1 Confirmation de sauvegarde
	7. Modification du giplet sur le tableau de bord



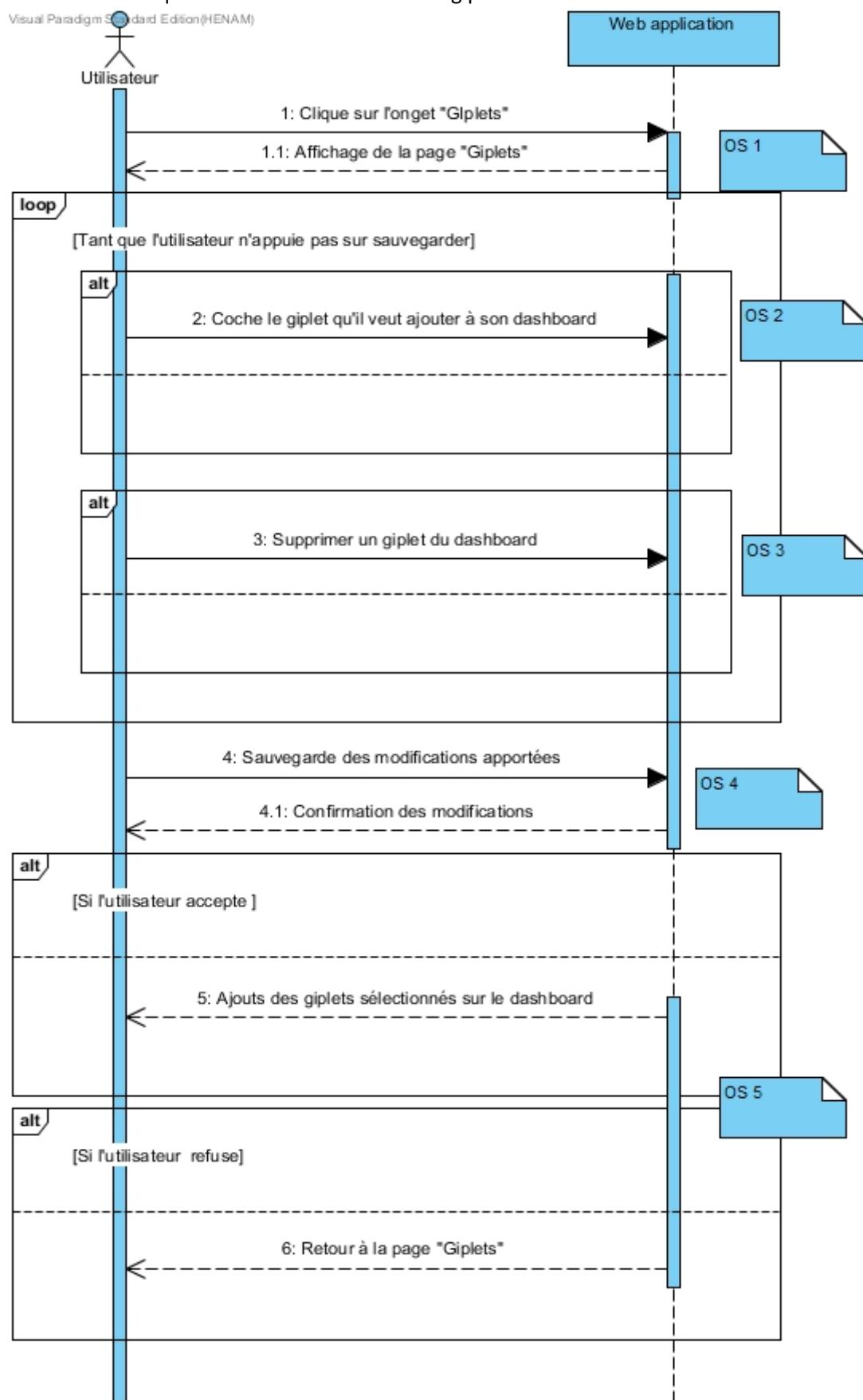
7.3.11 Ajouter des giplets au dashboard

Pré conditions

Les giplets disponible à l'ajout doivent être au préalable ajouté dans la configuration de l'application par un administrateur

Scénario	
Utilisateur	Web application
1. Clique sur l'option de menu « Giplet »	2. Ouverture de la page web « Giplet »
3. Tant que l'utilisateur n'appuie pas sur le bouton sauvegarder	
Soit il ajoute un giplet à son tableau de bord	
Soit il supprime un giplet de son tableau de bord	
	4. Demande de confirmation de sauvegarde
	5. Si l'utilisateur a accepté les modifications alors prise en compte des modifications apportées
	6. Si l'utilisateur n'a pas accepté les modifications alors il est redirigé vers la page des giplets.

Le diagramme de séquence suivant représente une interaction logique entre l'application web et l'utilisateur lorsque celui-ci désire afficher des giplets dans son tableau de bord.

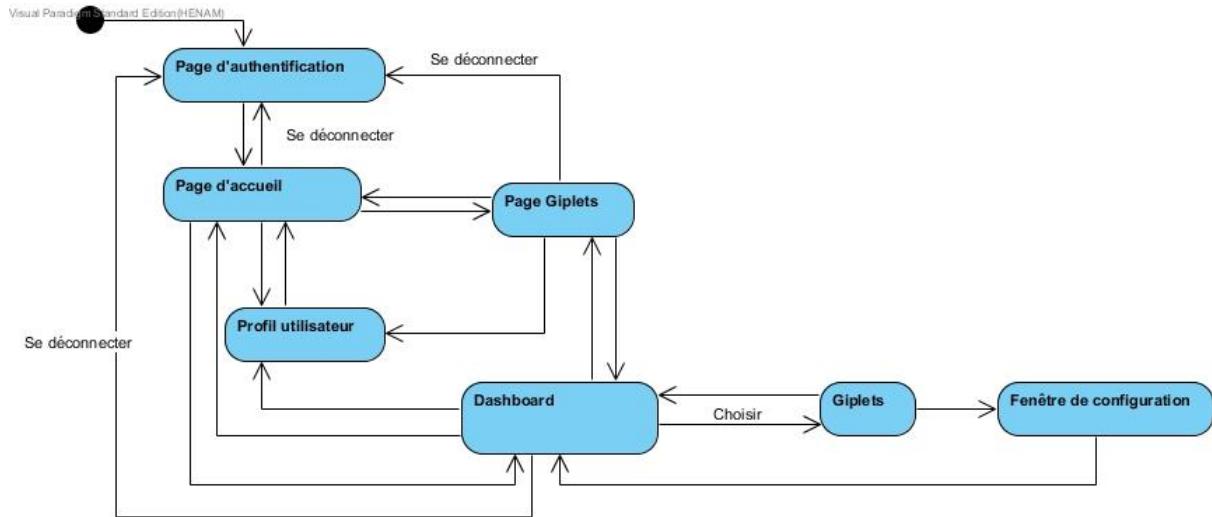


7.3.12 Profil utilisateur

Administrateur	Client
Description	
Personne qui possède plus de droits qu'un client (Consultant, directeur parc informatique, etc).	
Droits	
Configuration du giplet « IFrame »	Configuration du giplet « Flight » Configuration du giplet « Beacon » Configuration du giplet « Charts »

7.4 Diagramme de navigation

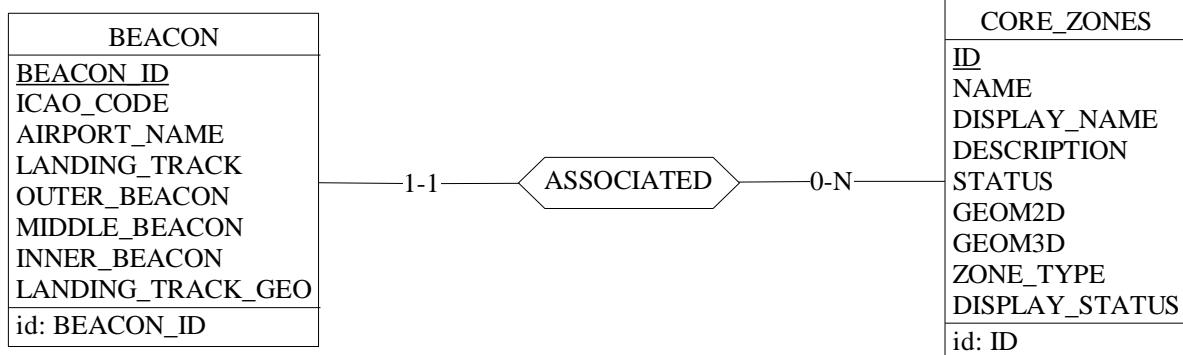
Le diagramme de navigation permet de connaître l'enchaînement des écrans utilisateurs.



