Université Abdelmalek Essaâdi Ecole Nationale des Sciences Appliquées

Al Hoceima

## PROJET DE FIN D’ETUDES

#### En vue de l’obtention du diplôme

**INGENIEUR D’ETAT**

**Filière : Génie informatique Option : Génie Logiciel**

**Développement et Déploiement d’une solution E-commerce Sécurisée intégrée avec l’ERP Odoo**

##### Réalisé par :

**Mr. Hamza HADDAA**

**Soutenu le 24/06/2019, devant le Jury :**

**Pr. Fouzia MORADI Professeur à L’ENSAH Encadrante**

**pédagogique**

**Mr. Ahmed EL HAMIDI**

**Mr. Karim EL Moutaouakil**

**Mr. Younes ABOULHANOUNE**

**Ingénieur à Orange Bytes Encadrant professionnel**

**Professeur à L’ENSAH Examinateur**

**Professeur à L’ENSAH Examinateur**

Année universitaire 2018/2019

## Dédicaces

##### Résumé

Les magasins en ligne sont depuis des années, largement recommandés aux entreprises qui comptent sur la vente des produits, voire des services. Ce types de sites Web représentent un outil mondial permettant aux clients un accès à toutes les informations, produits et services à partir d’un portail unique en rapport avec leur activité.

Dans ce cadre, on s’est engagé dans une équipe de jeunes ingénieurs compétents pour la création du projet innovant *Joodperfumes.com*, qui peut être considèré le meilleur de son genre, au sein de l’entreprise **Orange Bytes**. Le projet consiste à la conception et à la réalisation d’une plateforme E-commerce de vente des parfums dans le monde arabe .

Au cours de cette expérience, une autre de mes missions était également la création et l’automa- tisation de l’infrastructure de Jood dans le cloud.

Dans ce contexte que vient s’inscrire notre **projet de fin d’études** pour l’obtention du **diplôme Ingenieur** en **génie Informatique** à **L’Ecole Nationale des Sciences Appliquées**.

**Mots clefs :** E-Commerce, Informatique, Programmation, Développement logiciel,Devops.Cloud

##### Abstract

**Table des matières**

|  |  |
| --- | --- |
| [**Liste des figures**](#_bookmark0) |  |
| [**Liste des tableaux**](#_bookmark1) |  |
| [**Introduction générale**](#_bookmark2) |  |
| 1. [**Contexte général du projet**](#_bookmark3)   [Introduction](#_bookmark4)…………………………………………………………………………………… |  |
| [1.1 Présentation de MNS](#_bookmark5) Consulting Group………………………………………………….. |  |

* 1. [Présentation du projet](#_bookmark11) 5
  2. [Problématique du projet](#_bookmark15) 7
  3. [Solution](#_bookmark17) proposée 8
  4. [Conduite](#_bookmark20) et gestion du projet 9
  5. [Contraintes du Projet](#_bookmark23) 11
  6. [Planification du Projet](#_bookmark25) 11

