|  |  |  |
| --- | --- | --- |
| DIALLO Issaga Master 2 MITIC | septembre 6  2013 | |
| Université de Rennes 1 - ISITIC | | Etude et développement sur plateforme PaaS |

# Remerciements

# Résumé

# Sommaire

[Résumé 1](#_Toc364781791)

[Sommaire 1](#_Toc364781792)

[1. Introduction 2](#_Toc364781793)

[2. Présentation de SODIFRANCE 2](#_Toc364781794)

[2.1. Historique 2](#_Toc364781795)

[2.2. Organigramme 2](#_Toc364781796)

[2.3. Prestations 2](#_Toc364781797)

[2.4. Marchés et clients 2](#_Toc364781798)

[3. Présentation de l’entité ANTEO Consulting 2](#_Toc364781799)

[4. Description du stage 2](#_Toc364781800)

[4.1. Définitions 2](#_Toc364781801)

[4.1.1. Cloud computing 2](#_Toc364781802)

[4.1.2. PaaS 3](#_Toc364781803)

[4.2. Description 3](#_Toc364781804)

[4.3. Objectifs 3](#_Toc364781805)

[4.3.1. Besoins (du groupe ANTEO) 4](#_Toc364781806)

[4.3.2. Contraintes 4](#_Toc364781807)

[4.4. Enjeux des PaaS 4](#_Toc364781808)

[5. Etude et développement sur plateforme PAAS 4](#_Toc364781809)

[5.1. Etude comparative des solutions 4](#_Toc364781810)

[5.1.1. Terminologies 4](#_Toc364781811)

[5.1.2. Recensement de solutions PaaS 6](#_Toc364781812)

[5.1.3. Présentation de certains leaders fournisseurs de PaaS 7](#_Toc364781813)

[5.1.1. Prise en mains des solutions PaaS (en vue de comparaison) 10](#_Toc364781814)

[5.1.2. Cas d’usages du développement sur cloud (en utilisant la vue projet) 11](#_Toc364781815)

[5.1.3. Comparaison des solutions PaaS 11](#_Toc364781816)

[5.2. Document comparatif (méthodes d’obtention des données, mail au PaaS, etc…), présentation PPT 11](#_Toc364781817)

[5.3. Développement de l’application Swift sur OpenShift 11](#_Toc364781818)

[5.3.1. Description de l’application développée (+ besoin, …) 11](#_Toc364781819)

[5.3.2. Méthodes de développement (+ Redmine) 11](#_Toc364781820)

[5.3.3. Choix des technologies 11](#_Toc364781821)

[5.3.4. Réalisation 11](#_Toc364781822)

[5.4. Connaissances acquises 11](#_Toc364781823)

[5.5. Documents réalisés 11](#_Toc364781824)

[5.6. Recueil d’informations 11](#_Toc364781825)

[6. Conclusion 11](#_Toc364781826)

[7. Annexes 11](#_Toc364781827)

[8. Bibliographie 11](#_Toc364781828)

# 

# Introduction

# 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. Grâce à d’importants investissements en Recherche et Développement, SODIFRANCE met au point un atelier de génie logiciel basé sur des solutions outillées qui permettent l’évolution des grands systèmes d’information dans une approche automatisée et industrialisée. Cette offre devient un des principaux fers de lance de SODIFRANCE et rencontre un succès croissant auprès des grands comptes.

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 (source : Pierre Audoin Consultants).

#### Industrialisation et CMMI

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%.

## Prestations

## Marchés et clients

# Présentation de l’entité ANTEO Consulting

# 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… 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.

### 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 destinée 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 est ce que 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 besoins, contraintes et exigences définis par le groupe ANTEO.

### Besoins (du groupe ANTEO)

### Contraintes

## Enjeux des PaaS

# 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 est 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 Etat 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 .

* 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

#### Les couches principales du cloud

Le cloud est structuré en trois couches principales

##### IaaS

**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 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, et 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

**Plateforme en tant que service (PaaS)** 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 (SaaS**) 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 « Services-as-a-Software » 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.