7.5 Conception

7.5.1 Base de données

Les giplets « IFrame » et « flight information » n'ont pas eu besoin d'une implémentation en base de données contrairement au giplet « beacon ».



BEACON		
Définition	Description	Attribut
Définition	Ensemble des beacons existants	
Attributs	Définition	Domaine
BEACON_ID	Identifiants unique d'un beacon	Number (10, 0) AI
ICAO_CODE	icao code de l'aéroport	Varchar2 (10)
AIRPORT_NAME	Nom de l'aéroport	Varchar2 (50)
LANDING_TRACK	Nom de la piste d'atterrissement	Varchar2 (15)
OUTER_BEACON	Clé étrangère vers le nom d'une zone	Varchar2 (80)
MIDDLE_BEACON	Clé étrangère vers le nom d'une zone	Varchar2 (80)
INNER_BEACON	Clé étrangère vers le nom d'une zone	Varchar2 (80)
LANDING_TRACK_GEO	Forme géométrique 2D de la piste d'atterrissement	SDO_GEOMETRY

CORE_ZONES		
Définition	Définition	Domaine
ID	Identifiants unique d'une zone	Number AI
NAME	Nom de la zone	Varchar2 (80) unique
DISPLAY_NAME	Nom affichable de la zone	Varchar2 (80)
DESCRIPTION	Description de la zone	Varchar2 (200)
STATUS	Actif ou inactif	Varchar2 (20)
GEOM2D	Représentation 2D de la zone	SDO_GEOOMETRY
GEOM3D	Représentation 3D de la zone	SDO_GEOOMETRY
ZONE_TYPE	Type de zone	Varchar2 (80)
DISPLAY_STATUS	Status affichable de la zone	Varchar2 (80)

Outer beacon, middle beacon et inner beacon sont unique pour une piste d'un aéroport et pour un aéroport. Il est impossible de retrouver 2 beacons identique.

7.6 Interface homme-machine

Les écrans utilisateur suivant sont le résultat d'une analyse effectuée précédemment. Chaque écran utilisateur correspond à un cas d'utilisation décrit dans « Chapitre 6 : Analyse ».

7.6.1 Authentification

L'authentification n'est pas encore disponible dans l'application actuelle, les 2 écrans suivants relèvent d'une analyse et n'ont pas été implémentés durant le stage.

« Page de login » Illustre la page de login.

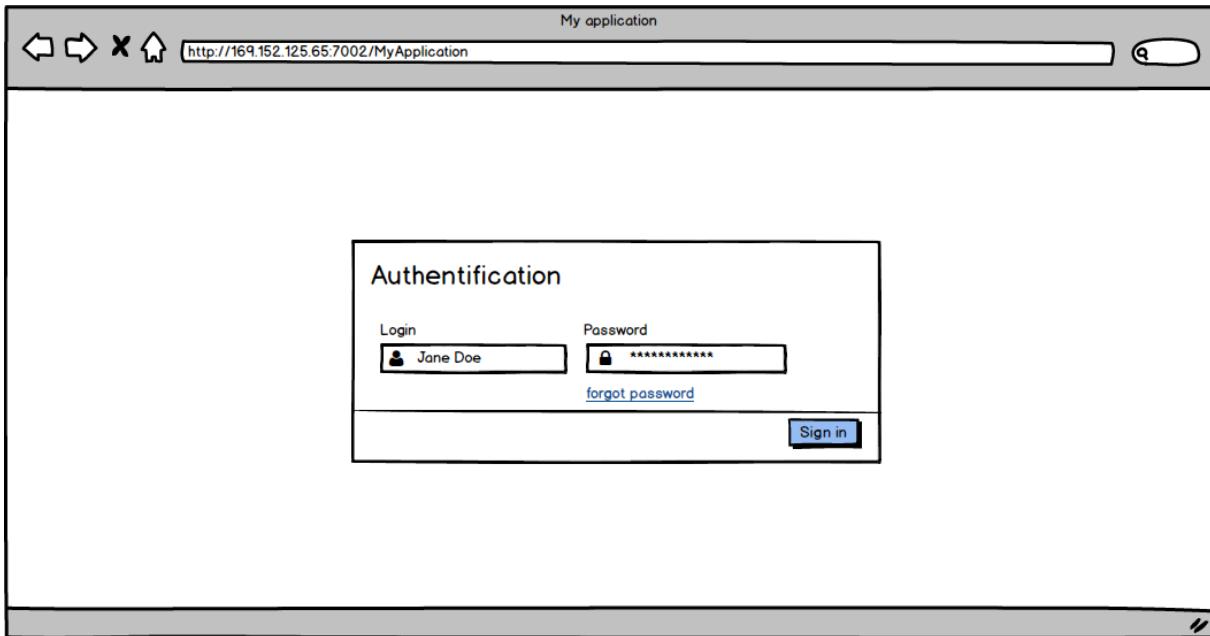


Figure 22 Page de login

« Login / Password failed » Illustre la page de login si l'utilisateur rentre un mauvais login ou un mauvais mot de passe.

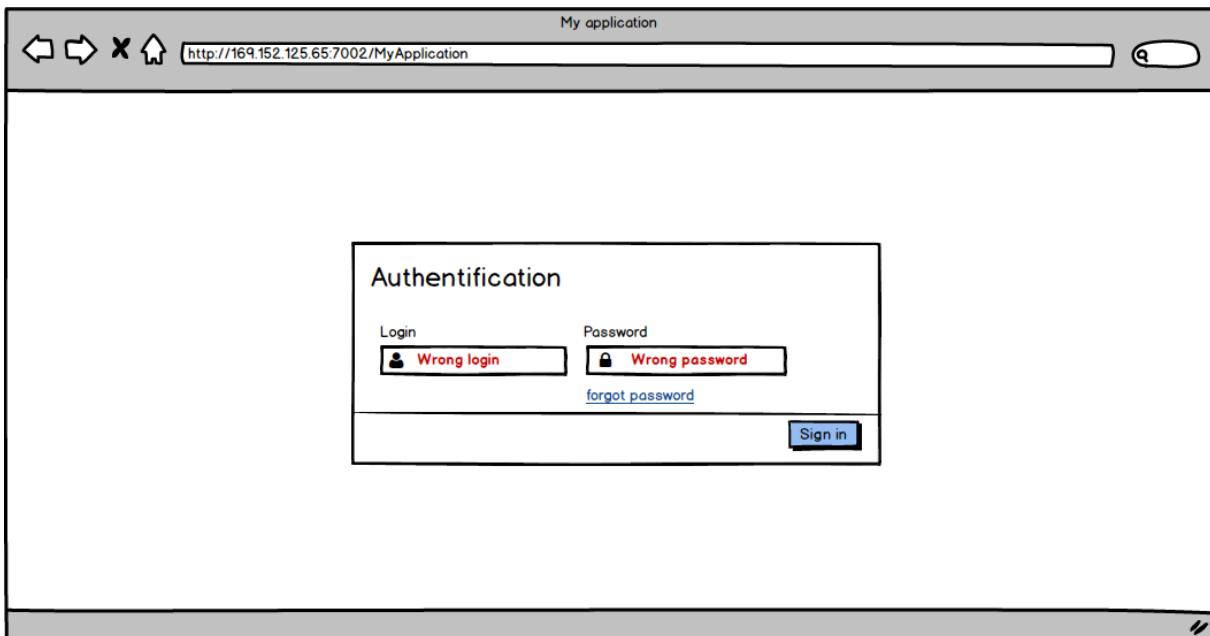


Figure 23 Login / Password failed

« Page d'accueil » Illustre la page d'accueil après que l'utilisateur a réussi à s'authentifier.

