



# HD_cotopaxi:Users:richy:Documents:Mes_logos:AUTRES:ur1-ueb:91144_UR1_UEBtransparent.psd

ISTIC – UFR Informatique et Électronique  
Université de Rennes 1, Campus de Beaulieu

263, avenue du Général Leclerc

CS 74205, 35042 Rennes CEDEX, France

**02 23 23 39 00**

DIALLO Issaga

**Master 2 Informatique :** Méthode Informatiques et Technologiques de l’Information et de la Communication

**Stage effectué du** 4 mars 2013 au 30 août 2013 à Saint Grégoire

Soutenance à Rennes le 06 septembre 2013

**Sous la responsabilité de :** Yann BARRAULT

Développement en JavaEE avec une plateforme PAAS : Cloud Foundry de Spring Source (VMware)

# Remerciements

Je tiens à remercier avant de commencer, toute l’équipe pédagogique de l’ISTIC ainsi que les intervenants de la formation de MITIC pour m’avoir assuré la partie théorique. Je remercie également Monsieur François POULET et Madame Virginie SANS, pour l’aide et les conseils concernant le déroulement et la suite du stage qu’ils m’ont apporté lors des différents suivis.

Je tiens à remercier tout particulièrement et à témoigner toute ma reconnaissance aux personnes suivantes, pour l’expérience enrichissante et pleine d’intérêt qu’elles m’ont fait vivre durant ces mois de stage au sein de SODIFRANCE : Monsieur Yann BARRAULT, Monsieur Emmanuel LESNE et Manuel DORANLO tous consultants et tuteurs, pour l’accueil et la confiance qu’ils m’ont accordé dès mon arrivée dans l’entreprise mais aussi pour le temps consacré tout au long de cette période.

# Sommaire

[Remerciements 0](#_Toc365019491)

[Sommaire 2](#_Toc365019492)

[Introduction 3](#_Toc365019493)

[1. Présentation de SODIFRANCE 4](#_Toc365019494)

[1.1. Historique 4](#_Toc365019495)

[1.2. Services 5](#_Toc365019496)

[1.3. Présentation de l’entité ANTEO Consulting 6](#_Toc365019497)

[1.3.1. Missions d’ANTEO Consulting 6](#_Toc365019498)

[1.3.2. Objectifs 6](#_Toc365019499)

[2. Description du stage 7](#_Toc365019500)

[2.1. Définitions 7](#_Toc365019501)

[2.1.1. Cloud computing 7](#_Toc365019502)

[2.1.2. PaaS 7](#_Toc365019503)

[2.2. Description 7](#_Toc365019504)

[2.3. Objectifs 8](#_Toc365019505)

[2.3.1. Besoins 8](#_Toc365019506)

[2.3.2. Objectifs 8](#_Toc365019507)

[3. Etude et développement sur plateforme PAAS 9](#_Toc365019508)

[3.1. Etude comparative des solutions 9](#_Toc365019509)

[3.1.1. Terminologies 9](#_Toc365019510)

[3.1.2. Recensement de solutions PaaS 12](#_Toc365019511)

[3.1.3. Présentation de certains leaders fournisseurs de PaaS 12](#_Toc365019512)

[3.1.4. Prise en mains des solutions PaaS 17](#_Toc365019513)

[3.1.5. Cas d’usages du développement sur cloud 19](#_Toc365019514)

[3.1.6. Comparaison des solutions PaaS 21](#_Toc365019515)

[3.2. Documents Livrables comparatif 26](#_Toc365019516)

[3.2.1. Document de comparaison des PaaS 26](#_Toc365019517)

[3.2.2. Présentation des PaaS leader 29](#_Toc365019518)

[3.2.3. Compte rendu de semaine 30](#_Toc365019519)

[3.3. Développement de l’application Swift sur OpenShift 30](#_Toc365019520)

[3.3.1. Description de l’application développée 30](#_Toc365019521)

[3.3.2. Méthodes de développement 30](#_Toc365019522)

[3.3.3. Caractéristiques techniques 31](#_Toc365019523)

[3.3.4. Réalisations 31](#_Toc365019524)

[3.4. Connaissances acquises 32](#_Toc365019525)

[3.5. Recueil d’informations 32](#_Toc365019526)

[Conclusion 32](#_Toc365019527)

[Annexes 33](#_Toc365019528)

[Tables des figures 36](#_Toc365019529)

[Bibliographie 36](#_Toc365019530)

# Introduction

Durant une enthousiasmante année universitaire de Master 2 informatique, spécialité MITIC au sein de l’ISTIC, il m’a été demandé de réaliser un stage 6 mois en entreprise durant lequel je devrais me mettre en situation d’actif, afin de répondre à certaines attentes vis-à-vis de l’entreprise accueillante.

Cette expérience a pour objectif d’apporter à l’étudiant une vision différente de celle vue en cours, puisqu’elle se déroule dans le milieu professionnel. Outre, le travail sur le mémoire, cette expérience permet à l’étudiant d’évaluer son adaptation en entreprise et de mettre en pratique les connaissances théoriques qui lui ont été dispensées lors de la formation.

Pour ma part, mon stage de fin d’étude s’est déroulé chez SODIFRANCE. Cette société, basée à Saint Grégoire, est spécialisée dans la migration d’applications ainsi que dans la modernisation des systèmes d’informations.

Dans le cadre de l’amélioration de leur pole conseil, il m’a été demandé d’étudier et de comparer des plateformes de développement disponibles sur le cloud.

A travers la recherche, l’analyse et la comparaison des solutions, c’est tout le cycle de vie du projet qui doit être étudié.

Concernant la structure du document, je commencerai par une description de SODIFRANCE et du groupe ANTEO Consulting. J’introduirai le sujet du stage par des définitions. Ensuite, par des études, je comparerais plusieurs solutions par rapport à des critères à définir. Pour finir, je présenterais une application qui a été déployé sur l’une de ces solutions.

# Présentationde SODIFRANCE

## Historique

Depuis sa création en 1986, SODIFRANCE a mis en œuvre une stratégie basée sur l’expertise technologique, la proximité géographique et l’adaptation permanente de ses offres et compétences pour répondre au mieux aux besoins de ses clients.

#### Création et développement

SODIFRANCE a été créée en 1986 à Rennes par Francis MAZIN. Positionnée à sa création sur le métier de l'externalisation pour le secteur bancaire, SODIFRANCE développe rapidement son offre de services informatiques et accroît son implantation nationale.

En 1992, SODIFRANCE créé une nouvelle activité stratégique pour son développement : la [**transformation automatisée**](http://www.sodifrance.fr/services/modernisation-migration-du-si/) des systèmes d’information.

En 2004, SODIFRANCE créé [**Mia-Software**](http://www.mia-software.com/) pour commercialiser ses solutions logicielles issues de cet important savoir-faire en ingénierie de modèles, notamment dans le domaine du Model Driven Engineering (MDE).

Depuis 2005, la stratégie de SODIFRANCE conjugue :

#### Le développement permanent des offres et des compétences

* SODIFRANCE poursuit ses investissements en R&D pour développer le périmètre de ses offres de transformation automatisée : extension de la couverture technologique de ses offres de retro documentation et de migrations, création d'une offre d’[**évolution d’architecture**](http://www.sodifrance.fr/services/modernisation-migration-du-si/) baptisée FAST (Future Architecture System Transformation). Ainsi SODIFRANCE devient le N°1 français sur le marché du « LEGACY MODERNISATION »
* En complément, SODIFRANCE enrichit ses offres de services informatiques. Cela se traduit notamment par l’extension de ses compétences technologiques, qui combinent compétences grands systèmes (mainframe) et nouvelles technologies (J2EE, .NET …) et par de nouvelles offres autour des [**services d’infrastructure**](http://www.sodifrance.fr/services/infrastructure-services/infrastructure-management/), du [**développement applicatif**](http://www.sodifrance.fr/services/developpement-integration/filieres-de-developpement/) et de l’ [**intégration de solutions**](http://www.sodifrance.fr/services/developpement-integration/filieres-de-developpement/) (Décisionnel, Portails, etc.).

En 2007, SODIFRANCE acquiert [**ONEXT**](http://www.onext.fr/), société spécialisée dans la réalisation de portails web (internet, intranet, extranet) basés sur des technologies de gestion de contenu CMS.

#### Une présence forte dans le secteur Banque-Assurance-Protection sociale

Déjà fortement présente dans ce secteur (+ de 60% du CA), SODIFRANCE acquiert en 2007 la société API-GROUP. Cela donne lieu à la naissance d’un pôle significatif dans le domaine du service informatique Banque-Finance : ainsi, depuis 2007, SODIFRANCE figure chaque année dans le TOP 10 des fournisseurs informatiques du secteur de l’Assurance.

#### Industrialisation et CMMI (*Capacity Maturity Model Integrated*)

Réalisant de nombreux projets au forfait, SODIFRANCE a mis en place un modèle de production industriel piloté par un dispositif dédié - SODIFRANCE Delivery Center - et bénéficiant d'une approche qualité rigoureuse.

Le succès de la stratégie est validé par sa dynamique de croissance avec une progression chaque année de son chiffre d’affaires. En 2012, SODIFRANCE a réalisé un chiffre d’affaires de 73,1 M€, en croissance de + 7,2%.

## Services

SODIFRANCE diversifie ses services. Il s’agit de :

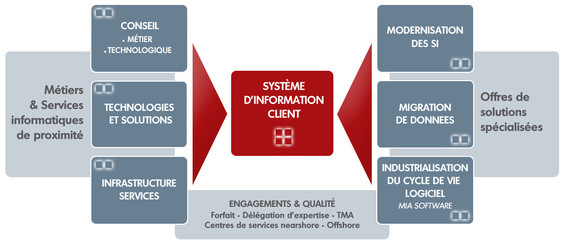


Figure - L'offre de solutions et de services de SODIFRANCE

* **Conseil (ANTEO)**: qui est pôle conseil de SODIFRANCE, nous en parlerons dans la partie suivante.
* **Développement et intégration** : SODIFRANCE dispose de compétences et d'expertises technologiques dans le développement NTIC (JEE, .NET, PHP…) et Mainframe (Cobol et Natural).
* **Modernisation et migration du SI**:SODIFRANCE est spécialisée dans l’évolution des SI vers .NET, JEE et FLEX, la migration de plate-forme, la conversion de données, la conversion de langages ou d’AGL, l’architecture orientée services, l’urbanisation du système d’information.
* **Data Management** : SODIFRANCE propose, au travers de son offre Datalliance, une démarche globale pour maîtriser le cycle de vie des données.
* **Infrastructure Services** : cette offre est basée sur 5 volets distincts : le conseil, la virtualisation, la sécurité, la supervision et l’infogérance.
* **Prestations et engagements** : l’offre de services de SODIFRANCE met en œuvre des solutions **industrialisées** à engagement de résultat qui se déclinent à travers ses offres de **TMA**, de **Centres de Services** de proximité et **Offshore.** Elles sont réalisées dans le cadre d'un **référentiel unique CMMI.**
* **Formation** : avec SODIFRANCE Institut qui est la filiale Formation de SODIFRANCE.

## Présentation de l’entité ANTEO Consulting

ANTEO Consulting est l’offre conseil et expertise de SODIFRANCE.

Véritable accélérateur de projets et de décisions stratégiques, ANTEO Consulting intervient auprès de ses clients sur des missions de conseil et d'expertise en architecture et méthode. ANTEO Consulting accompagne efficacement les réflexions et les équipes et apporte les réponses aux problématiques soulevées par les enjeux des Systèmes d'Information. Ces réponses apportées tournent autour de six axes :

* Gouvernance et processus ;
* Urbanisation et architecture d’entreprise ;
* Architecture technique ;
* Industrialisation et MDA ;
* Alignement MOA – MOE ;
* Formation et conduite de changement.