### 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 | 2011 |
| Heroku | Salesforces | 2012 |
| Amazon Web Services EC 2 | Amazon Web Services |  |
| Windows Azure | Microsoft | 2009 |
| OpenShift | Red Hat |  |
| Engine Yard | Engine Yard |  |
| AppHarbor | AppHarbor |  |
| CloudControl | CloudControl |  |
| sTATIC | STATIC | 2013 |

### Présentation de certains leaders fournisseurs de PaaS

#### CloudFoundry de pivotal (présentation, SERVICES …)

Cloud Foundry 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 et est accessible à travers [une variété de distributions clouds](https://core.cloudfoundry.org/listings) privées incluant [CloudFoundry.com](http://cloudfoundry.com).

Cloud Foundry 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 change le code source. CloudFoundry est un socle à partir duquel d’autres fournisseurs se basent pour offrir des services PaaS plus Elargis. 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 |

#### 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.

|  |  |
| --- | --- |
| 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 |  |

#### 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.

|  |  |
| --- | --- |
| 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 |  |

#### 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.

|  |  |
| --- | --- |
| 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) |

### 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**

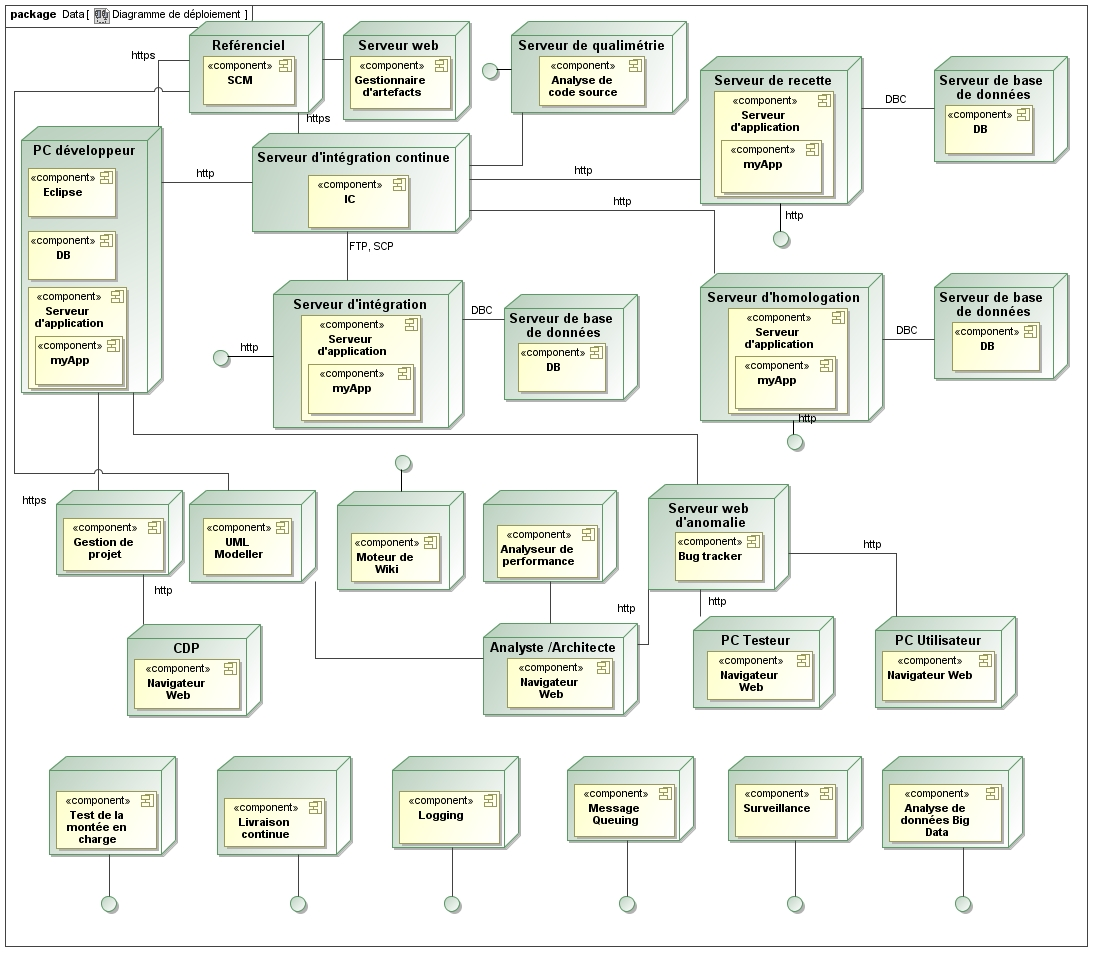
Avec le plugin Heroku, créer une application de type « *Embedded Jetty-Servlet application »*. Heroku ne possède pas de conteneur d’application. Il se sert d’un fichier Procfile pour lancer les applications. A chaque type d’application correspond une commande avec ses paramètres à exécuter. Dans le cas 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.