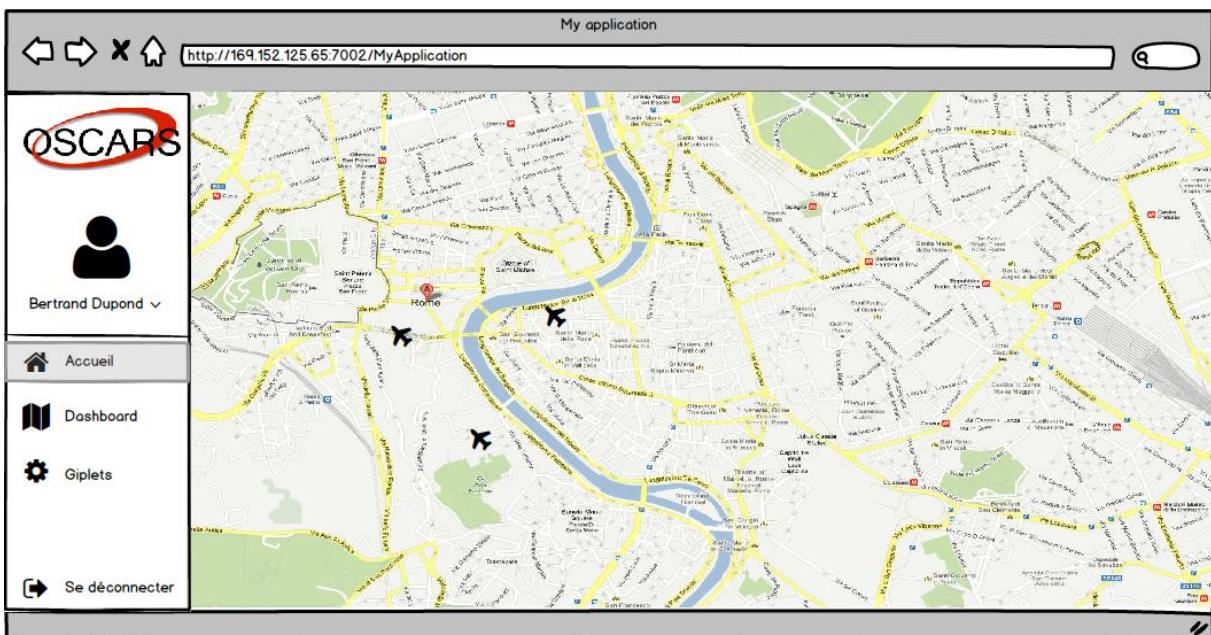
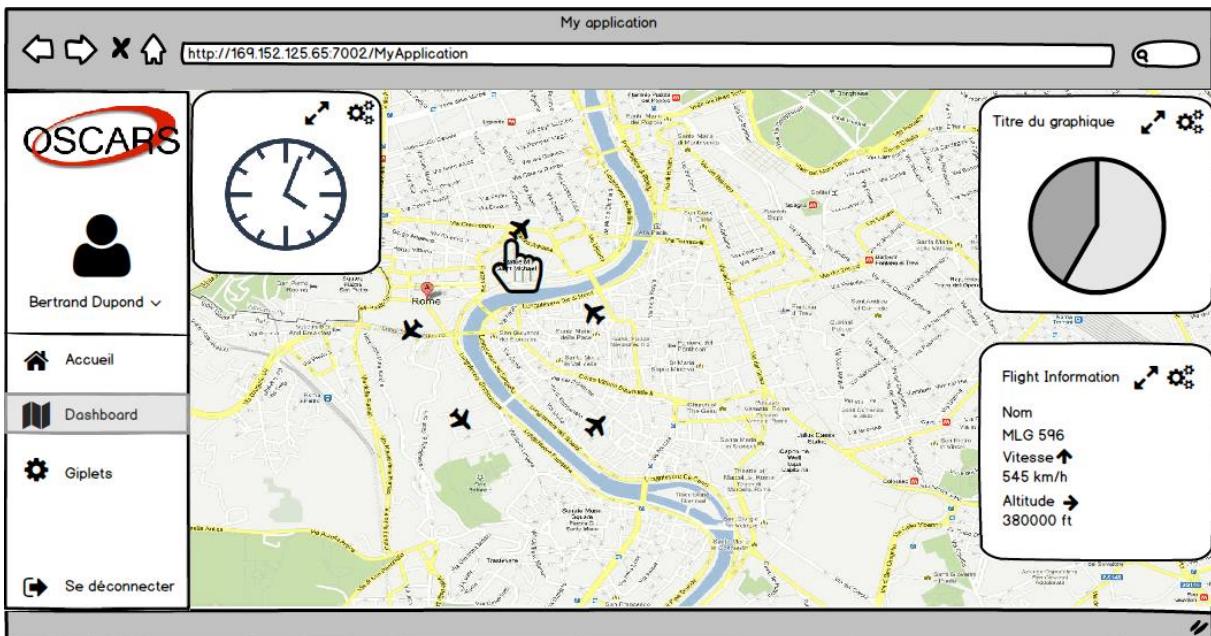


Figure 24 Page d'accueil

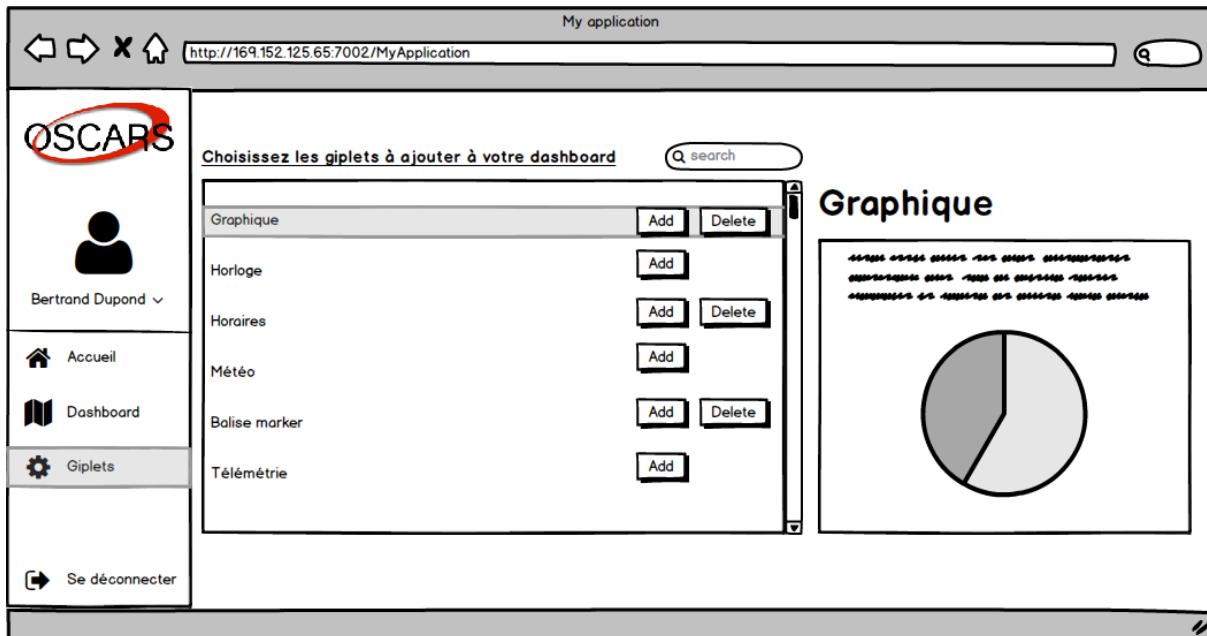
7.6.2 Affichage dashboard

« Page du tableau de bord » Illustre le tableau de bord lorsque le client à cliquer sur l'option de menu « Dashboard ».