### Missions d’ANTEO Consulting

Face aux enjeux de coût et d'agilité du SI, ANTEO-Consulting aide les organisations à structurer leurs réflexions et leurs démarches :

* Comment **structurer mon organisation** pour s’aligner sur les bonnes pratiques du marché ?
* Comment **garantir à mon métier la bonne prise en compte de ses exigences** ?
* **Quels choix technologiques et d'architecture pour un système d’information pérenne**, agile et ouvert, tenant compte de mon patrimoine applicatif existant ?
* Comment **maîtriser la qualité, les coûts et les délais** de mes nouveaux projets ?
* Comment **accompagner mes équipes vers les nouveaux enjeux techniques** et fonctionnels ?

### Objectifs

L’entité effectue des interventions agiles et sur mesure. ANTEO Consulting a pour objectif d'aider les entreprises en les accompagnants dans leurs réflexions stratégiques sur leur SI. Ceci au travers la conduite de missions accélératrices, conjuguant retours d’expérience clients et connaissance des technologies et du marché :

* Audit (technique, organisationnel…) ;
* Etude d’opportunité / étude de cadrage / orientations stratégiques ;
* Schéma Directeur SI ;
* Etude d’architecture / conception d’usine logicielle ;
* Dossier de choix de solution ou de technologie ;
* AMOA technique ou fonctionnelle / conduite d’appel d’offre / pilotage de sous-traitant ;
* Support / accompagnement / Tutorat ;
* Formation institutionnelle.

# Description du stage

## Définitions

Avant de rentrer dans le vif du sujet, nous allons définir certains termes nécessaires pour bien cadrer le domaine.

### Cloud computing

Le cloud computing décrit un environnement informatique composé de ressources de calcul mutualisées et distribuées basées sur la demande client. Il peut être utilisé par tout type d’utilisateur. Les services cloud sont utilisés pour traiter une variété d’applications et de besoins en infrastructures tels que les bases de données, les espaces de stockage, capacité de calcul, etc. Cet environnement a la propriété d’être mise à disposition du consommateur en quelques minutes. Le cloud computing est un style d’utilisation de ressources de calcul. Ce style est définit par des capacités technologiques qui sont livrées en tant que services en utilisant internet. Le cloud computing se compose de plusieurs caractéristiques essentielles et se divise en plusieurs couches de services. C’est l’une de ces couches que nous étudierons, dans notre cas, la couche PaaS : Platform-as-a-Service (plateforme en tant que service).

### PaaS

PaaS : est une couche du cloud computing. Nous reviendrons en détail pour situer celle-ci dans la hiérarchie du cloud. Cette couche délivre en service, tout un ensemble de technologies de développement destiné aux développeurs, chef de projet, gestionnaire d’une application ou autre personne pouvant être impliquée dans le développement d’une application.

## Description

Dans ce stage, il s’agit d’abord de comparer plusieurs offres de services PaaS selon des critères et besoins à définir ensuite d’en choisir une pour une étude de cas. L’étude comparative comprend plusieurs phases pouvant être abordées dans l’ordre suivant :

* **Prise de connaissances du sujet** : explication du sujet par les différents tuteurs de stage sur suivant un périmètre limité dans le but de mieux faire comprendre et éviter une grande complexité.
* **Premières utilisations de PaaS**: dans cette phase, il s’agit de prendre en main la solution PaaS **CloudFoundry de Pivotal** afin d’avoir un premier aperçu de l’utilité d’une plateforme PaaS et de comment elle pourrait être utilisé. Pour ce faire, une petite application Java Web Dynamique est développée en utilisant les ressources de cette plateforme. Cette application Java utilise les technologies JSF + Hibernate JPA avec une base de données MySQL de cinq tables environ. Nous reviendrons sur cette application dans la partie « Prise en main des solutions PaaS ».
* **Définition des termes** : après les premières utilisations un recul est jugé nécessaires dans le but d’avoir une plus grande vue de ce à quoi pourrait servir une plateforme PaaS. Ainsi, après avoir déterminé et analysé quelques possibilités d’usages, il semble bien important de définir certains termes intimement liés aux PaaS. Nous aborderons à cet effet, une partie nommée « Terminologies ».
* **Elargissement et délimitation des frontières du sujet** : le recul pris dans la phase précédente et la recherche des définitions des termes liés aux PaaS permet de mieux comprendre jusqu’où l’on pourrait aller avec une PaaS. Concrètement, nous devons étudier différents environnements cloud avec pour objectif de recenser les fonctionnalités pouvant accompagner une PaaS.
* **Définition des besoins** : une bonne compréhension des besoins s’impose. Pour cela, plusieurs mises en situation réelles de développement sont faites avec d’autres stagiaires développant d’autres applications.

## Objectifs

Tout en restant dans le cadre du stage, nous allons décrire les objectifs tout en tenant compte des besoins et contraintes définis par le groupe ANTEO.

### Besoins

Dans le but d’améliorer et d’élargir son offre de service, ANTEO souhaite approfondir ses connaissances en solutions PaaS pour mener les clients à choisir l’offre la plus adaptées aux attentes. Ces attentes peuvent être d’ordre technique ou non : temps de déploiement, type de serveur d’application, partage de connaissance, possibilité d’obtenir un rapport de la consommation des ressources etc. Pour faire un bon choix de PaaS, il semble donc nécessaire d’étudier et de comparer le fonctionnement, les services offerts, les possibilités d’usages et les prix des différentes plateformes.

### Objectifs

* **Etude comparative des solutions/offres PAAS**
  + Quels sont les apports, avantages/inconvénients, limites des solutions du marché ?
  + Développer une application en JavaEE/Spring
  + Déployer l’application et celles des autres stagiaires
* **Evaluer les contraintes**
  + Minimiser l’utilisation des ressources locales du poste de développement
  + Installer un nouveau poste de développement en moins de 10 minutes, une utopie ?
* **Restitution & Transfert de compétences**
  + Restitution de l’étude comparative : ***présentation PowerPoint & rapport***
  + Réaliser des ***supports et ateliers de formation*** sur la solution retenue

# Etude et développement sur plateforme PAAS

## Etude comparative des solutions

Notre étude porte sur la comparaison de certaines offres de services de cloud computing. Nous avons ainsi ciblé parmi les leaders du marché, des fournisseurs permettant d’avoir les ressources nécessaires pour un environnement supportant tout ou une partie du cycle de vie d’une application allant de la spécification des besoins à la livraison, et la maintenance.

### Terminologies

Le cloud computing est un modèle pour permettre l’accès réseau à la demande de ressources configurables partagées omniprésentes (réseau, espace mémoire, applications, et services), qui peuvent être rapidement fournies et libérées avec un effort minimal de gestion ou d’interaction avec le fournisseur. Ce modèle est caractérisé par certaines propriétés et composé de modèles de déploiement et de modèles de services ([les différentes couches](#_Couches_principales_du)).

#### Caractéristiques d’une offre cloud

Une offre cloud doit présenter certaines caractéristiques. Nous allons reprendre celles définies par l’institut national des standards et de la technologie du département de commerce des Etats Unis : **NIST** pour **N**ational **I**nstitute of **S**tandards and **T**echnology. Cet institut NIST est responsable de l'élaboration de normes et de lignes directrices, y compris les exigences minimales, pour assurer la sécurité de l'information adéquate même si ces normes et lignes directrices ne s'applique pas aux systèmes de sécurité nationaux des Etats Unis. Par ailleurs, pour montrer l’importance de cet institut, nous pouvons prendre l’exemple d’[Eurocloud](http://www.eurocloud.fr/index.php?option=com_content&view=article&id=44&Itemid=67) (1er réseau d’acteurs cloud en Europe) qui reprend les mêmes définitions du cloud computing.

* Libre-service à la demande
* L’offre cloud doit être accessible sur l’ensemble d’un réseau
* Les ressources sont mutualisées
* Le cloud doit s’adapter rapidement aux différentes variations des besoins lors de l’utilisation des ressources
* Le service doit être mesurable

#### Modèles de cloud

Il existe quatre modèles de déploiement cloud.

* Le cloud privé ;
* Le cloud publique ;
* Le cloud hybride ;
* Le cloud communautaire.

Voir en annexe les définitions de ces 4 modèles.

#### Les couches principales du cloud

Le cloud est structuré en trois couches principales.

##### IaaS (Infrastructure-as-a-Service)

**Infrastructure en tant que services** est une infrastructure informatique consommée en tant que service. Chaque utilisateur ou locataire accède à une portion de ressources fédérées pour créer et utiliser sa propre infrastructure au besoin, quand et comment il le veut.

Ce qui est offert au consommateur, c’est la capacité à approvisionner le traitement, le stockage, les réseaux, et autre ressources fondamentales où le consommateur est capable de déployer et d’exécuter un logiciel qui peut inclure des systèmes d’exploitations, ou applications. Le consommateur ne gère ni ne contrôle l’infrastructure cloud sous-jacente, mais a le contrôle sur les systèmes d’exploitations, le stockage, les applications déployées et éventuellement un contrôle limité sur les composants réseaux.

L'IaaS est le premier modèle de cloud, où :

* **Le fournisseur Cloud** maintient : la virtualisation, le matériel serveur, capacité de stockage et capacité de calcul à la demande sur un réseau;
* **L'entreprise cliente** maintient : les applications, les environnements de développement et d'exécution d'applications (incluant systèmes d’exploitation), l'intégration SOA, les bases de données, le logiciel serveur.

L’IaaS est le modèle adapté aux entreprises souhaitant utiliser leur propre cloud sans avoir à supporter des investissements en infrastructure. Le client IaaS peut être un fournisseur PaaS. Cette couche donne la possibilité au consommateur d’avoir un contrôle total sur l’environnement qu’il veut fournir, mais nécessite plus de travail (administration, gestion…)

##### PaaS (Platform-as-a-Service)

**Plateforme en tant que service** est un environnement de computing accédé au besoin sur un réseau à partir d’un fournisseur de service. PaaS est utilisé pour développer et exécuter des logiciels comme alternative à l’architecture, la construction et l’installation d’un environnement local de développement et de production.

Un client utilise la plateforme PaaS pour porter ses applications dans le cloud ou en développer d'autres à partir des outils de développement fournis par la plate-forme de cloud computing, sans se soucier du coût et de la complexité de l’achat et de la gestion du matériel sous-jacent, des logiciels et utilitaires.

PaaS fournit toute la facilité requise pour supporter le cycle de vie complet d’une application, de sa construction à la livraison.

La plateforme PaaS est le second modèle de cloud, où :

* **Le fournisseur cloud** maintient : les applications, les environnements de développement et d'exécution d'applications (incluant systèmes d’exploitation), l'intégration SOA, les bases de données, le logiciel serveur, la virtualisation, le matériel serveur, le stockage, les réseaux. Exemple de fournisseur : OpenShift qui propose une suite de logicielle Linux + Apache + MySQL + Java;
* **L'entreprise cliente** maintient uniquement les applications qu’il déploie au travers des services, librairies, langages fournis. Exemple de services fournis: base de données MySQL, serveur JBoss.

Le PaaS est le modèle adapté aux entreprises souhaitant contrôler le déploiement de ses applications ou qui veulent développer leurs propres applications. Le client ici fournit l’application finale. Cette couche permet d’une part de gagner en temps de développement, en termes de coût des infrastructures et de consommation en ressource (pour une meilleure scalabilité). Dans une autre part, les équipes de développement ont moins de contrôle sur leurs environnements de travail.