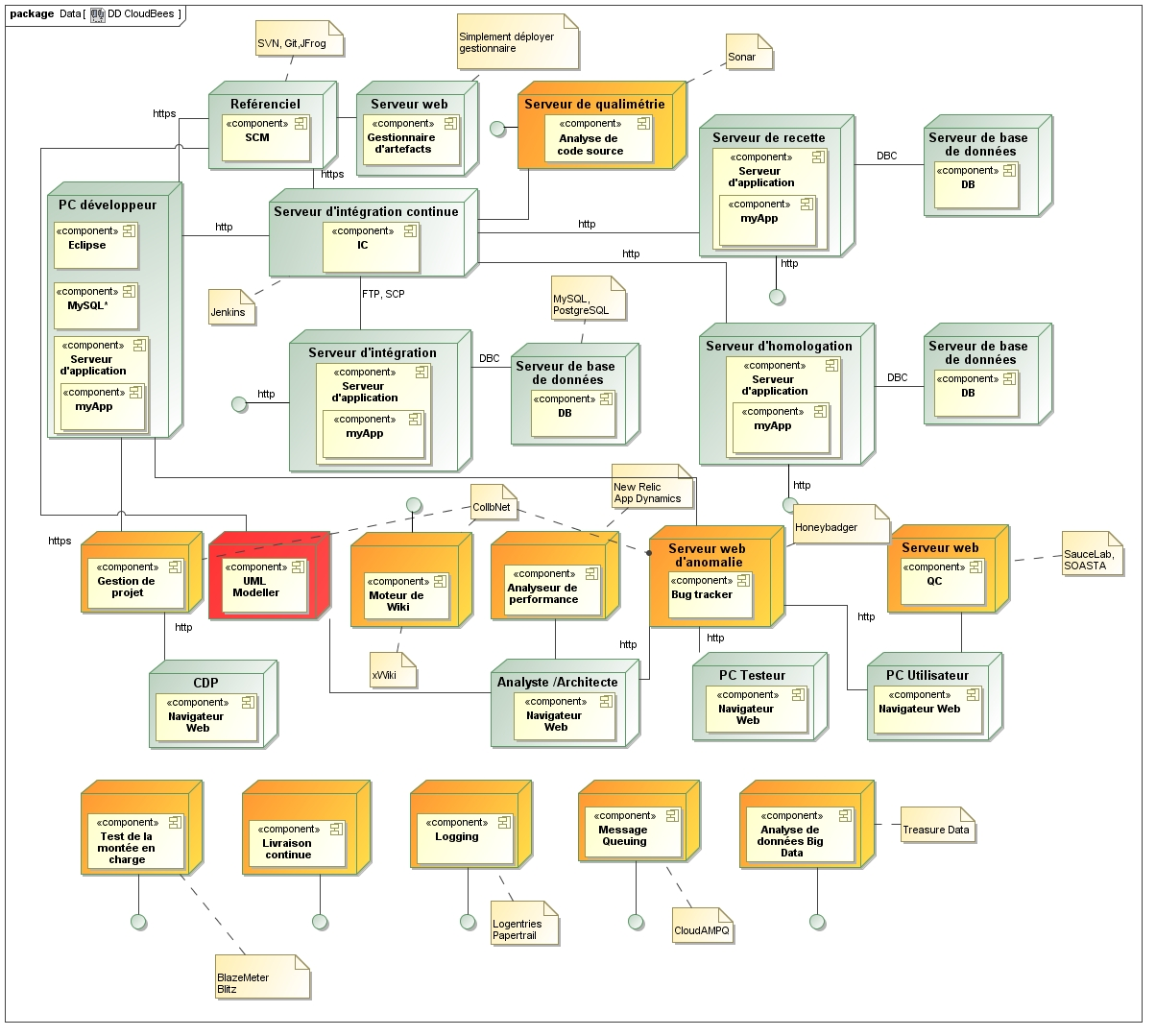
Avec ce schéma, nous pouvons distinguer une vingtaine de cas d’usages à partir duquel nous pouvons comparer les solutions PaaS en déterminant si un cas est réalisé ou pas, ou s’il est réalisé à l’aide d’un partenaire.