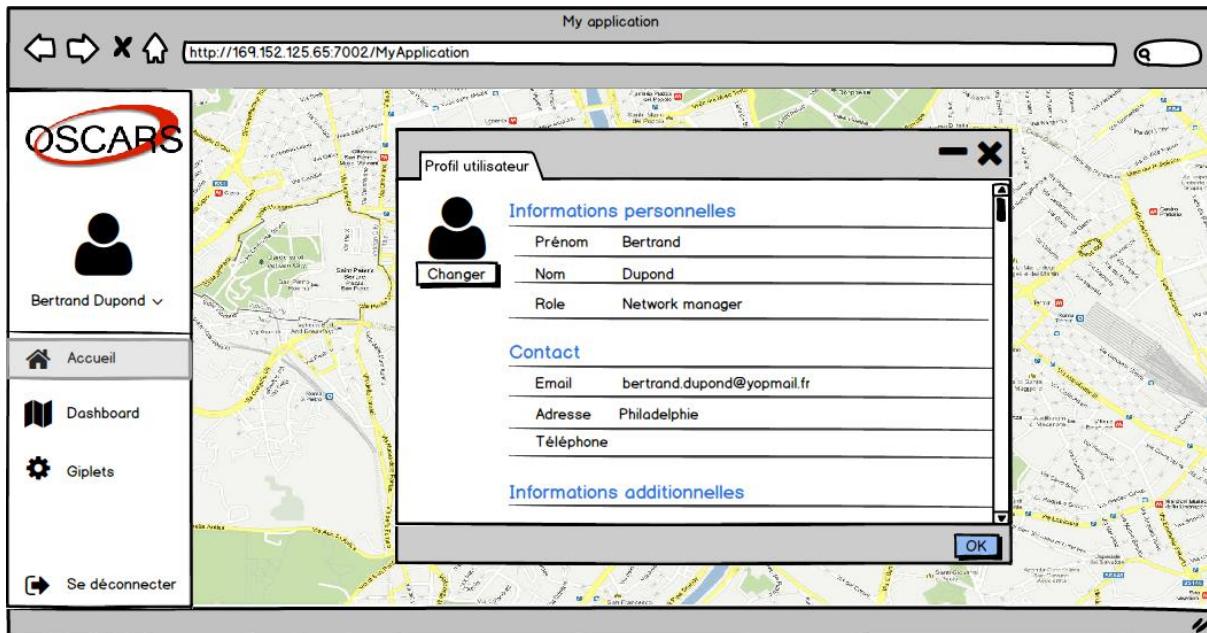
7.6.3 Ajouter des giplets au dashboard

« Page giplets » Illustre la page où l'utilisateur à la possibilité d'ajouter ou de supprimer des giplets sur son tableau de bord. Cet écran n'a pas été implémenté.



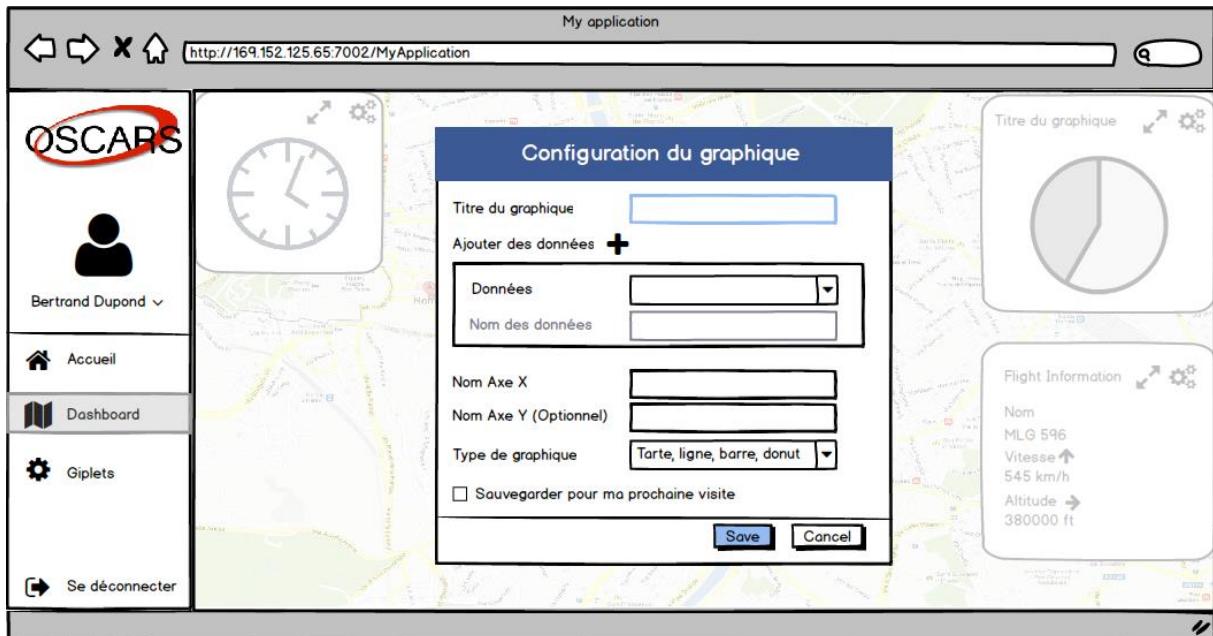
7.6.4 Consultation de session

« Fenêtre du profil utilisateur » représente la fenêtre qui rassemble toute l'information concernant l'utilisateur actuel de l'application. Cette fonctionnalité n'existe pas encore dans l'application actuelle.

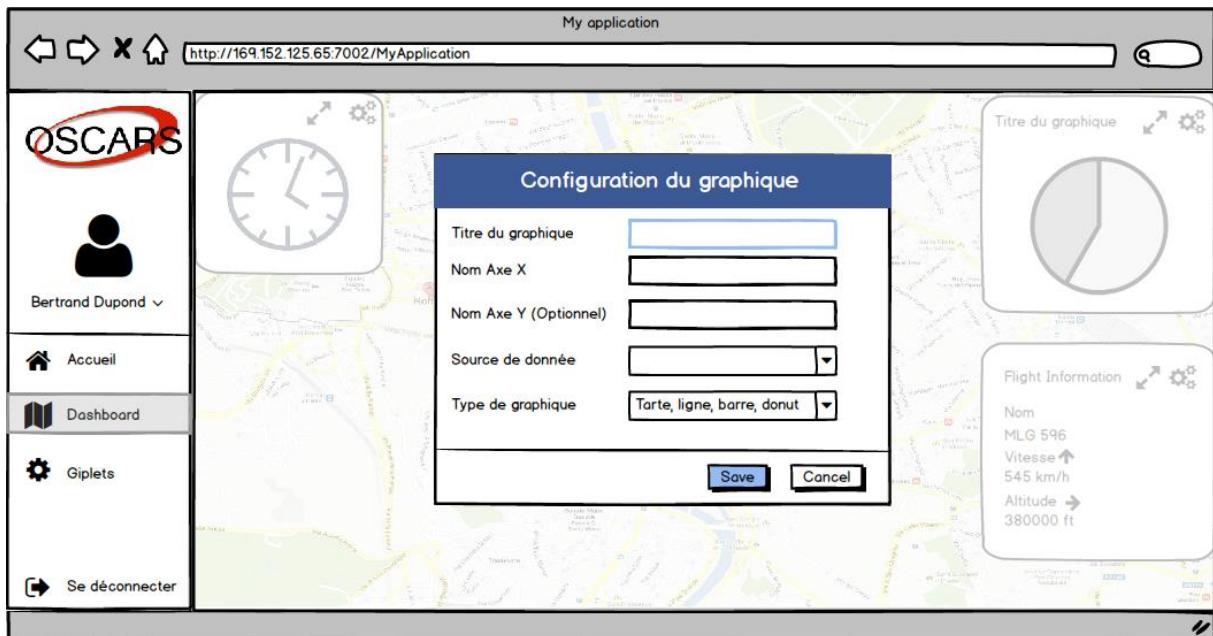


7.6.5 Configuration giplet graphique

« Configuration d'un giplet graphique » Illustre la fenêtre qui permettra à l'utilisateur de paramétriser son giplet graphique lorsqu'il aura cliqué sur le rouage de celui-ci.