##### SaaS (Software-as-a-Service)

**Un logiciel en tant que service** est un logiciel utilisé sur un réseau sans qu’il soit téléchargé et installé dans un environnement local. Le logiciel est accédé par d’internet (avec un navigateur web ou autre programme interfacé) à partir d’un fournisseur SaaS et est exécutée dans un environnement prédéfini par ce même fournisseur. Le consommateur ne gère ni ne contrôle l’infrastructure cloud sous-jacente.

SaaS pour « Software-as-a-Service » est l'ultime modèle de cloud, où :

* **Le fournisseur** Cloud maintient : les applications qu’il déploie, les environnements de développement et d'exécution d'applications (incluant systèmes d’exploitation), l'intégration SOA, les bases de données, le logiciel serveur, la virtualisation, le matériel serveur, le stockage, les réseaux ;
* **Le client** accède aux applications et services via un navigateur internet. Par exemple, ce peut être une banque qui loue un logiciel de comptabilité en ligne, à la demande.

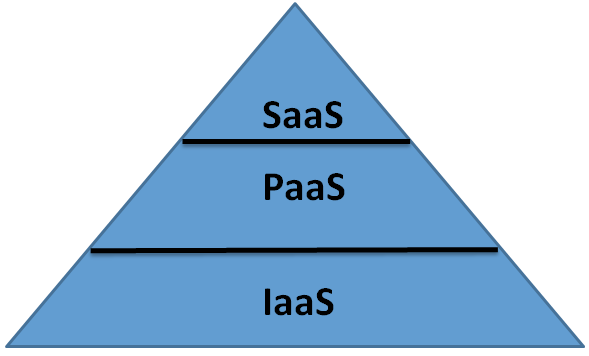


Figure - Hiérarchie des couches du cloud

### Recensement de solutions PaaS

Dans le marché actuel, certaines plateformes se positionnent parmi les leaders de par les services offerts et la popularité du fournisseur. Nous retrouvons ces plateformes parmi les plus hautes classées par les revues informatiques telles que Journaldunet ou Cloudtimes.

|  |  |  |
| --- | --- | --- |
| Nom | Entreprise | date de création |
| CloudFoundry de Pivotal | PIVOTAL | 2011 (par VMware) |
| CloudBees | CloudBees | 2010 |
| Heroku | Salesforces | 2007 |
| Amazon Web Services EC 2 | Amazon Web Services |  |
| Windows Azure | Microsoft | 2009 |
| OpenShift | Red Hat | 2011 |
| Engine Yard | Engine Yard | 2006 |
| AppHarbor | AppHarbor | 2010 |
| CloudControl | CloudControl |  |
| sTATIC | STATIC | 2013 |

Figure - Liste des solutions cloud PaaS étudiées

### Présentation de certains leaders fournisseurs de PaaS

#### CloudFoundry de pivotal

CloudFoundry est une plate-forme PaaS libre, au départ développée par VMware sous les termes de la licence Apache License 2.0. Elle est développée en [Ruby](http://www.ohloh.net/p/cloudfoundry) et actuellement maintenue par l’entreprise [Pivotal](http://www.gopivotal.com). Cloud Foundry fournis un choix de clouds, de Frameworks de développement et de services pour les applications. Cloud Foundry rend simple la construction, le test, le déploiement et la mise à l’échelle d’une application. C’est un projet open source accessible à travers [une variété de distributions clouds](https://core.cloudfoundry.org/listings) privées incluant [CloudFoundry.com](http://cloudfoundry.com).

CloudFoundry permet au développeur de plus se concentrer sur l’implémentation des applications que sur l’infrastructure. Cette PaaS permet d’écrire des applications portables, tester, adapter, déployer sans changer le code source. CloudFoundry est un socle à partir duquel d’autres fournisseurs se basent pour offrir des services PaaS plus élargis. Il s’agit à ce jour de :

* Static
* TIER 3
* Uhuru
* NTT Communications
* Appfog
* MoPaaS

|  |  |
| --- | --- |
| Type de cloud | Fournisseurs d’infrastructure |
| Publique, privé, hybride | * Par défaut : * Autres partenaire : vSphere, AWS, OpenStack, Rackspace |
| Fonctionnalités | **Interfaces** |
| * Déploiement * Gestion du cycle de vie d’une application * Surveillance des applications * Gestion d’identité * Collaboration pour le développement * Accès au système par avec des API REST | * CLI * Plugin IDE * Console web * API et librairies |
| Frameworks | **Services complémentaires** |
| Grails, Java Web, Lift, Node.js, Play, Rack, Sinatra, Spring, Standalone | Base de données, outils de recherche, queuing, analyse de données, analyse de performance |
| Avantages | **Inconvénients** |
| * Possibilité d’intégration à un IDE * Possibilité d’ajouter plusieurs environnements d’exécution pour une même application * Il existe plusieurs fournisseurs CloudFoundry pour étendre les services et fonctionnalités offertes | * Très peu de modules complémentaires * La console web n’offre pas large contrôle sur les applications, services et sur les utilisateurs * Pas de gestionnaire de code source * Pas de sauvegarde d’application |

Figure - Carte d'identité de CloudFoundry

#### OpenShift online de red hat

OpenShift Online est l’offre cloud de Red Hat. C’est une plateforme Open source où les développeurs peuvent construire, tester, déployer et exécuter leurs applications. Elle fournit également des outils intégrés pour le développement pour supporter le cycle de vie d’une application; il s’agit de l’intégration à Eclipse, Jenkins, Maven et Git. La plateforme fournit une scalabilité manuelle ou automatique des ressources d’une application.

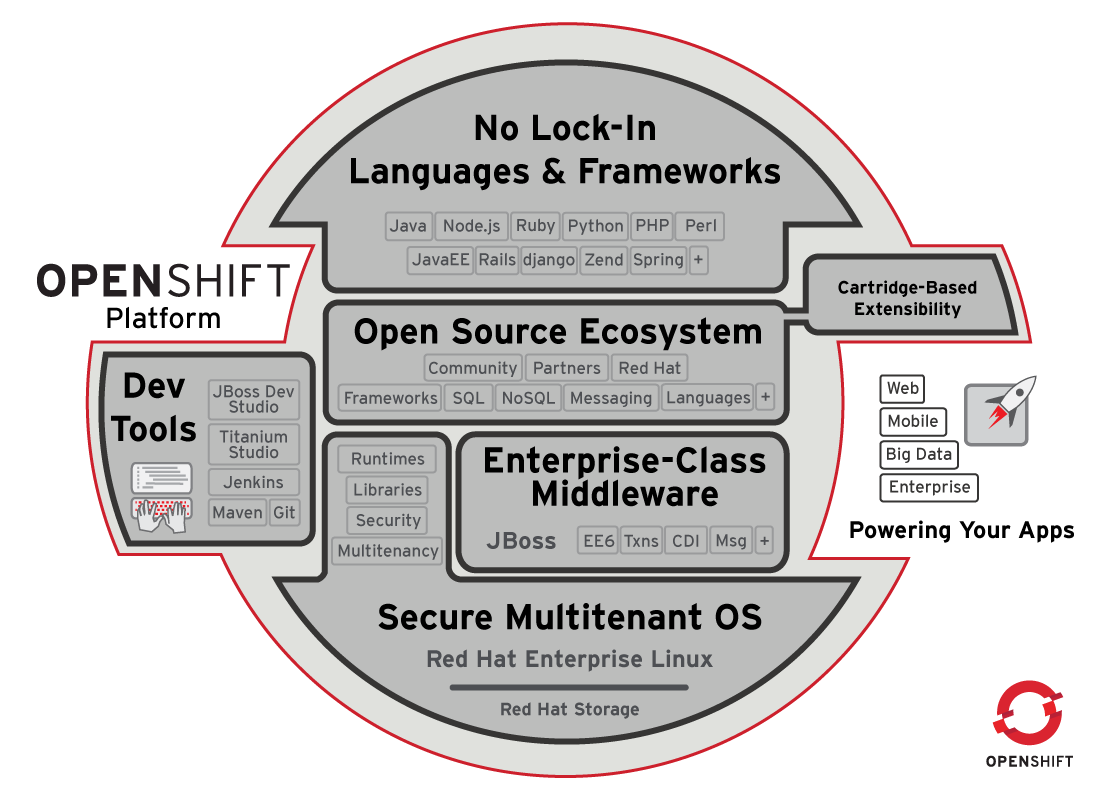


Figure - Présentation d'OpenShift

|  |  |
| --- | --- |
| Type de cloud | Fournisseurs d’infrastructure |
| Publique, privé |  |
| Fonctionnalités | **Interfaces** |
| * Variables d’environnement * Exécution de tâches * Surveillance d’applications * Gestion des services * Sauvegarde des applications * Environnements multiples * Intégration continue avec Jenkins * Gestion de code source * Scalabilité automatique ou manuelle | * CLI * Console Web * Plugin IDE |
| Frameworks | **Services complémentaires** |
| JBoss, Spring, Tomcat, Zend, Codelgniter, Rails, Node.js, Django, Flask, Perl, DIY (Do-It-Yourself) | Base de données SQL et NoSQL, analyse de données Big Data, gestionnaire de base de données, intégration continue, mail, exécution de tâches, analyse de la charge, collaboration, gestionnaires de code sources |
| Avantages | **Inconvénients** |
| * Support technique de la communauté actif |  |

Figure - Carte d'identité d'OpenShift Online

#### CloudBees

La plateforme PaaS  a été créée en 2010; l’équipe est actuellement composée d’un ancien directeur technique de JBoss de Kohsuke Kawaguchi, le fondateur et développeur principal de Jenkins CI et de travailleurs aux USA, en Europe, et en Australie. La vision de CloudBees est d’éviter au  développeur les tâches de maintenances des infrastructures ou mises à jour logicielles, lui permettant ainsi de mieux se concentrer sur l’implémentation de son application. CloudBees offre une plate-forme qui englobe la totalité du cycle de vie d’une application allant du développement à la mise en production. La figure ci-dessous illustre la plateforme, son fonctionnement et les services offerts.