### Comparaison des solutions PaaS

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

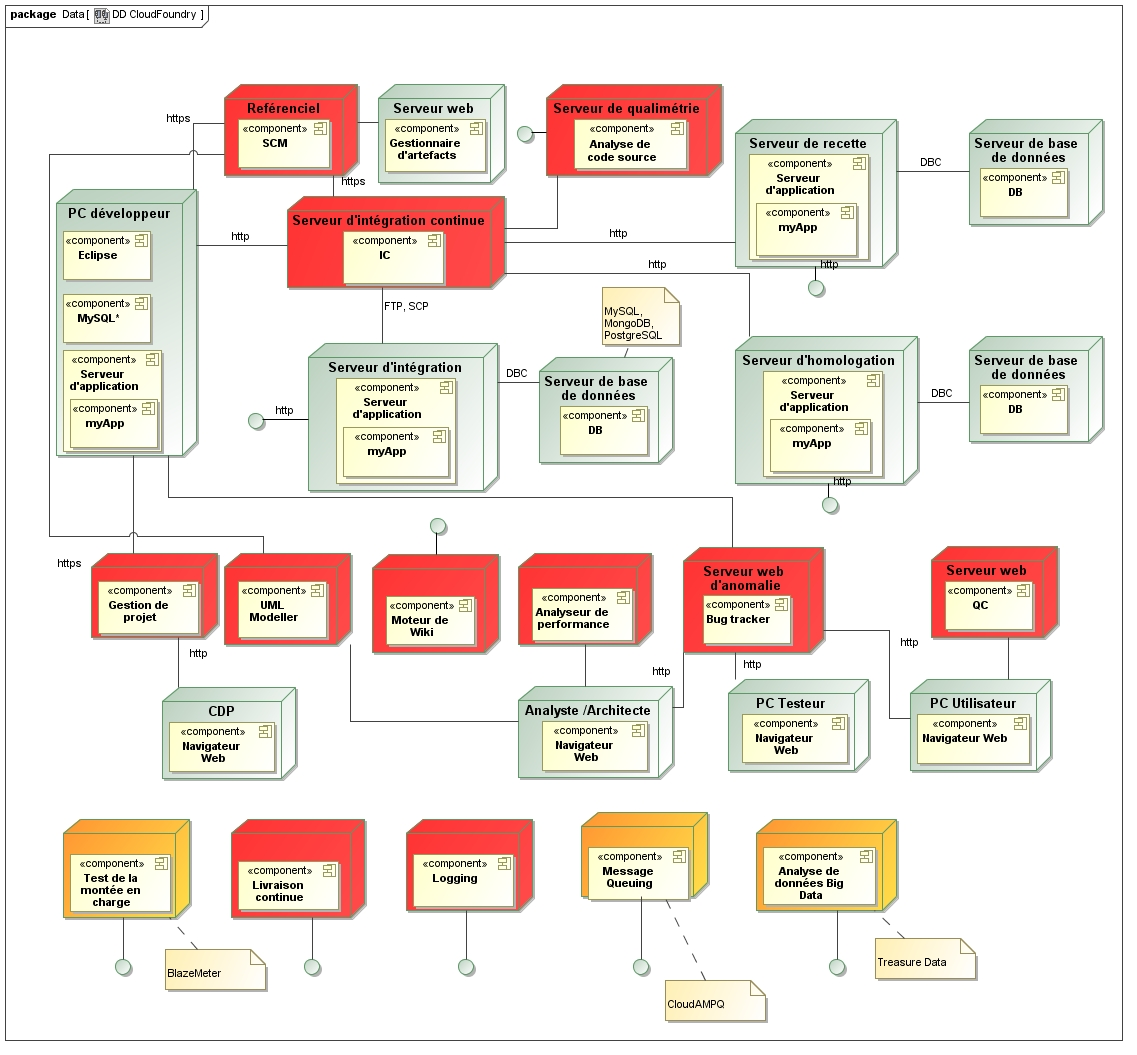
Vue projet de CloudBees.