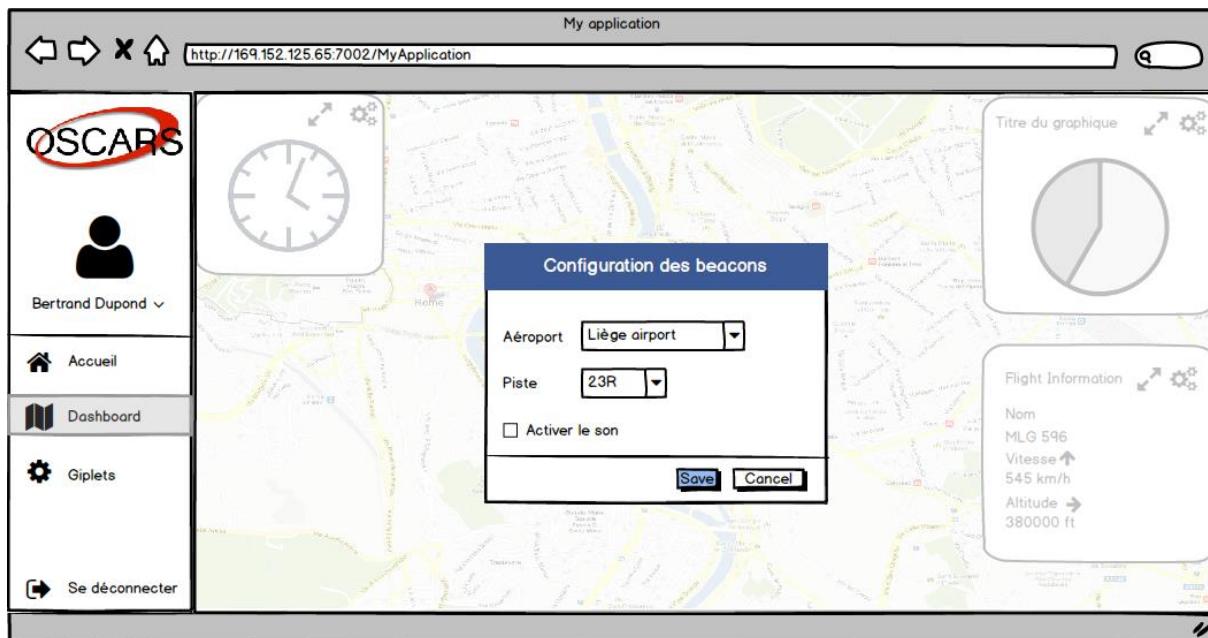


« Configuration du giplet graphique real-time » Illustre la configuration disponible pour un graphique dit « real-time ».

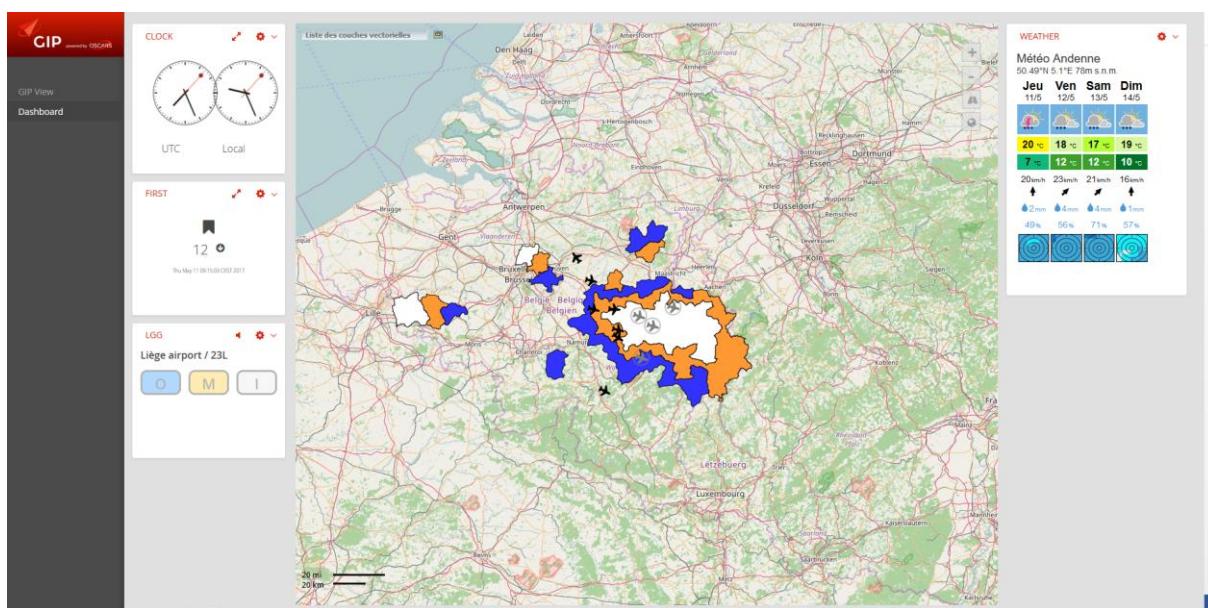


7.6.6 Configuration giplet « beacon »

« Configuration d'un beacon marker » Illustre la fenêtre qui s'ouvrira lors du clic adéquat de la part de l'utilisateur et qui permettra de configurer les beacon marker.



« Giplet beacon marker » Illustre le giplet des beacons marker insérer dans un dashboard.



7.7 Architecture des giplets

L'architecture des giplets se concentre principalement sur le fait que celles-ci peuvent être modulable. En effet, le code java est compilé en librairie jar pour être utilisé dans différentes applications.

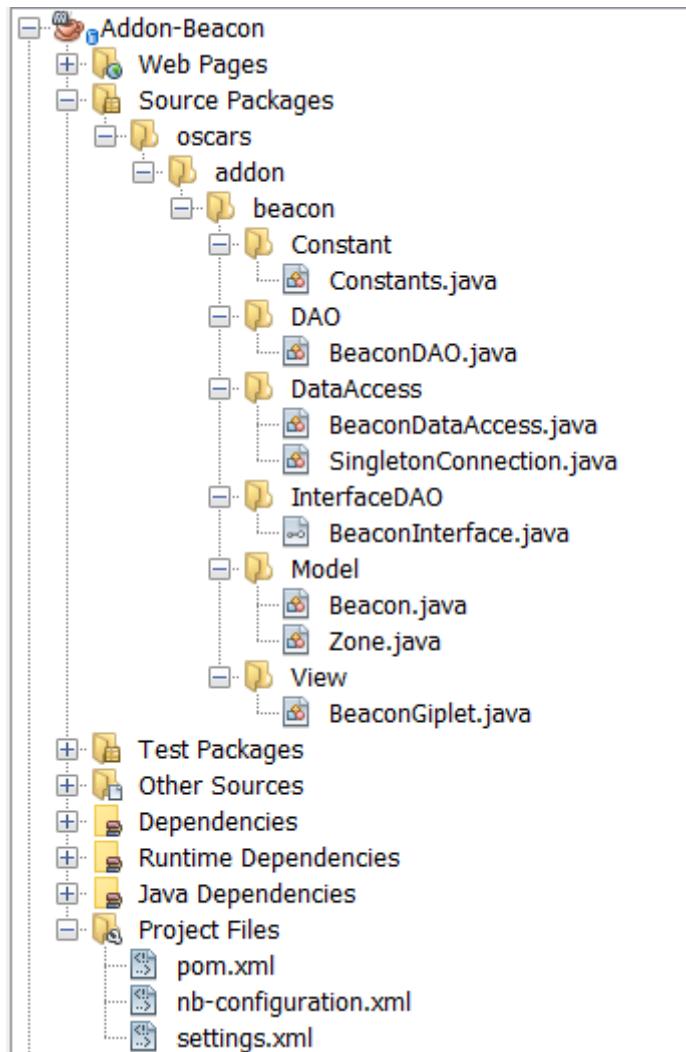
Les giplets « weather » et « flight information » recevront des données de l'application Vaadin sous forme de JSON, ceux-ci effectueront un parsage et afficheront les données.

7.7.1 Giplet beacon

Le giplet beacon se base sur le patron d'architecture « DAO », en effet, étant donné que l'application doit se connecter à une base de données, il est plus intelligent de séparer la partie persistance des données de la partie logique.

L'image ci-dessous illustre l'application correspondant au giplet organisé en package respectant le patron d'architecture « DAO ».

Pour se connecter à la base de données, c'est la classe « PreparedStatement » qui a été utilisée pour plus de simplicité et empêcher la multi persistance xml.



8 Implémentation

Ce chapitre explique plus en détails le fonctionnement et les technologies utilisées dans Vaadin et référencées au chapitre 4.