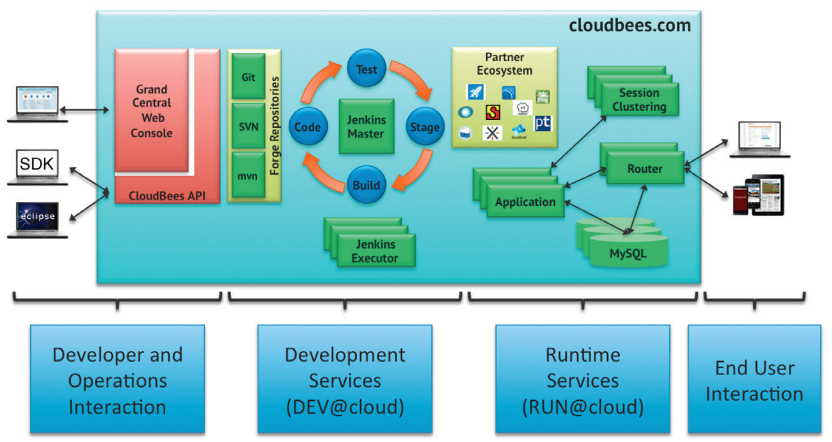


Figure - Présentation de CloudBees

|  |  |
| --- | --- |
| Type de cloud | Fournisseurs d’infrastructure |
| Publique, privé |  |
| Fonctionnalités | **Interfaces** |
| * Création d’application * Intégration continue avec Jenkins * Notification d’évènements de diverses sources * Contrôle de la scalabilité * Gestion de code source * Choix du fournisseur cloud * Surveillance d’applications * Intégrations de services externes | * CLI * Console Web * Plugin Eclipse * API basée sur HTTP |
| Frameworks | **Services complémentaires** |
| Tomcat 7, Hibernate, Java EE 6 Web Profile, Glassfish, Jetty, JBoss 7, Akka, Liferay, Grails, iU Web Apps, SOASTA CloudTest, MacOS executors, Google Go Webapp, Facebook App, Drupal, Play, Lift, Webmachine, Dart App, Clojure/Compojure, Node.js, Metor.js, Angular.js, Backbone.js, JAX RS, JRuby on Rails, Railo | Analyse de performance, message queuing, ALM, gestionnaire de dépôts, base de données SQL et NoSQL, journalisation, analyse continue de code source, moteur de recherche, mail, wiki, IDE partenaires, mémoire cache, analyse de données Big Data, déploiement continu (CF, GAE, Codenvy). |
| Avantages | **Inconvénients** |
| * Possibilité d’intégration à un IDE. * Une variété de modules complémentaire * Large choix de Framework, * Choix du fournisseur d’infrastructure * Console web offrant un contrôle étendu des applications, des services et des utilisateurs * No vendor lock-in |  |

Figure - Carte d'identité de CloudBees

#### Heroku de salesforces

Heroku a été fondée en  2007 rachetée en 2010 par Salesforces. Heroku étend alors son offre  au développement Java. C’est une plateforme de d’application cloud initialement prévue pour les langages Ruby, Python, Scala et Clojure. La zone Europe est disponible depuis avril 2013 sur le Datacenter d’Amazon situé en Irlande; ce qui permet un gain de 100ms par requête par rapport à la zone US.

La philosophie de Heroku, c’est d’augmenter la productivité du développeur en minimisant le nombre d’étapes, en le permettant de passer moins de temps dans la gestion des serveurs d’applications, le déploiement ou la configuration de la scalabilité. Heroku est une plate-forme PaaS pour construire, déployer et exécuter des applications sur cloud. L’architecture de la plate-forme inclut des outils pour le déploiement et la gestion (plugin et SDK pour l'exécution en ligne de commande), un environnement d'exécution ainsi que des modules complémentaires.

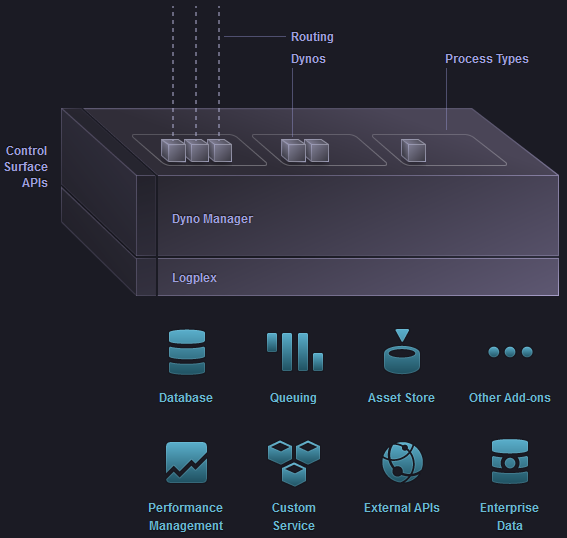


Figure - Présentation d'Heroku

|  |  |
| --- | --- |
| Type de cloud | Fournisseurs d’infrastructure |
| Publique | AWS |
| Fonctionnalités | **Interfaces** |
| * Déploiement * Connexion aux services * Contrôle de l’application et des ressources * Collaboration * Journalisation des évènements de l’application, de la plateforme, des services… * Configuration de la scalabilité | * CLI * Plugin IDE * Console web * API REST full |
| Frameworks | **Services complémentaires** |
| Rails, Express, Play, Sinatra, Spring, Jetty, Django, Flask | Base de données, outils de recherche, mail et SMS, worker et queuing, analyse de données, analyse de performance, caching, outils de facturation, journalisation et autres utilitaires |
| Avantages | **Inconvénients** |
| * Un large choix de modules complémentaires ; * Possibilité d’intégration à un IDE. | * Il n’y a pas de conteneurs ou serveurs d’applications, ces dernières sont exécutées par des processus configurés dans des fichiers Procfile. * Voir les limites [\*](https://devcenter.heroku.com/articles/limits) |

Figure - Carte d'identité d'Heroku

### Prise en mains des solutions PaaS

Pour comparer les différentes offres de PaaS, nous allons développer et déployer une application ces plateformes afin de relever les différences qui existent au niveau de l’architecture de l’application, la manière de déployer et les requis. Les solutions PaaS que nous ciblons sont : CloudFoundry, CloudBees, Heroku et OpenShift Online.

#### L’application

Dans le but de gagner en temps et de mieux se consacrer sur la comparaison des plateformes PaaS que sur le développement de l’application, nous pouvons utiliser l’IDE NetBeans pour générer une application Java EE 6 avec JSF et Hibernate JPA. En voici les spécifications :

* L’application est développée sous le modèle MVC.
* Nous avons une base de données gérées par MySQL.
* Au niveau du Modèle, les classes entités sont gérées par des contrôleurs JPA. Ces entités ne contiennent aucune annotation Hibernate. Nous avons donc des fichiers de mapping qui permettent à Hibernate de faire la relation entre les objets et les tables de la base de données. Les contrôleurs JPA nous servent à assurer les opérations classiques avec la base de données : Create – Read – Update - Delete.
* Au niveau Vue, nous avons des pages au format « JSP » permettant d’introduire des librairies de tags JSF. La partie mise en forme est séparée par une feuille de style CSS externe.
* Enfin des Contrôleurs JSF sont implémentés pour la gestion des évènements créés par l’utilisateur.

Pour obtenir cette application ouvrir NetBeans IDE 7.3 et créer un nouveau projet à partir des exemples déjà existants. Choisir le projet s’appelant « JSF JPA CRUD ». Ensuite, il suffit d’ajouter et modifier certains fichiers pour que l’application supporte Hibernate. Nous devons donc ajouter des fichiers hibernate.xml, entite.xhm.xml. Avec cette génération de projet, nous avons en même temps : les modèles, les vues ainsi que les contrôleurs prêts à l’utilisation.

#### Le déploiement sur les plateformes

La manière de déployer une application varie d’une plateforme à une autre. Les outils mis à disposition sont également différents et multiple. Dans notre cas, nous allons utiliser l’IDE Eclipse accompagné des plugins respectifs à chaque plateforme PaaS.

**N.B** :

Avant toute utilisation de la plateforme, créer comptes utilisateur sur le site dédié.

* **Déploiement sur OpenShift Online**

Pour créer le projet, utiliser le plugin OpenShift inclut dans JBoss Tool, le type d’application est jbossews-2-0 qui correspond à une application qui sera déployée sur Tomcat 7. Dans le fichier pom.xml, faire attention avec la version javax.el-api.jar utilisée. Contrairement aux autres déploiements qui vont suivre, l’emplacement du fichier hibernate.cfg.xml est « src/ressources».

* **Déploiement sur CloudFoundry**

Pour déployer notre application sur CloudFoundry, nous avons besoins de l’IDE Eclipse et du plugin CloudFoundry installable à partir d’[Eclipse Marketplace](http://marketplace.eclipse.org/). Tout d’abord, créer un projet « Dynamic Web Project » version 2.5 **(CloudFoundry ne supporte pas encore à ce jour la version 3.0 des projets web dynamique)**. Sur cette plateforme, nous avons un serveur Tomcat 6.0.35. Avec cette plateforme, le développeur peut se passer d’un fichier pom.xml. Il lui suffit juste d’un fichier .war ou de glisser-déposer sur le serveur CloudFoundry dans la vue Serveur de l’IDE. L’accès à la base de données est très simple, qu’il s’agisse d’une connexion distante ou non.

* **Déploiement sur CloudBees**

Sur CloudBees, nous avons besoins de l’IDE Eclipse et du plugin CloudBees. Créer un projet à partir du plugin, ajouter les natures JSF, etc. Faire attention, car dans le projet généré, le dossier de contenu web s’appelle « webapp » et non « webContent ». Dans le fichier hibernate.cfg.xml, la documentation du site de CloudBees donne le chemin *«*java:com/env/jdbc/mydb » au lieu de « java:**/**com/env/jdbc/mydb», ce qui empêche de Hibernate de trouver la source de données. Une mise à jour de la documentation CloudBees s’impose alors ! Pour cette plateforme, nous remarquons un fichier en plus à prendre en compte : cloudbees.xml qui sert à construire l’application.

* **Déploiement sur Heroku**

Avec le plugin Heroku, créer une application de type « *Embedded Jetty-Servlet application »*. Heroku ne possède pas de conteneur d’application. Il se sert d’un fichier Procfile pour lancer les applications. A chaque type d’application correspond une commande avec ses paramètres à exécuter. Dans le cas de notre application, nous devons utiliser Jetty et lui indiquer le port ainsi que .war. Le contenu du fichier Procfile est :

Web: java $JAVA\_OPTS -jar target/dependency/jetty-runner.jar --port $PORT target/\*.war

La connexion distante à la base de données sur Heroku nécessite l’utilisation d’une connexion SSL. L’interface web H*erokuPostgres* est très pratique, car il permet d’avoir des informations de connexion pour différents langages en même temps.

### Cas d’usages du développement sur cloud

Dans cette partie, nous allons déterminer les différents cas d’utilisations possibles dans le cycle de vie d’un projet informatique. Pour cela, nous pouvons nous aider de la figure suivante.

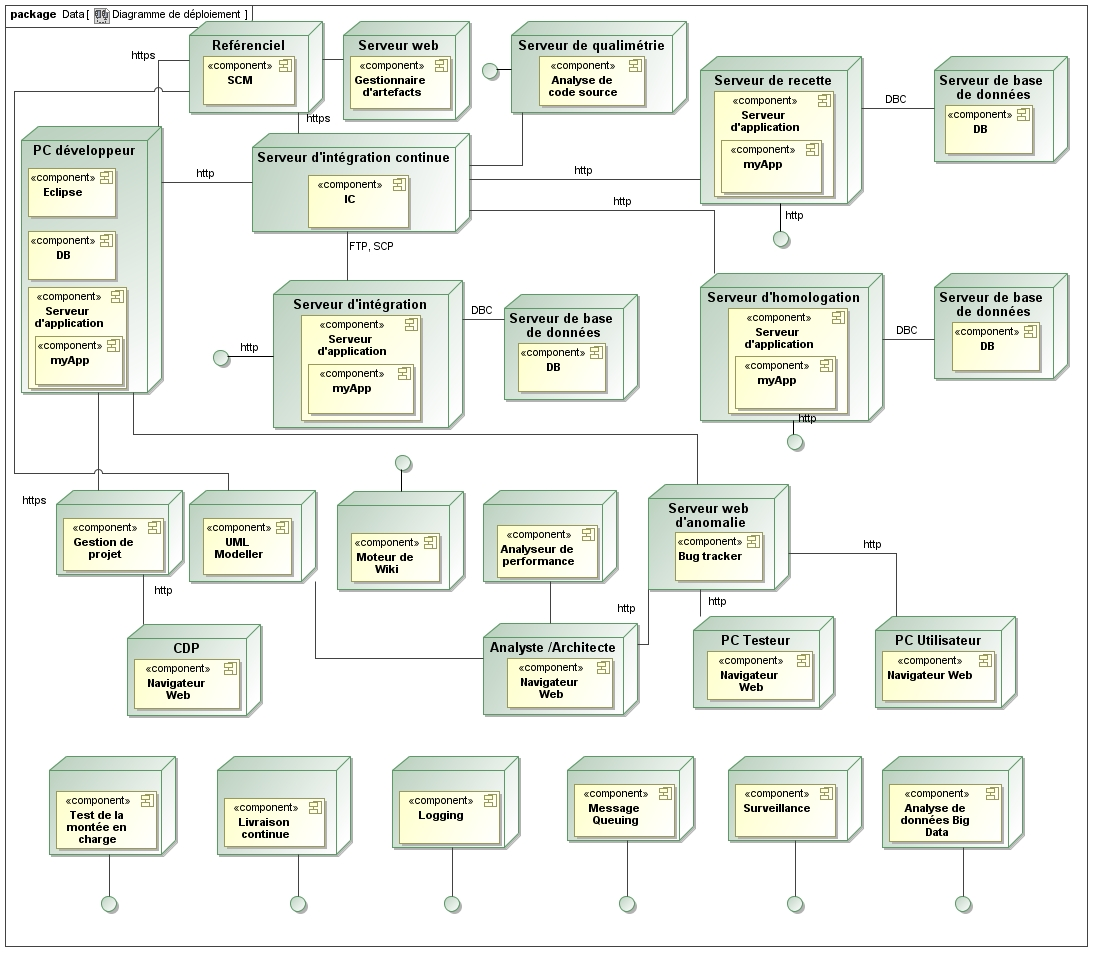
Avec ce schéma, nous pouvons distinguer une vingtaine de cas d’usages à partir duquel nous pouvons comparer les solutions PaaS en déterminant si un cas est réalisé ou pas, ou s’il est réalisé à l’aide d’un partenaire.