[Conclusion](#_bookmark27) 12

1. [Etude fonctionnelle - technique](#_bookmark28)
   1. [Introduction](#_bookmark29) 13
   2. [Spécification des besoins](#_bookmark30) 13
   3. [Analyse fonctionnelle](#_bookmark31) 14
      1. [Identification des acteurs](#_bookmark32) 14
      2. [Identification des différents modules (applications)](#_bookmark33) 14
      3. [Diagrammes de cas d’utilisation](#_bookmark34) 15
      4. [Diagrammes de séquence](#_bookmark39) 17
      5. [Diagrammes de classes](#_bookmark43) 20
   4. [Etude technique du projet](#_bookmark46) 21
      1. [Modèles de conception adoptés](#_bookmark47) 21
      2. [Architecture Global - Applications](#_bookmark50) 22
   5. [Conclusion](#_bookmark52) 23
2. [Technologies Utilisées](#_bookmark53)24
   1. [Introduction](#_bookmark54) 24
   2. [Outils d’infrastructure Cloud](#_bookmark55) 24
   3. [Outils de développement -Backend](#_bookmark57) 25
   4. [Outils de Tests](#_bookmark61) 30
   5. [Outils de development Front-end](#_bookmark62) 31
3. [L’infrastructure cloud de joodpefumes](#_bookmark63)35
   1. [Introduction](#_bookmark64) 35
   2. [Qu’est-ce que le cloud computing](#_bookmark65) 35
      1. [Principaux avantages du cloud computing](#_bookmark66) 35
      2. [Types de cloud computing](#_bookmark67) 36
      3. [Types de services cloud : IaaS, PaaS, SaaS](#_bookmark68) 36
   3. [le choix de AWS](#_bookmark69) 37
   4. [Infrastructure en tant que code](#_bookmark71) 38
   5. [Terraform - Concepts](#_bookmark72) 38
      1. [Terraform - Fournisseur (Provider)](#_bookmark76) 39
      2. [Terraform - Variables](#_bookmark78) 40
      3. [Terraform - Modules](#_bookmark80) 40
      4. [Terraform - Outputs](#_bookmark82) 41
      5. [Terraform - Sources de données (Data Sources)](#_bookmark85) 41
      6. [Terraform - Etat (State)](#_bookmark88) 42
      7. [Terraform - CLI](#_bookmark91) 43
   6. [Infrastructure de Jood](#_bookmark92) 43
      1. [Jood-Services de calcul et de mise en réseau](#_bookmark93) 43
      2. [Jood-Services de stockage et de livraison de contenu](#_bookmark99) 45
      3. [Jood-Services de sécurité et d’identité](#_bookmark101) 46
      4. [Jood-Services de Management](#_bookmark104) 47
      5. [Jood-Outils de développement](#_bookmark105) 47
      6. [La sécurisation de l’infrastructiure de Jood](#_bookmark108) 48
      7. [L’architecture global](#_bookmark111) 50
   7. [Conclusion](#_bookmark113) 52
4. [Jood - développement backend](#_bookmark114)53
   1. [Architecture backend du Jood](#_bookmark115) 53
      1. [Répertoire Pytest\_cache](#_bookmark117) 54
      2. [Répertoire Project](#_bookmark119) 54
      3. [Répertoire Templates](#_bookmark121) 55
      4. [Répertoire apps](#_bookmark122) 55
      5. [Répertoire Tests](#_bookmark125) 56
      6. [Les fichiers Docker](#_bookmark127) 56
      7. [Autre Fichiers](#_bookmark129) 57
   2. [Interaction avec l’ERP Odoo](#_bookmark130) 57
   3. [L’API de Jood](#_bookmark136) 58
   4. [Conclusion](#_bookmark144) 61
5. [Jood - développement Front-end](#_bookmark145)62
   1. [Introduction](#_bookmark146) 62
   2. [Organisation du projet](#_bookmark148) 63
   3. [Workflow de JoodPerfumes](#_bookmark150) 64
      1. [Interfaces d’utilisateur principale](#_bookmark151) 64
      2. [Autres pages et composants](#_bookmark160) 70
   4. [Conclusion](#_bookmark164) 72