8.1 Vaadin

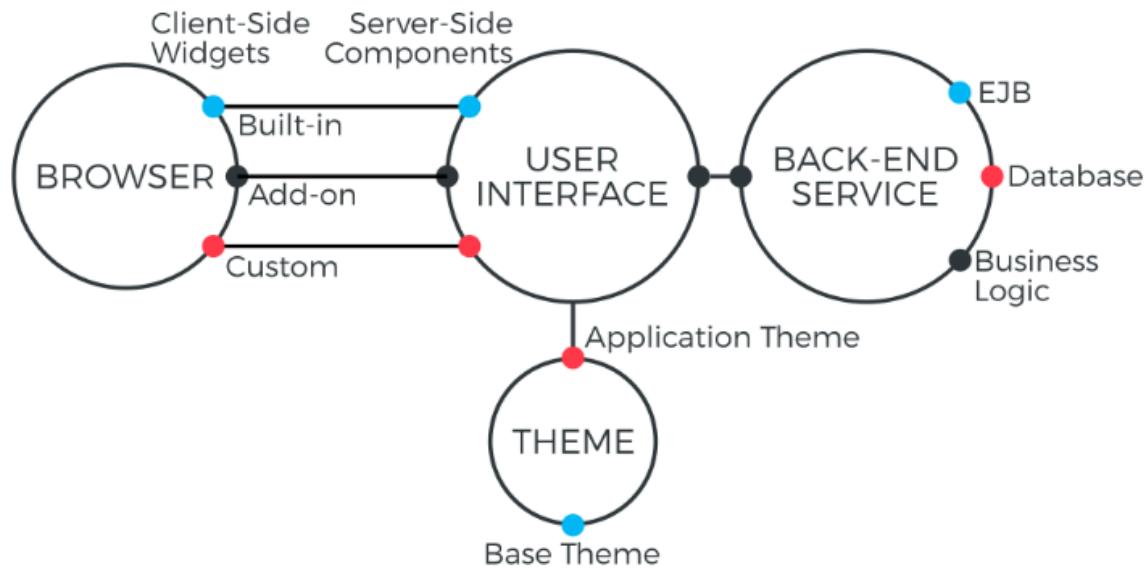


Figure 17 Architecture basique du framework Vaadin

Vaadin est framework relativement complet et modulable, effectivement, les éléments Vaadin reste indépendants entre eux, la partie logique / serveur de l'application est séparé de la partie client / browser, de la partie back-end ainsi que de la partie rendu graphique comme le montre le schéma ci-dessus.

Le code Java de l'application tourne sur le client -navigateur web – comme du code Javascript, celui-ci permet l'affichage de l'interface graphique et les interactions avec l'utilisateur. La partie logique de l'application tourne sur des servlets Java. L'avantage de Vaadin est qu'il n'est pas nécessaire de posséder de plugins dans le navigateur car l'application a été transformé en code Javascript (grâce à Google Web Toolkit).

GWT est un outil qui permet de compiler du code Java en code Javascript lisible par le navigateur web. (cf chapitre 3)

Grâce à Vaadin, le développeur n'a plus à se soucier de la partie client et de se concentrer d'avantage sur le coté business logic de l'application.

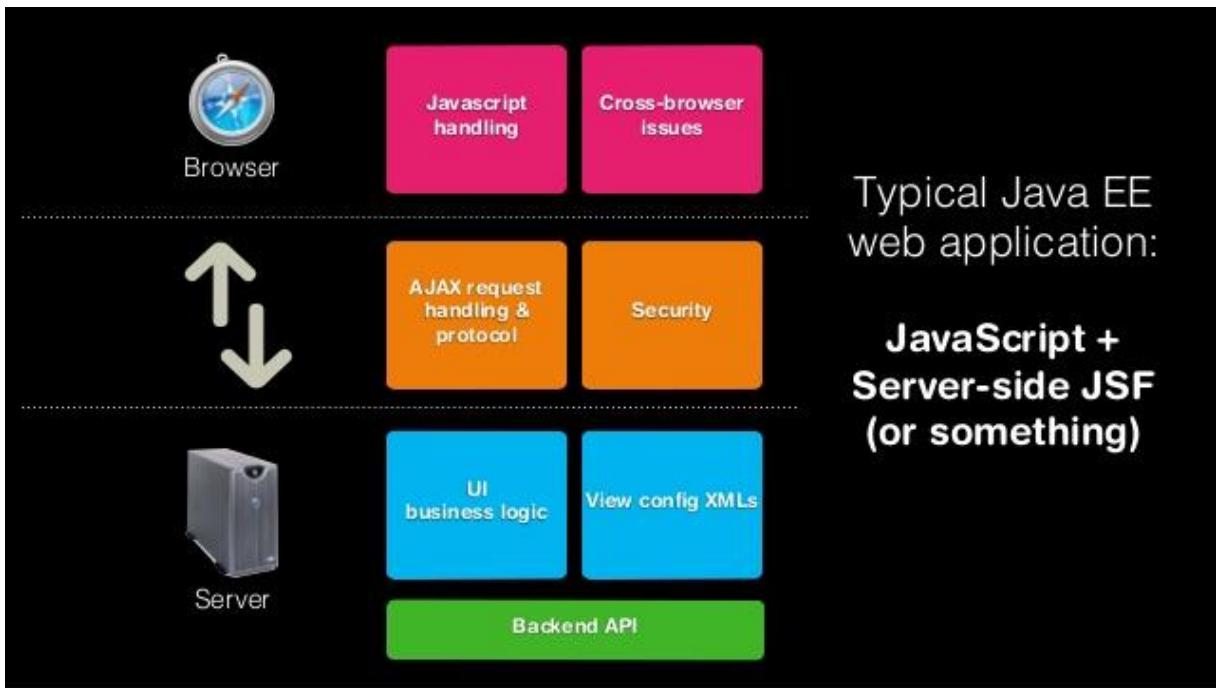


Figure 20 Schéma d'une application Java EE

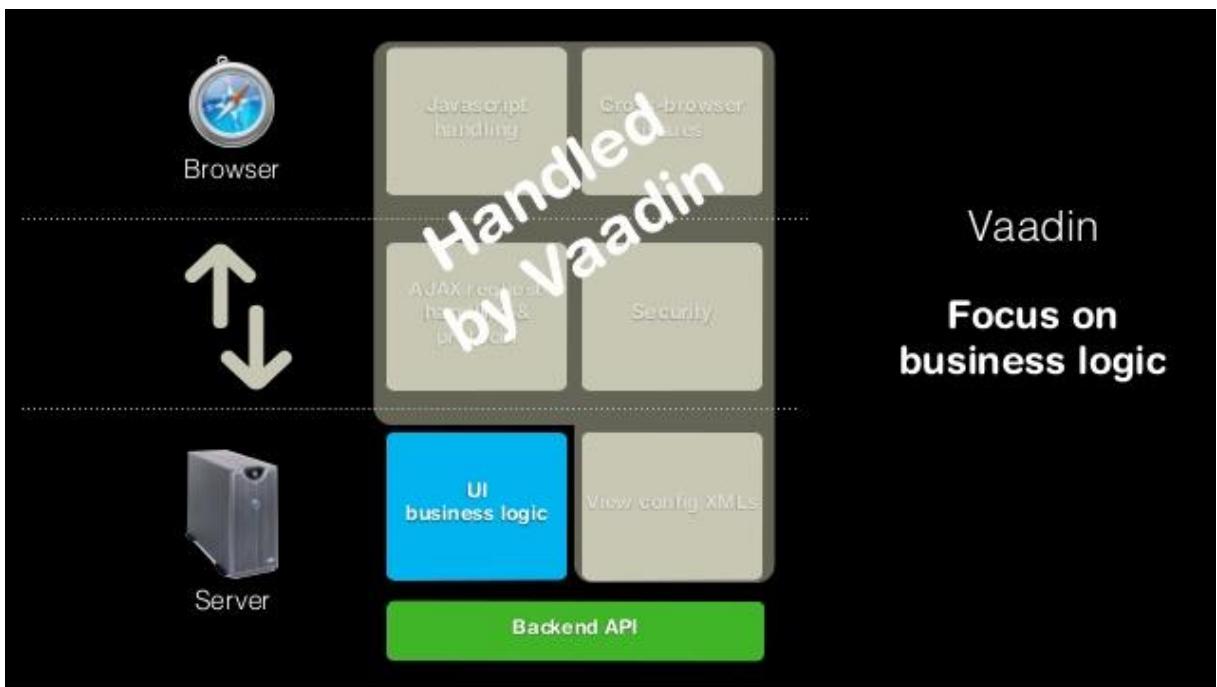


Figure 21 Schéma d'une application Java EE avec le framework Vaadin

Les images ci-dessus illustre le contraste entre une application Java EE basique et une application Java EE avec le framework vaadin. GWT gère le coté sécurités et compatibilité (JavaScript), Vaadin gère , avec la meilleures des manières, le coté AJAX, sécurité et configuration et nous laisse donc la gestion de la business logic.