Figure - Vue projet générale

Figure - Vue projet

Dans les diagrammes qui vont suivre : un composant coloré en rouge signifie que le service relatif n’est pas pris en compte, un composant coloré en orange signifie que le service est offert mais seulement avec l’intermédiaire d’un partenaire et les composant en vert représentent les services en natif à la plateforme PaaS.

### Comparaison des solutions PaaS

#### Comparaison en fonction des services et cas d’usages

Visualisons ici sous formes de diagrammes de déploiements, quelques services offerts par les plateformes.

* **CloudFoundry**

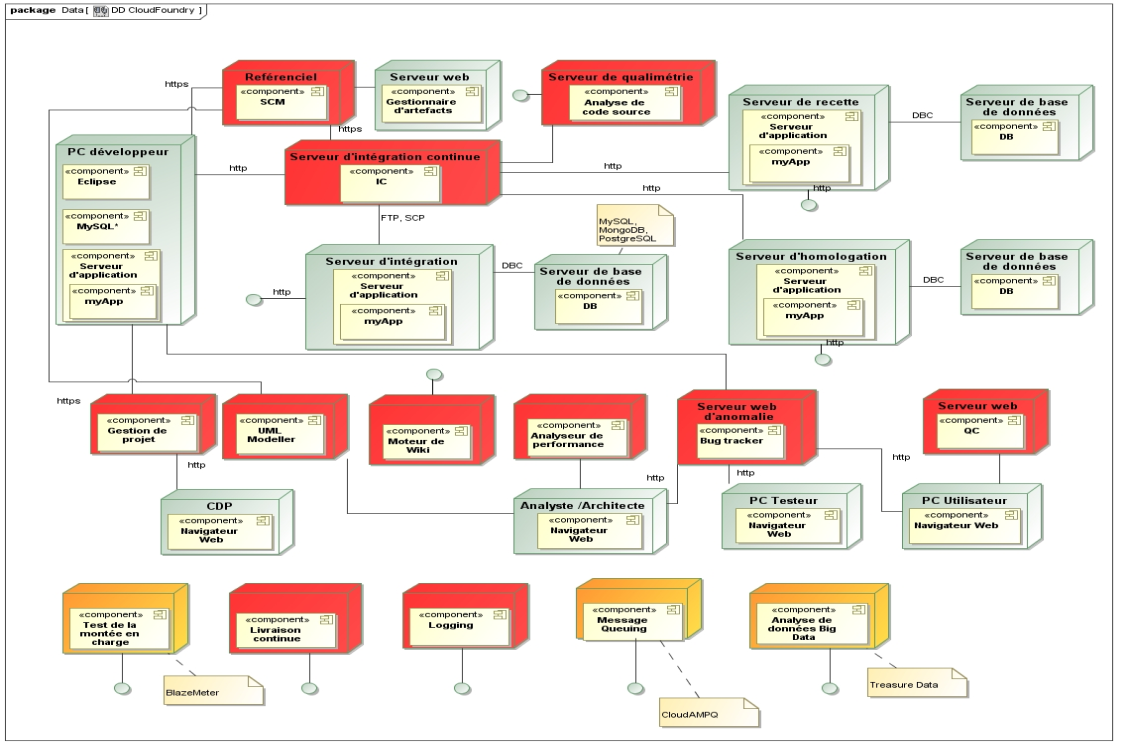
Commençons par la plateforme CloudFoundry. Cette plateforme complètement Open Source ne dispose que du minium pour mettre en place un environnement de développement ; c’est-à-dire : d’un serveur d’application et d’un gestionnaire de base de données et de quelques services offerts par les partenaires (Message Queuing, Analyse de données Big Data et test de la montée en charge). Remarquons que cette plateforme, ne fournis pas de service d’intégration continue ni de gestionnaire de code source. C’est une plateforme pour le moment destinée au développeur voulant tester son application en ligne.

Figure - Vue projet avec CloudFoundry

* **Heroku**

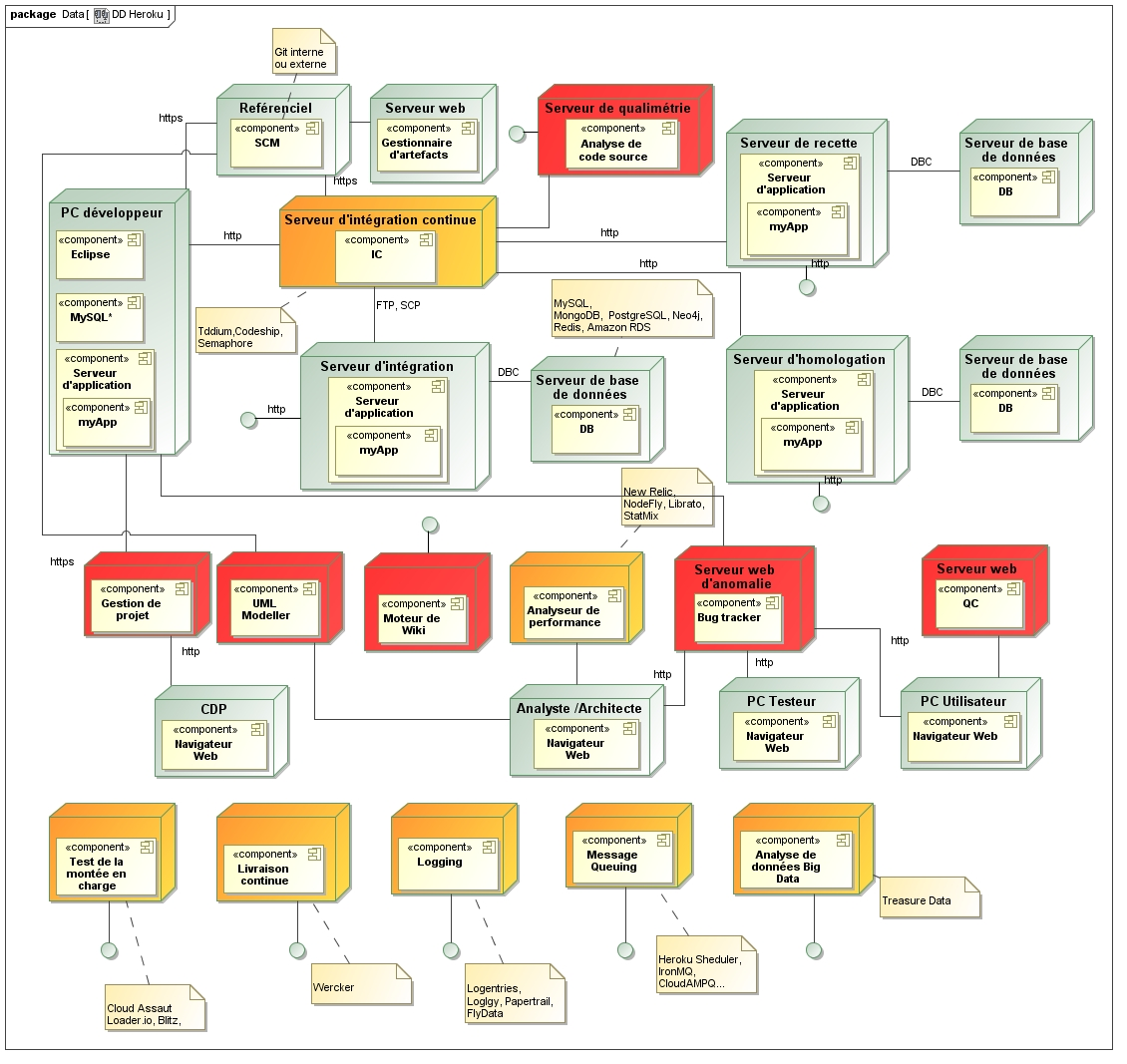
Heroku est la plateforme offrant le plus grand nombre de services complémentaires même si ceux-ci ne couvrent pas la totalité des types de services. Nous retrouvons entre autres l’intégration continue, l’analyse de données Big Data, le test de la montée en charge, l’analyse de performance…

Figure - Vue projet avec Heroku

* **CloudBees**

Parmi toutes les plateformes étudiées, CloudBees est celle qui offre les services les plus diversifiés.