En regardant le diagramme, nous nous rendons bien compte que la plateforme CloudBees

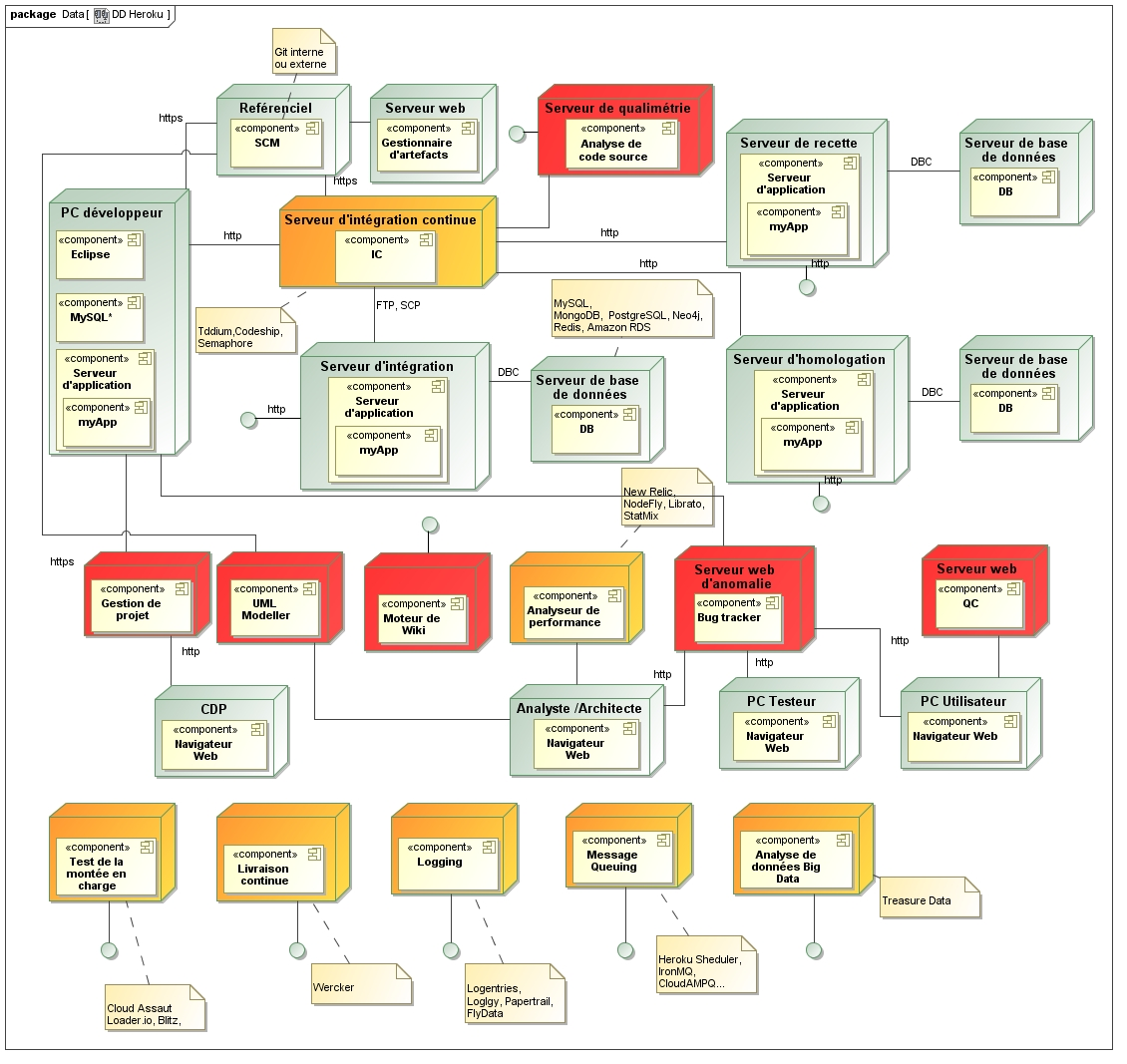


Vue projet de CloudFoundry

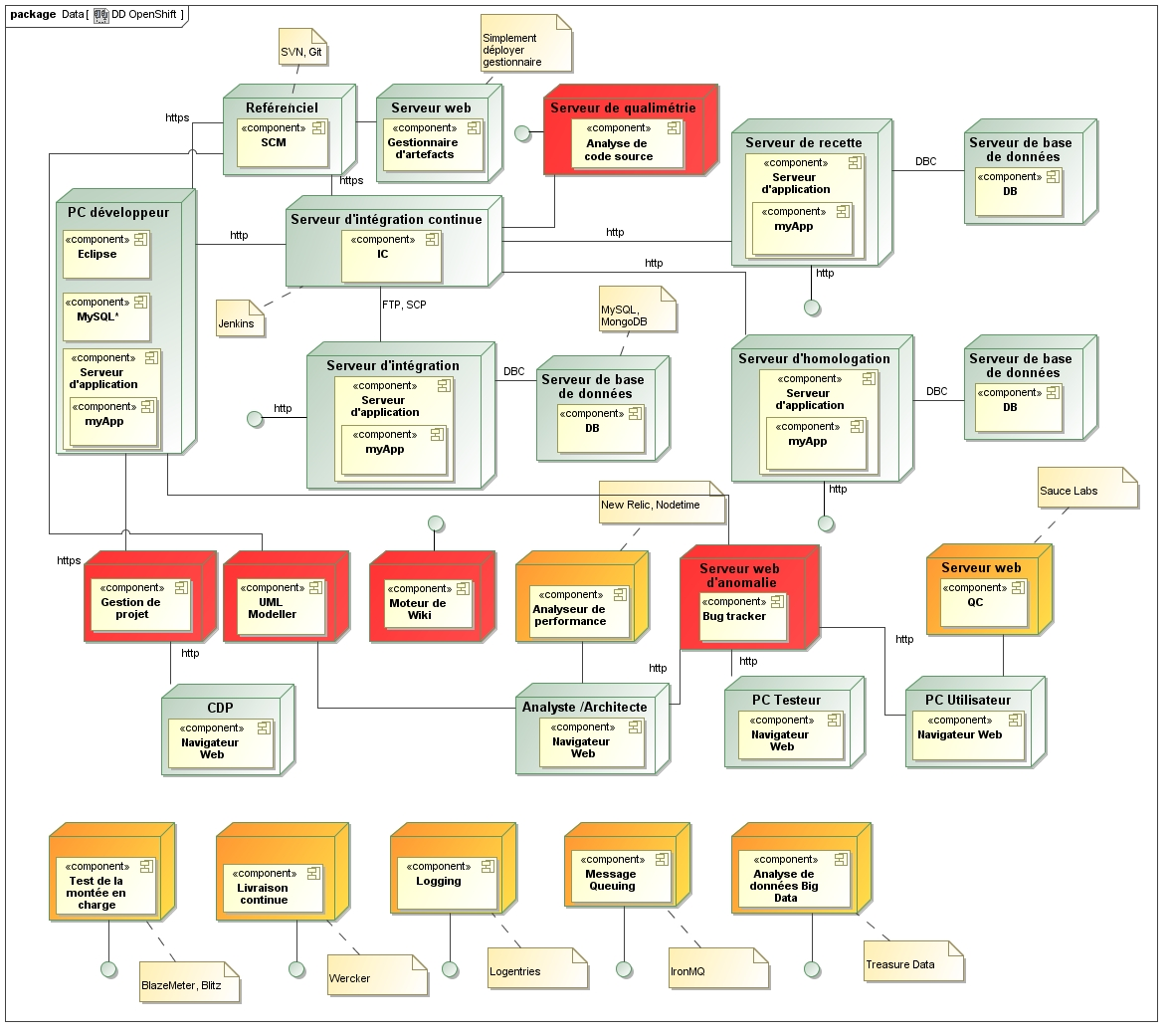
Commençons par la plateforme



Vue projet d’Heroku



Vue projet d’OpenShift



#### 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 |

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

Complexité de la comparaison en fonction des prix.

## Documents Livrables comparatif (méthodes d’obtention des données, mail au PaaS, etc…), présentation PPT

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 |  |
| Méthodes de Gestion des ressources |  |
| Prix |  |
| Compte gratuit |  |
| Support technique |  |

* La deuxième matrice
* La troisième matrice

### Présentation des PaaS leader

### Compte rendu de semaine

## Développement de l’application Swift sur OpenShift

### Description de l’application développée (+ besoin, …)

### Méthodes de développement (+ Redmine)

### Choix des technologies

### Réalisation

## Connaissances acquises

## Documents réalisés

## Recueil d’informations

# Conclusion

# Annexes

# Bibliographie

http://www.gartner.com/it-glossary/cloud-computing/