Le fait que les différents éléments de vaadin soit modulable nous permet de décider, comme bon nous semble, le style que l'on veut donner à l'application, le choix peut s'orienter vers la création de nouveau style, réutilisation de celui déjà existant ou bien ajouter à notre projet des thèmes incorporer dans des add-ons.

8.1.1 Injection de dépendance – Vaadin CDI

// « Learning Vaadin 7 » chapter 8 p.299 les méthodes et les annotations utilisées

L'inversion de contrôle est le principe que ce n'est plus au développeur d'instancier les objets mais à l'application de gérer les dépendances entre eux ce qui permet d'avoir un faible couplage entre les 2 objets concernés.

Pour utiliser l'injection de dépendance dans Vaadin il faut tout d'abord l'ajouter comme dépendance dans le pom.xml

```
<dependency>
    <groupId>com.vaadin</groupId>
    <artifactId>vaadin-cdi</artifactId>
    <version>[replaceable]LATEST</version>
</dependency>
<dependency>
    <groupId>javax.enterprise</groupId>
    <artifactId>cdi-api</artifactId>
    <version>[replaceable]1.2</version>
</dependency>
```

Pour notifier que l'application va utiliser l'injection de dépendance il faut annoter la classe qui étend UI comme illustré ci-dessous.

```
@CDIUI("")
@Theme("mytheme")
public class MyUI extends UI {
```

Pour injecter les objets il suffit d'ajouter l'annotation @Inject à la variable.

```
@Inject
private ClientDAO clientDAO;
```

L'injection de dépendance est très utile pour naviguer entre les différents pages de l'application, voici un exemple de code qui créer des pages et qui permet, via l'IoC, de naviguer entre les pages.

Le code suivant n'est pas extrait de l'application GIP mais d'une application créée entre temps pour m'entraîner avec le framework Vaadin, l'utilisation des View est la même que celle de l'application GIP.

//<https://vaadin.com/docs/-/part/framework/advanced/advanced-cdi.html>

Tout d'abord, il faut créer une énumération de « View » qui répertorie toutes les vues qui seront utilisées, l'énumération est beaucoup plus utile que de créer des vues à la volée si par la suite on veut ajouter des vues (il suffira d'ajouter une constante).

```

public enum viewType {
    HOME("home", HomeView.class, FontAwesome.HOME),
    CLIENT("client", ClientView.class, FontAwesome.USER),
    PROVINCE("province", ProvinceView.class, FontAwesome.MAP),
    EMPLOYEE("employee", EmployeeView.class, FontAwesome.GEAR),
    CHART("chart", ChartTestView.class, FontAwesome.PIE_CHART);

    private String viewName;
    private Class<? extends View> viewClass;
    private Resource icon;

    private viewType(String viewName,
                    Class<? extends View> viewClass, Resource icon) {
        this.viewClass = viewClass;
        this.viewName = viewName;
        this.icon = icon;
    }

    public Class<? extends View> getViewClass() { return viewClass; }
    public String getViewName() { return viewName; }
    public Resource getIcon() { return icon; }
}

```

Ensuite, nous allons créer un menu qui nous permettra de naviguer dans l'application, pour cela,

```

public final class Menu extends CustomComponent {

    public Menu() { ...6 lines }

    private Component buildContent() { ...14 lines }

    private Component buildToggleButton() { ...11 lines }

    private Component buildUserMenu() { ...11 lines }

    private Component buildTitle() { ...9 lines }

    private Component buildMenuItems(){

        CssLayout menuItemsLayout = new CssLayout();
        menuItemsLayout.addStyleName("valo-menuitems");

        for(ViewType view : ViewType.values()){

            Component menuItemComponent = new ValoMenuItemButton(view);
            menuItemsLayout.addComponent(menuItemComponent);
        }

        return menuItemsLayout;
    }

    public class ValoMenuItemButton extends Button{

        private ViewType view;

        public ValoMenuItemButton(ViewType view) {
            this.view = view;
            this.setStyleName("valo-menu-item");
            this.addStyleName(ValoTheme.BUTTON_BORDERLESS);
            this.setIcon(view.getIcon());
            setCaption(view.getViewName().substring(0, 1).toUpperCase()
                    + view.getViewName().substring(1));
            this.addClickListener(event -> UI.getCurrent().getNavigator().navigateTo(view.getViewName()));
        }
    }
}

```

Ce code permet de créer un menu (insérer image)

Pour pouvoir naviguer entre ses pages il nous faut un provider, pour cela il faut l'injecter dans la classe de l'UI principal via

```

@.Inject
private CDIVViewProvider viewProvider;

```

Nous devons ensuite ajouter les pages au provider

```

Navigator navigator = new Navigator(this, content);
navigator.addProvider(viewProvider);
for(ViewType viewType : ViewType.values()){
    navigator.addView(viewType.getViewName(), viewType.getViewClass());
}

//default page to display
navigator.navigateTo("home");

```

Une fois que tout cela est fait, il suffit de créer les pages sur lesquelles les informations seront présentent (tableau, carte, etc). Pour cela il faut annoter la classe avec CDIVView pour notifier à

l'application que c'est une vue, si la vue n'est pas présente dans l'énumération alors il y aura erreur d'exécution.

```
@CDIView("client")
public class ClientView extends Panel implements View {
```

8.1.2 Concept du push avec Vaadin

//Server push, « Learning Vaadin 7 » page 260

//<https://vaadin.com/docs/-/part/framework/advanced/advanced-push.html>

Lorsque l'on veut mettre à jour l'interface utilisateur via un Thread présent du côté serveur on veut que ce soit fait immédiatement , pour cela il faut utiliser le concept du “Push” qui envoie les données au navigateur immédiatement.

Pour utiliser push il faut ajouter les dépendances nécessaires dans le pom.xml que voici

```
<dependency>
    <groupId>com.vaadin</groupId>
    <artifactId>vaadin-push</artifactId>
    <version>${vaadin.version}</version>
</dependency>
```

Plusieurs modes de transport existent, le mode automatic qui va “push” après la méthode access() ou bien le mode “manual” qui permet de contrôler quand il faut push via la méthode “push” ce qui entraîne plus de flexibilité pour le développeur.

Pour activer les “push” sur l'UI, on peut utiliser les annotations suivantes,

Le mode automatic par défaut

```
@Push
public class PushyUI extends UI {
```

Active le mode manuel

```
@Push(PushMode.MANUAL)
public class PushyUI extends UI {
```

Les push sont utiles lorsque l'on veut mettre un graphique à jour en temps réel, ils permettent de mettre à jour l'interface graphique à chaque fois que l'on redessine le graphique.

8.1.3 Communication « Vaadin – Giplet »

La communication se base sur l'implémentation de publish subscribe, l'application Vaadin va souscrire un abonnement à un topic de GIP (chaque fois qu'un message sera reçu, le destinataire envoie les messages à ses destinataires ?), Lorsque l'application Vaadin va recevoir un message (Screen « @subscribe + explications) une méthode sera appelé sur l'instance d'un giplet. (Mettre des screens des @Subscribe + méthodes ?).

8.1.4 Remote Procedure Call (RPC)

Les « Remotes Procedure control » permettent de faire appel à des fonctions présentent dans la coté client, le navigateur web et de les utilisées du côté serveur (application).

Tout d'abord, il faut savoir que la carte présente sur le tableau de bord est un giplet implémenté en Javascript. L'image suivante illustre l'annotation ainsi que la classe héritée utilisée pour faire le lien entre le script JavaScript et l'application Java, cette classe est implémenté en JavaScript mais utilisée comme un composant Java.

```
@JavaScript({
    "vaadin://gipmap/javascript/oraclemaps2.js",
    "vaadin://javascript/dojo.xd.js",
    "vaadin://gipmap/javascript/map-connector.js"})
public class GipMap extends AbstractJavaScriptComponent {

    EventBus eventbus = new EventBus();
```

Des fonctions JavaScript sont déjà présentes pour répondre au besoin d'un clique sur un avion, il est dès lors possible de récupérer ses fonctions via RPC pour les implémentées en Java.