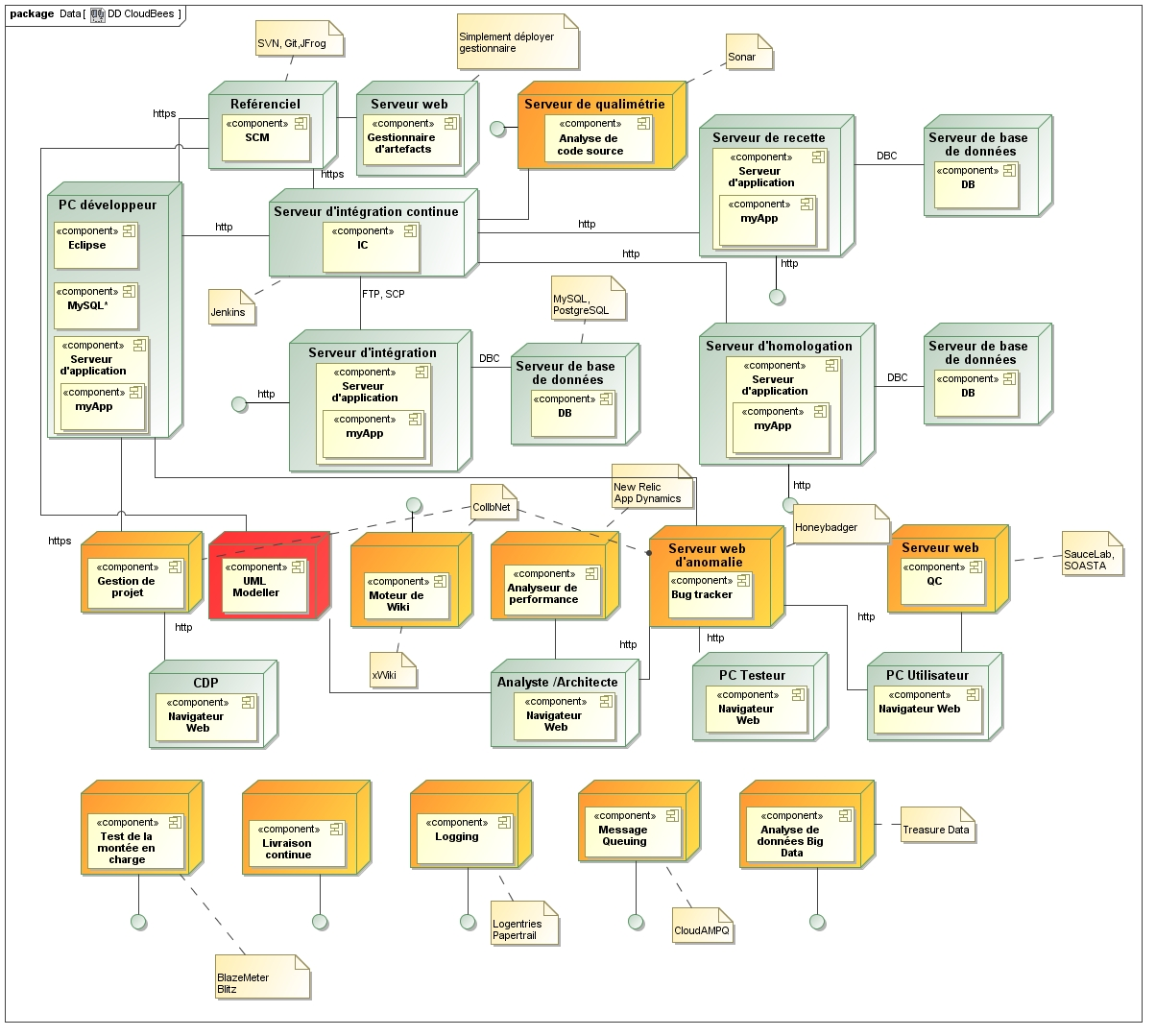


Figure - Vue projet avec CloudBees

* **OpenShift Online**

L’offre de Red Hat est principalement tournée vers le Java. Les services offerts sont en grand manques. Par exemple, il est impossible de partager ses connaissances (avec un moteur de Wiki).

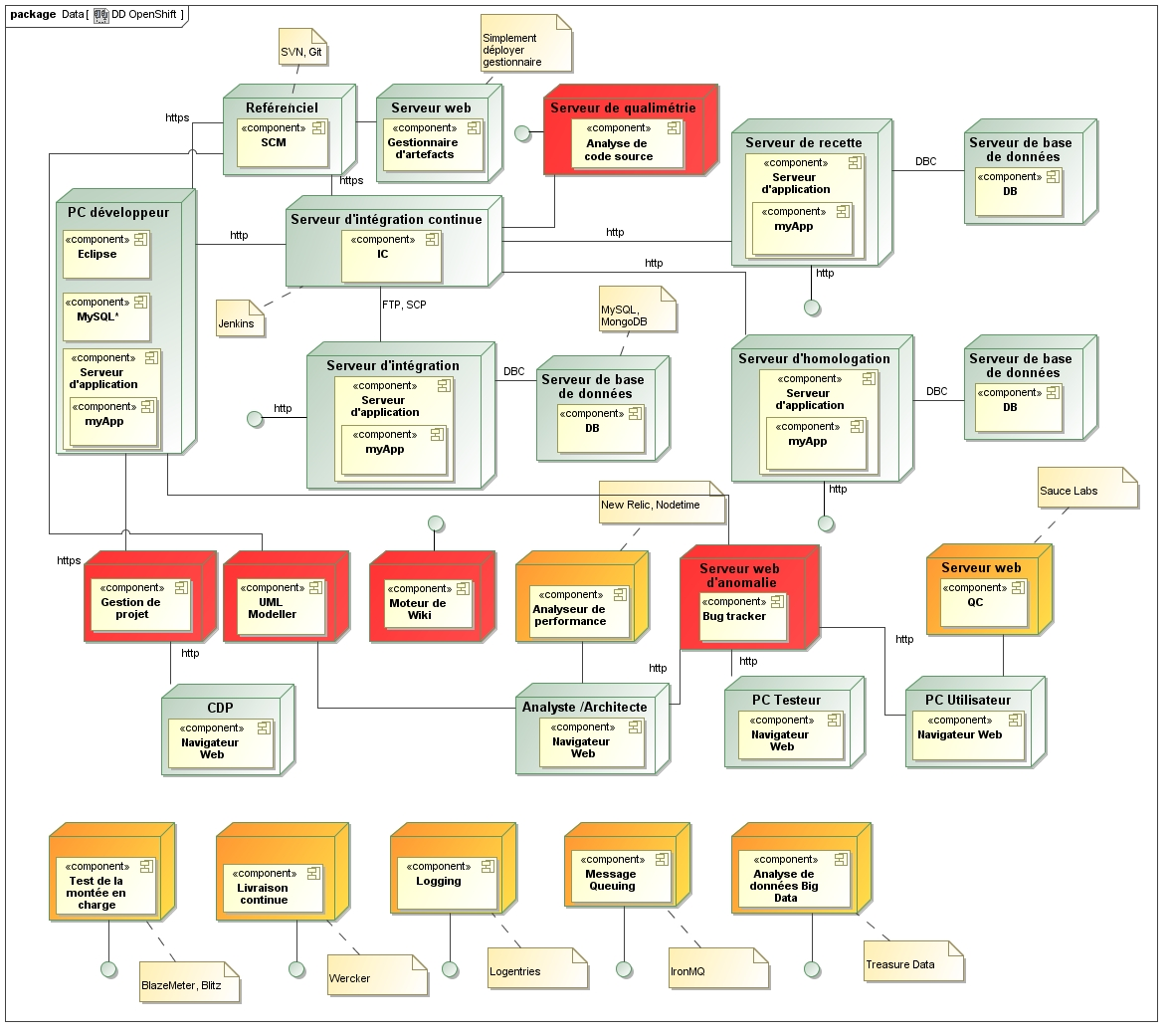


Figure - Vue projet avec OpenShift Online

#### Comparaison en fonction des Frameworks supportés

|  |  |  |
| --- | --- | --- |
| Nom | Entreprise | Langages et Frameworks |
| CloudFoundry de Pivotal | PIVOTAL | Ruby, Rails, Grails, Java Web, Lift, Node.js, Play, Rack, Sinatra, Spring, Standalone |
| CloudBees | CloudBees | Java, Spring, JRuby, Grails, Scala, Groovy, Tomcat 7, Hibernate, Java EE 6 Web Profile, Glassfish, Jetty, JBoss 7, Akka, Liferay, Grails, iU Web Apps, SOASTA CloudTest, MacOS executors, Google Go Webapp, Facebook App, Drupal, Play, Lift, Webmachine, Dart App, Clojure/Compojure, Node.js, Metor.js, Angular.js, Backbone.js, JAX RS, JRuby on Rails, Railo |
| Heroku | Salesforces | Ruby, Java, Python, Clojure, Scala, Node.js, Rails, Express, Play, Sinatra, Spring, Jetty, Django, Flask |
| Amazon Web Services EC 2 | Amazon Web Services | Java, Js, PHP, Python, Ruby, .NET, Mobile (Android et iOS) |
| Windows Azure | Microsoft | Mobile (Wind. Phone 8, iOS, Android), .NET, Node.js, Java, PHP, Python, Ruby |
| OpenShift | Red Hat | Java, Ruby, Node.js, Python, PHP, Perl, JBoss, Spring, Tomcat, Zend, Codelgniter, Rails, Node.js, Django, Flask, Perl, DIY (Do-It-Yourself) |
| Engine Yard | Engine Yard | PHP, Js, Ruby (Grappe, Rails, Sinatra) |
| AppHarbor | AppHarbor | .NET |
| CloudControl | CloudControl | Java, PHP, Node.js, Ruby, python, Symphony, Code Igniter, Flask, CakePHP, Django, Zend Framework, Joomla, Rails, Sinatra, Java, Python |
| sTATIC | STATIC | Spring, ZendPHP, Symphony, Java, Play, Lift, Python, Grails, Rack, Sinatra, Node.js, Tomcat, Django, Flask, Express, WordPress, Drupal, Joomla, Bootstrap  FuelPHP, Grails, Kohana, Lavarel, Rails, Spring, Symphony, WSGI |

Figure - Comparaison en fonction des Frameworks

#### Comparaison en fonction des en fonction des prix

Lors de la comparaison en fonction des prix, la difficulté que l’on rencontre est que les offre diffèrent par les capacités en mémoire, nombre de processeur qui calcul en même temps, services activés etc. D’où une complexité de la comparaison en fonction des prix.

|  |  |
| --- | --- |
| Nom | Prix |
| CloudFoundry de Pivotal | 0,03$/h/Go  Le coût total = mémoire \* durée \* nombre d’instances d’application \* 0,03$ [\*](http://www.cloudfoundry.com/hosted-pricing) |
| CloudBees | 0,019$/h/app-cell  [App-cell](http://www.cloudbees.com/platform/pricing/runcloud-multi/example.cb) = 128Mo [\*](http://www.cloudbees.com/platform/pricing/devcloud.cb) |
| Heroku | 0,05$/h/dyno  Dyno = 521Mo [\*](https://www.heroku.com/pricing#1-0) |
| Amazon Web Services EC 2 | http://aws.amazon.com/fr/ec2/pricing/ |
| Windows Azure | 0,0149€/h/A0  \_A0 = 768Mo [\*](http://www.windowsazure.com/fr-fr/pricing/details/cloud-services/) |
| OpenShift | \_0,04$/h/s-gear  \_0,10$/h/m-gear  \_Small Gear = 512Mo  \_Medium Gear = 1Go [\*](https://www.openshift.com/products/pricing) |
| Engine Yard | Le prix dépend de plusieurs facteurs : nombre et taille des serveurs, support technique, bande passante, nombre de backups. [\*](https://www.engineyard.com/products/cloud/pricing) |
| AppHarbor | https://appharbor.com/pricing |
| CloudControl | 0,01$/h/128Mo [\*](https://www.cloudcontrol.com/pricing) |
| sTATIC | http://www.static.com/pricing/cloud-hosting |

Figure - Prix des offres

Grace à ces différents diagrammes de déploiement et tableaux récapitulatifs montrant de façon synthétique les différentes plateformes, un consultant d’ANTEO ou autre décideur pourrait rapidement choisir une plateforme par rapport aux besoins et contraintes.

## Documents Livrables comparatif

Plusieurs documents livrables ont été produits. Nous allons dans les parties suivantes décrire ces livrables.

### Document de comparaison des PaaS

Ce document est le plus important de tous. Il présente les concepts des différents fournisseurs PaaS et compare les offres selon plusieurs critères. Il est introduit par une mise dans le contexte dans le monde du cloud dans le but de permettre au lecteur de comprendre ou se situer dans la hiérarchie du cloud (il s’agit dans notre cas d’étude de la seule couche PaaS). Cette phase de contextualisation est très importante pour de nombreuses raisons :

* elle permet de cibler les offres cloud de la couche « PaaS » des fournisseurs;
* un utilisateur pourrait se positionner sur la couche IaaS par exemple et décider de créer un environnement de développement.

Sans cette phase de cadrage, nous pourrions nous retrouver, pour cause d’incompréhension à comparer des offres SaaS, PaaS et IaaS. Par la même occasion, nous décrivons les caractéristiques qu’une offre cloud devrait avoir.

Après cette introduction, nous venons à une phase de présentations de fournisseurs tels que Windows Azure ou CloudFoundry, dans laquelle est réalisée de façon synthétique une carte d’identité de la plateforme. Dans cette carte se trouve :

* Le nom du propriétaire de la solution PaaS ;
* Le nom du fournisseur de l’infrastructure sur laquelle est construite la solution ;
* Le type de cloud : publique, privé, hybride ou communautaire ;
* Les fonctionnalités offertes ;
* Les services offerts (y compris services provenant des partenaires) ;
* Les Frameworks pouvant être utilisé dans l’environnement ;
* Les avantages et inconvénients de la plateforme ;
* Les interfaces mis à disposition des utilisateurs pour permettre d’interagir avec la plateforme : ligne de commande, console web, plugin d’IDE.

Dans la troisième partie, nous abordons le comparatif avec plusieurs matrices différentes.

* **La première matrice** permet de comparer en fonction de caractéristiques générales.

Nous décrivons dans le tableau suivant les champs utilisés.

|  |  |
| --- | --- |
| Fournisseurs | Le client de la solution PaaS pourrait par exemple avoir besoin de savoir à qui appartient la plateforme pour savoir s’il pourrait ou non faire un partenariat |
| Type de logiciel | Le type de licence est très important pour prévoir si le client pourrait ou non installer et modifier la plateforme en interne dans ses propres infrastructures. |
| Système d’exploitation |  |
| Datacenter | La localisation géographique des infrastructures pour permettre de choisir le fournisseur PaaS avec qui le client pourrait avoir le meilleur temps de réponse de son application |
| Type de cloud | Les différents types de cloud ont des niveaux de sécurités de données différents, des temps d’accès aux données différents, des prix différents… |
| Fourniture Self-Service | Détermine si le client peut lui-même choisir les services et fonctionnalités qu’il veut en n’importe quelle quantité et à n’importe quel moment. |
| Disponibilité du système | Disponibilités de la plateforme dans un mois en tenant compte des pannes et temps de maintenances. |
| Sécurité des données | Décrit la politiques de sécurité des données adoptées par la plateforme. |
| Applications privées | Détermine si une application est par défaut accessible ou non au publique |
| Environnement | Plusieurs environnements sont indispensables dans le cycle de vie d’une application : développement, test, production… |
| Interopérabilité | Le client peut vouloir que la plateforme puisse interagir avec son système interne ou un autre service cloud. |
| Applications portables | Si une application est portable, alors elle peut être développée en partie sur une plateforme différente que celle sur laquelle elle a été commencée |
| Polyglotte | Plusieurs langages de programmation ou un peuvent être supportés par une plateforme. |
| Mode de scalabilité | Scalabilité automatique ou manuelle réglable par l’utilisateur |
| Load Balancing | Une montée en charge de l’application doit s’accompagner d’une adaptation de la répartition des charges de la plateforme pour que la qualité du service ne soit pas altérée. |
| Politique en cas de panne | Ce que fait la plateforme lorsqu’elle tombe en panne : redémarrage des applications, pertes des données de sessions… |
| Plateforme mobile avec support d’application | La montée en nombre d’utilisation des appareils portables rend également important le développement sur ces appareils de nouvelles technologies. Il est donc important (mais pas nécessaires) d’avoir une plateforme fournissant des services pour mobile et une documentation. |
| Outils d’interaction | Outils mis à disposition des utilisateurs PaaS pour interagir avec la plateforme. |
| Services complémentaires | Pour savoir si des services sont offerts en plus par les partenaires. |
| Méthode de reporting | Avec certaines PaaS, l’utilisateur peut visualiser sa consommation et générer des rapports. |
| Méthodes de Gestion des ressources | Nous dis si les ressources peuvent être gérées par telle ou autre interface. |
| Prix | Il est intéressant d’avoir une vue simplifiée des prix des différentes offres, même si |
| Compte gratuit | Certaines PaaS permettent d’ouvrir des comptes utilisateur avec des ressources limitées ou des mettent à disposition des versions d’évaluations. |
| Support technique | La qualité et la disponibilité du support technique est également important pour le développeur ou gestionnaire de l’application. |

Figure - Champ du 1er tableau de comparaison

* **La deuxième matrice** permet de voir rapidement les langages est Frameworks supportés par une plateforme. En voici une partie :

|  |  |  |
| --- | --- | --- |
| Frameworks | Fournisseur Cloud PaaS | |
| Heroku | CloudFoundry |
| Spring |  |  |
| Java EE 6 Web Profile |  |  |
| Glass Fish App Server |  |  |
| Jetty |  |  |
| Zend PHP |  |  |
| Symphony |  |  |
| .NET |  |  |
| ASP. |  |  |
| Java |  |  |
| Play 2.0 |  |  |

Figure - Frameworks supportés

* + Frameworks supporté
* Frameworks non supporté
* **Deux dernières matrices** montrent de façon synthétique les cas d’utilisation réalisés par les différentes plateformes.

### Présentation des PaaS leader

Des documents sous format PowerPoint ont été réalisés dans le but de présenter au participant aux réunions hebdomadaire les différentes plateformes. Au total, trois plateformes sont ciblées pour ces présentations. Il s’agit de : CloudBees, OpenShift et Heroku. En voici le plan des présentations :

* Présentation : date de création, fournisseur, objectif
* Fonctionnalités
* Langages et Frameworks supportés
* Services et technologies complémentaires
* Prix
* Utilisation : type d’utilisateur et interfaces

### Compte rendu de semaine

Durant les premiers mois, à chaque réunion hebdomadaire, j’ai dû réaliser une présentation PowerPoint retraçant le travail effectué depuis la semaine précédente, le travail en cours les difficultés rencontrées ainsi que les prochaines étapes à suivre.

## Développement de l’application Swift sur OpenShift

### Description de l’application développée

Swift est une application permettant de piloter et d’accompagner le changement. La conduite de changement vise à maitriser le processus de transformation de l’entreprise dans un contexte de changement. Elle est composée de plusieurs phases représentée par des « Dossiers » dans l’application Swift. Dans mon cas, c’est le dossier de « Cadrage » que j’ai dû développer. Le dossier de cadrage se résume en six écrans différents permettant de modifier et afficher des informations liées au cadrage du projet.

### Méthodes de développement

Pour développer l’application Swift, nous avons utilisé un outil de gestion de cycle de vie de projet : Redmine. Le principe était alors de diviser le travail en plusieurs étapes, elles-mêmes subdivisé en action pour mieux organiser son travail. Redmine permet également aux différents tuteurs de stage de garder un œil sur le travail du stagiaire et de mieux le suivre.

Pour développer l’application, nous avons mon tuteur et moi commencer par une phase de conception avec un diagramme de classes. Ensuite, le choix me revenait pour décider des technologies et la plateforme PaaS à utiliser pour implémenter l’application et la déployer sur le cloud.

Ainsi, je disposais d’un environnement de développement local, et ce n’est qu’après avoir passé toutes les étapes de construction du projet que je le publiais sur la plateforme PaaS OpenShift.

Etapes de la conception de l’application :

* Réalisation du diagramme de classe avec Power AMC
* Génération des classes Java à partir du modèle
* Génération des tables de la base de données avec Hibernate
* Génération des contrôleurs JPA et JSF grâce à l’IDE NetBeans
* Implémentation des vues.

### Caractéristiques techniques

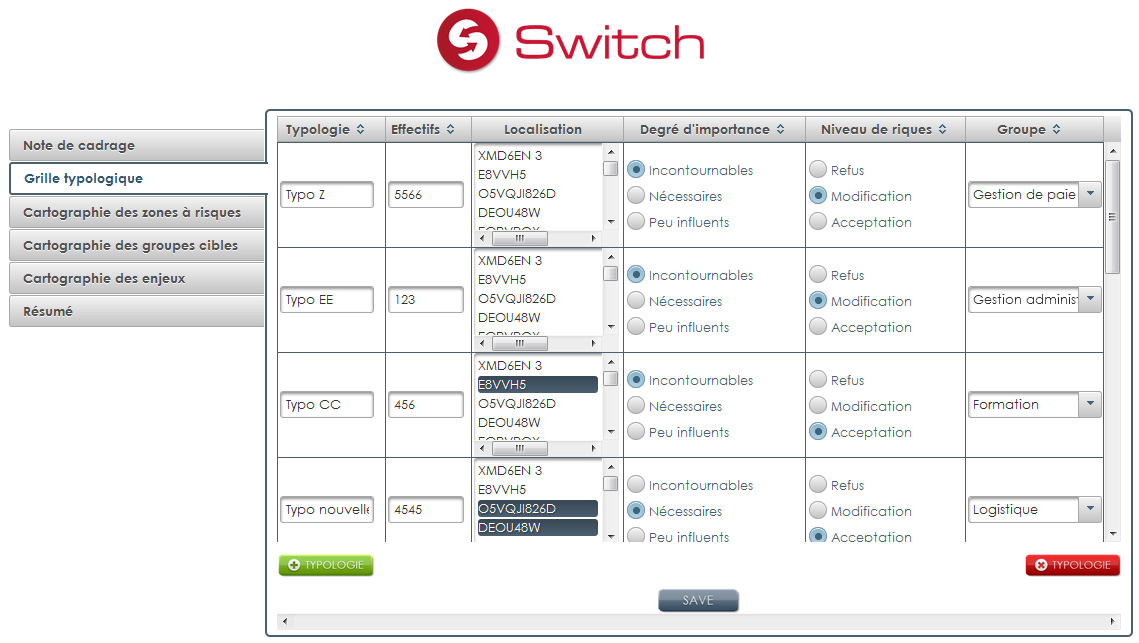
Sachant que j’avais le choix des technologies à utiliser, j’ai décidé d’adopter le modèle MVC pour que l’application soit plus maintenable et évolutive.

* Le modèle est constitué des classes entités sans annotations mais accompagnées de fichier de mapping « hbm.xml » permettant à Hibernate d’assurer la relation entre la base de données et les entités de la couche métiers. Dans cette même couche, nous retrouvons des classes permettant d’assurer les opérations avec la base de données
* Les vues sont des pages « XHTML » avec des composants JSF riches. L’implémentation JSF utilisée ici est Primefaces qui est la plus utilisée et la plus populaires.
* Les contrôleurs JSF : interceptant les actions utilisateurs et traitant les évènements.

### Réalisations

Les écrans :

* La note de cadrage : composé de champs textes.
* La grille typologique : liste les typologies et permet de les ordonner en fonction du nom et de l’effectif. Cet écran comporte le plus de composant riches Primefaces.



* La cartographie des zones à risques : montrant sous forme de graphe les différents services rangés en fonction de leur niveau de risque et du groupe.
* La cartographie des groupes cibles.
* Cartographie des enjeux.
* Le résumé : permettant de générer un rapport au format PDF

## Connaissances acquises

Au cours du stage, des connaissances ont été accumulée dans plusieurs domaines. Ces connaissances ont été acquises au travers d’une phase de montée en compétences et pendant la phase d’études des PaaS. Nous pouvons citer parmi ces connaissances :

* L’intégration continue avec Jenkins, Spring MVC, Ruby (avec une certification obtenue de Berkeley EDX) etc.
* En outre, j’ai pu au cours des développements de Swift, maîtriser l’implémentation Primefaces.

## Recueil d’informations

Pour récolter l’information, le principal moyen était la visite du site de la Plateforme, les blogs associés et Forum. Souvent, des prises de contact par conversation en direct ou mail, avec les responsables techniques des PaaS ont été nécessaires pour obtenir l’information.