Le pattern publish subscribe est nécessaire pour notifier l'application Vaadin lorsqu'un avion est cliqué. Un eventbus sera créé. Celui-ci enverra des notifications lors d'un clique sur un avion.

Etant donné que la logique de l'application et la logique du giplet sont séparées, il faut enregistrer l'application principale auprès du giplet via la ligne suivante.

```
gipMap.register(this);
```

La méthode « register » correspond à la méthode ci-dessous, la méthode « unregister » permet de se désabonner de l'objet. L'interface « FeatureClickRpc » permet de définir les méthodes que l'on

utilisera (récupérera dans le JavaScript)

```
public void register(Object listener) {
    eventbus.register(listener);
}

public void unRegister(Object listener) {
    eventbus.unregister(listener);
}

public interface FeatureClickRpc extends ServerRpc {

    public void onFeatureClickTest(int seriesIndex, int dataIndex);

    public void onAircraftClick(Aircraft aircraft);

    public void onTetraClick(Tetra tetra);

    public void deleteAircraft(String flight);

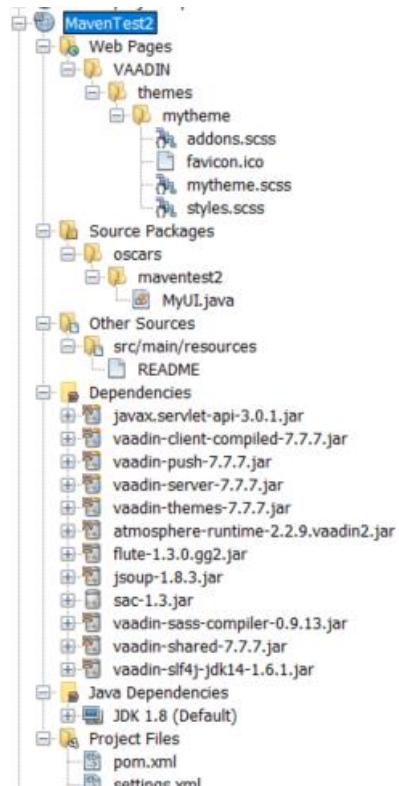
}
```

Pour effectuer le lien, voici les lignes qui ont dû être ajoutées dans le giplet Java.

Et voici les lignes qui ont dû être ajoutées dans le script JavaScript (insérer le code RPC javaScript)

Les avions seront reçus et affichés sur la carte Oracle grâce à des layers. Une méthode « onclick » est disponible qui a pour but d'afficher les infos d'un avion. Il faut récupérer, via RPC (image 2), ces méthodes et les implémentées en Java.

8.2 Hiérarchie de fichier d'un projet Vaadin



L'image ci-contre illustre la hiérarchie de fichier d'un projet Vaadin à son niveau le plus basique.

Comme présenté dans le chapitre 7.1 « Vaadin » l'architecture est d'avantage découplé visuellement dans le projet. La partie visuelle de l'application que sont principalement les themes, css et scss se trouve dans un répertoire séparé « Web Pages » tandis que la partie logique de l'application se trouve dans la partie « Source Packages ».

C'est dans cette dernière que les classes Java seront créées.

« Dependencies » représente l'ensemble des librairies utilisées par l'application qui se trouvent dans le pom.xml

« Project Files » est un package contenant des fichiers avec l'extension « .xml » qui permet une configuration de l'application. Par exemple nous pouvons modifier les configurations de lancement de l'application ou spécifier un emplacement d'une librairie Maven.

Plus précisément, Le fichier « pom.xml » se constitue de l'ensemble des configurations nécessaire pour l'application web, c'est notamment dans celui-ci qu'on y ajoute les dépendances nécessaires ainsi que des informations sur la manière dont le projet va être builder, sur quelle adresse local, etc. Tout cela grâce à Maven.

Lorsque que l'on veut se connecter à une base de données, il faut créer un fichier persistence.xml qui se situera dans le répertoire « src/main/resources ». Ce fichier est constitué de l'ensemble des informations et configuration nécessaire pour se connecter à une base de données déjà existante (serveur, port, JNDI, entity, etc).

« JDK 1.8 » signifie qu'on utilise les librairies de la huitième version de Java.

9 Conclusion

Tout d'abord, Ce stage de quinze semaines dans la société OSCARS m'a permis d'approfondir mes connaissances dans le langage Java grâce à une équipe R&D soudée et motivée mais j'ai également pu découvrir un nouveau framework Java2EE très utile, Vaadin. Ce framework fut une révélation, en effet, le fait que sa conception puisse gérer la partie interface client m'a conquis.

Ensuite, les outils et les technologies utilisés durant le stage ont demandé une recherche personnelle et une compréhension. Il m'a été difficile d'appréhender et de définir la place de chaque outil et technologie dans le business model de l'entreprise par faute de documentation. Lors du début du stage jusqu'à la moitié de celui-ci j'étais dans le brouillard jusqu'à ce qu'on m'explique le fonctionnement de GIP et la place des outils et technologies.

Enfin, les connaissances et l'expérience acquises durant le stage m'a permis de faire une corrélation avec les projets réalisés durant mon parcours scolaire, ceux-ci étant trop basique et trop facilement applicable. Il en est tout autre lors de l'implémentation et de l'analyse d'un projet en entreprise dû à certaines contraintes notamment les deadlines.

Pour conclure, je dirai que ce stage m'a permis d'en apprendre d'avantage sur les technologies Java2EE et m'a permis de renforcer mon intérêt pour celles-ci, néanmoins une phase de recherche intense m'a été demandé pour acquérir toutes les connaissances en rapport avec les outils et technologies utilisés, de plus j'ai pu me baser sur mes connaissances déjà acquises grâce à mon parcours scolaire ainsi que les différents projet qui ont dû être réalisés.

//////////

Durant ce stage j'ai su me familiarisé avec de nouvelles technologies en relation avec le corps métier de la société mais j'ai également su approfondir mes connaissances en JAVA grâce à une phase d'apprentissage endéans laquelle je me suis beaucoup renseigner sur les sites tels que dzone. J'ai pu approfondir d'une part le composant web et d'autre part la huitième version de Java –qui n'est pas récente- mais qui apporte énormément de nouveautés et de fraîcheur au langage.

Le premier mois de stage fut une phase de découverte concernant le framework Vaadin et les technologies utilisées mais également des membres de l'équipe d'OSCARS. Pour aider à cette familiarisation il m'a été indispensable de concevoir des applications (projets) diverses qui traitent toutes d'une fonctionnalité du framework pour en comprendre la quintessence de Vaadin.

Le deuxième mois s'est concentré sur la configuration, la mise en place des technologies et outils nécessaire au bon fonctionnement de l'environnement futur des applications ainsi qu'une

compréhension de la plateforme GIP et également une implémentation d'un premier widgets « IFrame ».

Le troisième et le quatrième mois furent la continuité des 2 autres, c'est-à-dire une relecture et une **modification** de l'analyse déjà effectuée précédemment mais également l'implémentation des widgets manquants.

Pour conclure, je dirai que ce stage m'a permis de découvrir énormément de nouvelles choses grâce à une équipe R&D toujours présente pour répondre à mes questions mais aussi pour m'accompagner dans l'accomplissement du travail demandé en début de stage. Il m'a également permis de me rendre compte de mes lacunes et les solutions pour les combler. Ce stage m'a permis de me diriger vers les développements web qui pour moi n'en sont qu'à leurs début.

10 Bibliographie – Webographie

<https://vaadin.com/vaadin-fw8-documentation-portlet/framework/introduction/intro-overview.html#figure.intro.architecture>

<https://vaadin.com/docs/-/part/framework/architecture/architecture-overview.html>

<http://www.oscars-sa.eu/fr/presentation>

<https://vaadin.com/vaadin-fw8-documentation-portlet/framework/architecture/architecture-technology.html#architecture.technology.ajax>

<https://vaadin.com/vaadin-fw8-documentation-portlet/framework/architecture/architecture-overview.html#architecture.overview>

https://fr.wikipedia.org/wiki/Google_Web_Toolkit

<https://maven.apache.org/what-is-maven.html>

https://fr.wikipedia.org/wiki/Oracle_SQL_Developer

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

https://fr.wikipedia.org/wiki/IntelliJ_IDEA

https://fr.wikipedia.org/wiki/Apache_Maven

<https://www.oracle.com/fr/middleware/weblogic/index.html>