# Conclusion

Enfin, nous pouvons relever une complexité de comparaison des offres cloud PaaS car les offres varient en fonctions des ressources allouées, le type d’application à déployer, la manière de gérer le cycle de vie d’une application, les services supplémentaires offerts, le mode de scalabilité, et d’autre d’options relatives aux fournisseurs. Pour être sûr d’utiliser l’une des meilleures plateformes, à partir d’un type d’application : cibler les plateformes pouvant l’exécuter, comparer en fonction des ressources mémoire CPU consommées dans le temps et services complémentaires souhaités, et enfin, procéder à une élimination en fonction du coût. D’autres paramètres pourraient également être pris en compte pour choisir la meilleure solution PaaS. Par ailleurs, l’un des meilleurs moyens de savoir si une plateforme est bonne ou pas serait de l’essayer.

# Annexes

Tableau - Cas d'usages

|  |  |
| --- | --- |
|  | User Stories |
| 1 | En tant que développeur,  Je souhaite mettre en place un environnement de développement déterminé (serveur ou conteneur d’application, base de données, etc.) sur le cloud,  Afin de développer une application. |
| 2 | En tant que développeur,  Je souhaite utiliser environnement de développement existant sur le cloud,  Afin de participer au développement d’une application. |
| 3 | En tant que développeur,  Je souhaite mettre en place un gestionnaire de dépendances  Afin de disposer d’un dépôt de librairies partagées et de proximités |
| 4 | En tant que développeur,  Je souhaite mettre en place un environnement d’intégration continue sur le cloud,  Afin de vérifier qu’il n’y a pas de régression de la qualité d’une application à chaque modification du code source. |
| 5 | En tant que développeur,  Je souhaite analyser le code source de mon application et en obtenir les métriques de qualités,  Afin d’évaluer rapidement la dette technique de mon application et recenser les corrections à apporter |
| 6 | En tant que développeur,  Je souhaite stocker et gérer les versions de mes fichiers,  Afin de partager mes fichiers et conserver l’historique des modifications. |
| 7 | En tant que chef de projet,  Je souhaite planifier, piloter, suivre l’avancement et gérer les ressources d’un projet,  Afin de m’assurer de la conformité par rapport aux plans initiaux. |
| 8 | En tant que membre du projet,  Je souhaite mémoriser et organiser mes connaissances,  Dans le but de capitaliser et les partager avec tous mes collaborateurs. |
| 9 | En tant que testeur ou développeur,  Je souhaite mettre en place un système de suivis d’anomalies  Afin de corriger des erreurs fonctionnelles ou techniques d’une application. |
| 10 | En tant que testeur,  Je souhaite mettre en place un environnement de test (serveur ou conteneur d’application, base de données, etc.) sur le cloud,  Afin de tester une application dans un environnement déterminé (homologation, recette, pré-production). |
| 11 | En tant que développeur,  Je souhaite identifier, partager et centraliser les versions de mon logiciel binaire de ses dépendances  Afin d’éviter les conflits et les erreurs liées aux versions des logiciels et des dépendances |
| 12 | En tant CP,  Je souhaite configurer les ressources allouées sur le cloud  Afin de mieux en contrôler le coût |
| 13 | En tant qu’administrateur,  Je souhaite choisir des fournisseurs d’infrastructures différents pour mon environnement d'exécution  Afin de contrôler les coûts de consommation. |
| 14 | En tant qu’expert,  Je souhaite surveiller, diagnostiquer une application  Afin de contrôler ses performances |
| 15 | En tant qu’architecte,  Je souhaite modéliser mon système par des vues logiques, d'implémentations, de déploiements et de processus,  Afin de mieux faire comprendre et avoir une meilleure visibilité du système. |
| 16 | En tant que développeur,  Je souhaite interagir avec la plateforme cloud avec mon code source (via une API), avec un SDK (exemple de RHC) ou un IDE,  Afin d’ajouter à mon application un nouveau service. |
| 17 | En tant que décideur,  Je souhaite collecter et analyser mes données Big Data  Afin d’avoir une vue d’ensemble sur une activité et de pouvoir prendre des décisions. |
| 18 | En tant qu’administrateur,  Je souhaite sauvegarder les images de mes fichiers, base de données et applications  Afin de pouvoir les restaurer en cas de perte de données ou de défaillance du système. |
| 19 | En tant CP,  Je souhaite consulter un rapport de la consommation en ressources / être prévenu d’un seuil...  Afin de maîtriser leurs coûts |

Figure - Radar de performance de quelques PaaS

\_ Série 3 : CloudBees

\_ Série 1 : OpenShift

\_ Série 2 : Heroku

\_ Série 4 : CloudFoundry

**Echelle :**

\_ 1 : pas réalisé

\_ 2 : réalisé avec via un partenaire

\_ 3 : réalisé via un service en natif

\_ 4 : liberté en plus d'utiliser son propre outil, ou de l'intégrer, les options de services en natifs simplifient la réalisation du cas d'usage

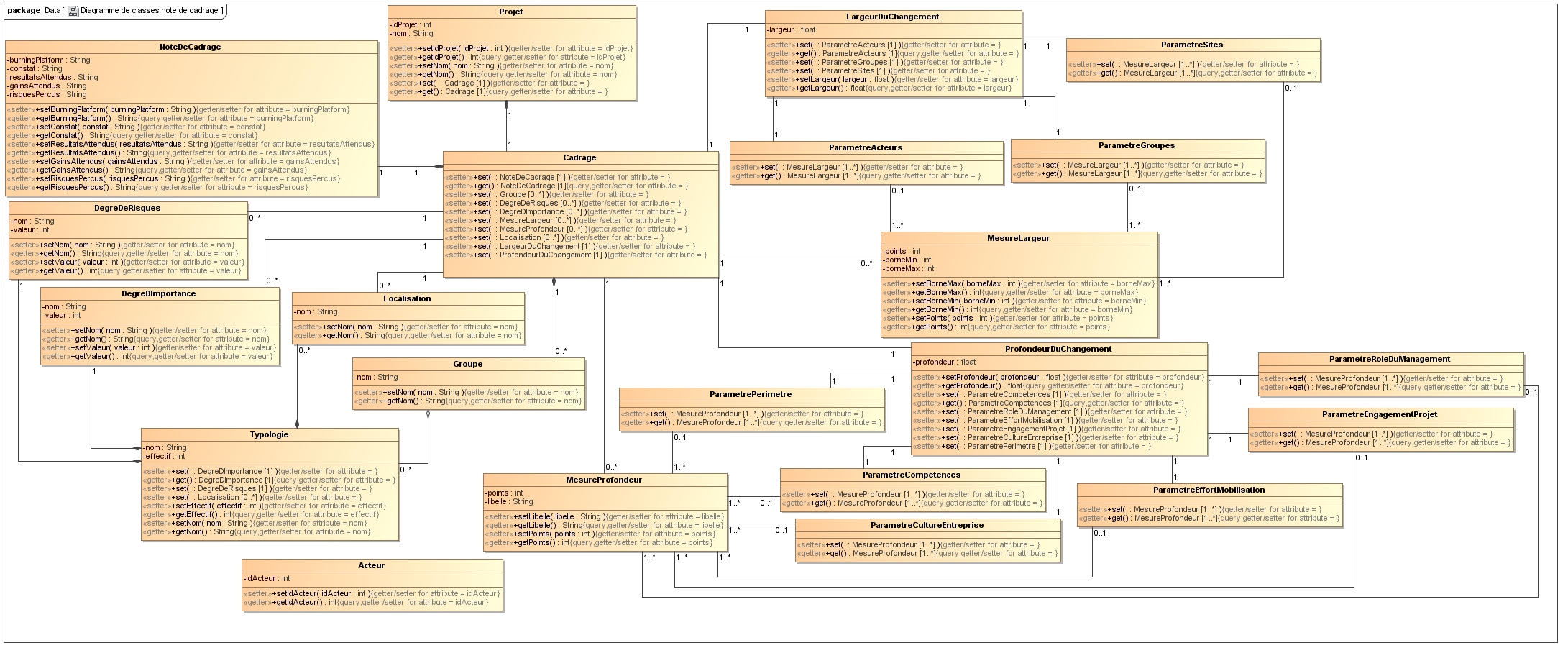


Figure - Diagramme de classes de Swift

# Tables des figures

[Figure 1 - L'offre de solutions et de services de SODIFRANCE 5](#_Toc365032938)

[Figure 2 - Hiérarchie des couches du cloud 12](#_Toc365032939)

[Figure 3 - Liste des solutions cloud PaaS étudiées 12](#_Toc365032940)

[Figure 4 - Carte d'identité de CloudFoundry 13](#_Toc365032941)

[Figure 5 - Présentation d'OpenShift 14](#_Toc365032942)

[Figure 6 - Carte d'identité d'OpenShift Online 14](#_Toc365032943)

[Figure 7 - Présentation de CloudBees 15](#_Toc365032944)

[Figure 8 - Carte d'identité de CloudBees 16](#_Toc365032945)

[Figure 9 - Présentation d'Heroku 16](#_Toc365032946)

[Figure 10 - Carte d'identité d'Heroku 17](#_Toc365032947)

[Figure 11 - Vue projet générale 20](file:///C:\Users\idiallo.stage\git\docPaaSSodifrance\Rapport%20de%20stage%20SODIFRANCE.docx#_Toc365032948)

[Figure 12 - Vue projet 20](file:///C:\Users\idiallo.stage\git\docPaaSSodifrance\Rapport%20de%20stage%20SODIFRANCE.docx#_Toc365032949)

[Figure 13 - Vue projet avec CloudFoundry 21](#_Toc365032950)

[Figure 14 - Vue projet avec Heroku 22](file:///C:\Users\idiallo.stage\git\docPaaSSodifrance\Rapport%20de%20stage%20SODIFRANCE.docx#_Toc365032951)

[Figure 15 - Vue projet avec CloudBees 23](file:///C:\Users\idiallo.stage\git\docPaaSSodifrance\Rapport%20de%20stage%20SODIFRANCE.docx#_Toc365032952)

[Figure 16 - Vue projet avec OpenShift Online 24](file:///C:\Users\idiallo.stage\git\docPaaSSodifrance\Rapport%20de%20stage%20SODIFRANCE.docx#_Toc365032953)

[Figure 17 - Comparaison en fonction des Frameworks 25](#_Toc365032954)

[Figure 18 - Prix des offres 26](#_Toc365032955)

[Figure 19 - Champ du 1er tableau de comparaison 29](#_Toc365032956)

[Figure 20 - Frameworks supportés 29](#_Toc365032957)

[Figure 21 - Radar de performance de quelques PaaS 34](#_Toc365032958)

[Figure 22 - Diagramme de classes de Swift 35](file:///C:\Users\idiallo.stage\git\docPaaSSodifrance\Rapport%20de%20stage%20SODIFRANCE.docx#_Toc365032959)

# Bibliographie

* Définition de termes liés au cloud computing

<http://france.emc.com/corporate/glossary/index.htm>

* Les modèles de cloud

<http://france.emc.com/corporate/glossary/private-cloud.htm>

<http://france.emc.com/corporate/glossary/public-cloud.htm>

<http://france.emc.com/corporate/glossary/hybrid-cloud.htm>

* Heroku

<https://devcenter.heroku.com/articles/how-heroku-works>

* CloudBees

<http://www.cloudbees.com/platform/how-it-works.cb>

* CloudFoundry

<http://docs.cloudfoundry.com/docs/running/architecture/>

* OpenShift

<https://www.openshift.com/get-started>