[Conclusion](#_bookmark165) 73

[Liste des abréviations](#_bookmark166)75

[Webographie](#_bookmark167) 76

**Table des figures**

* 1. [L’organigramme de la société d’accueil](#_bookmark7) 4
  2. [les coordonnées de l’entreprise](#_bookmark10) 5
  3. [Aperçu de Slack](#_bookmark18) 8
  4. [Les messages directs](#_bookmark19) 9
  5. [Workflow dans JIRA](#_bookmark21) 10
  6. [Sprint Backlog *Joodperfumes.com* dans JIRA](#_bookmark22) 10
  7. [Méthode agile Scrum](#_bookmark24) 11
  8. [Diagramme de Gantt](#_bookmark26) 12
  9. [Diagramme de cas d’utilisation d’un Client](#_bookmark36) 16
  10. [Diagramme de cas d’utilisation de l’admin du plateforme](#_bookmark38) 17
  11. [Diagramme de séquences : cas d’utilisation d’authentification](#_bookmark40) 18
  12. [Diagramme de séquences : cas d’utilisation d’achat sur la plateforme](#_bookmark41) 18
  13. [Diagramme de séquences : cas d’utilisation de paiment via KNET](#_bookmark42) 19
  14. [Diagramme de classes du site](#_bookmark45) 21
  15. [Architecture MVC](#_bookmark48) 22
  16. [Design Pattern Active Record](#_bookmark49) 22
  17. [Exemple d’architecture microservices](#_bookmark51) 23
  18. [Pipeline d’exécution de Terraform](#_bookmark56) 25
  19. [Aperçu de jood sur GitHub](#_bookmark58) 27
  20. [Appel d’une Tache celery](#_bookmark59) 28
  21. [Les principales fonctionnalités de ERP Odoo](#_bookmark60) 29
  22. [Tendances du Cloud Computing sur Stack Overflow](#_bookmark70) 38
  23. [HashiCorp Configuration Language](#_bookmark73) 39
  24. [Creation de S3 bucket avec terraform](#_bookmark74) 39
  25. [Résultat sur AWS](#_bookmark75) 39
  26. [Exemples de fournisseurs Terraform](#_bookmark77) 40
  27. [Variables avec Terraform](#_bookmark79) 40
  28. [Modules avec Terraform](#_bookmark81) 41
  29. [Outputs avec Terraform](#_bookmark83) 41
  30. [Appel d’une Output avec Terraform](#_bookmark84) 41
  31. [Data sources avec Terraform](#_bookmark86) 42
  32. [Appel d’une source de donnees avec Terraform](#_bookmark87) 42
  33. [état (state) avec Terraform](#_bookmark89) 42
  34. [le stockage d’un etat avec Terraform](#_bookmark90) 43
  35. [la gestion de conteneurs avec AWS Fargate](#_bookmark97) 45
  36. [Security groupe du service Jood-core](#_bookmark102) 46
  37. [Security groupe du Load Balancer](#_bookmark103) 47
  38. [Processus de publication avec CodePipelin](#_bookmark106) 47
  39. [Intégration-Déploiement Continue de Jood](#_bookmark107) 48
  40. [Sessions SSH établies par l’intermédiaire d’un hôte Bastion](#_bookmark109) 49
  41. [La configuration des hôtes VPN et Bastion dans jood](#_bookmark110) 50
  42. [L’architecture cloud du Jood sur AWS](#_bookmark112) 51
  43. [Arhitecture du Jood-Core](#_bookmark116) 53
  44. [Le Répertoire du Pytest\_cahe](#_bookmark118) 54
  45. [le répertoire d’arborescence du projet](#_bookmark120) 54
  46. [Le répertoire de templates](#_bookmark123) 55
  47. [Le répertoire des modules de projet](#_bookmark124) 56
  48. [Le répertoire des tests](#_bookmark126) 56
  49. [Le fichiers docker de projet](#_bookmark128) 57
  50. [Connexion Jood-Core au serveur Jood-Odoo avec XRPC](#_bookmark131) 57
  51. [Obtenir les modèles de Odoo](#_bookmark132) 57
  52. [Creation d’un ulisateur sur Odoo](#_bookmark133) 58
  53. [création d’un utilisateur sous Odoo avec celery](#_bookmark134) 58
  54. [La création d’un ordre pour un utilisateur dans Odoo](#_bookmark135) 58
  55. [Exemple d’un ProductType avec graphene-django](#_bookmark137) 59
  56. [Exemple de ProductQuery avec graphene-django](#_bookmark138) 59
  57. [Déclaration des requêtes et des mutations dans le schéma du Joodperfumes](#_bookmark139) 60
  58. [L’ajout de graphiql au urlpattern de projet](#_bookmark140) 60
  59. [La Vue de Graphiql api Jood](#_bookmark141) 60
  60. [La requête products dans graphiql](#_bookmark142) 61
  61. [Le résultat de la requête products dans graphiql](#_bookmark143) 61
  62. [Le référentiel du front-end sur github](#_bookmark147) 62
  63. [La structure du Jood-Web](#_bookmark149) 63
  64. [L’interface principale du Joodperfumes](#_bookmark152) 65
  65. [Le choix d’un papier cadeau (Gift Wrapping)](#_bookmark153) 66
  66. [L’icon de panier du plateforme Jood](#_bookmark154) 66
  67. [Le contenu d’un panier dans Jood](#_bookmark155) 67
  68. [Le choix de la méthode de paiment](#_bookmark156) 68
  69. [Ajout d’une nouvelle adresse](#_bookmark157) 69
  70. [Menu de gestion (d’adresses, mot de passe, profile)](#_bookmark158) 69
  71. [Interface de gestion d’adresses](#_bookmark159) 70
  72. [détails d’un produit](#_bookmark161) 71
  73. [Modification des données d’un utilisateur](#_bookmark162) 71
  74. [Interface pour changer un mot de passe](#_bookmark163) 72

* 1. [Scénarios des cas d’utilisation d’un Client](#_bookmark35) 15
  2. [Scénarios des cas d’utilisation de l’admin du plateforme](#_bookmark37) 17
  3. [Identification des Classes](#_bookmark44) 20
  4. [Les réseaux de l’infrastructure Jood](#_bookmark94) 44
  5. [Les Instances EC2 de l’infrastructure Jood](#_bookmark95) 44
  6. [Les Conteneurs Docker de l’infrastructure Jood](#_bookmark96) 44
  7. [Jood-Elb](#_bookmark98) 45
  8. [Les base de donnees RDS du Jood](#_bookmark100) 46

Au terme de nos cinq années d’études effectuées au sein de l’école nationale des sciences appliquées d’Al Hoceima.Nous avouns été amené à passer un stage de fin d’études en vue de l’obtention du diplôme d’ingénieur d’Etat en informatique.Ce stage s’est déroulé au sein de MNS Consulting Group,

Introduction générale

Dans **Le deuxième chapitre** on va aborder l’étude fonctionnelle du projet, qui vous donnera un aperçu sur les principales fonctionnalités de notre système. ainsi que l’étude technique qui décrit ce dernier en termes d’architecture.

Dans **Le troisième chapitre** qui présente les principales technologies et outils que nous utilisons avec Joodperfumes.

Puis on va passer au **quatrième chapitre** qui s’agit de l’architecture cloud du projet qui décrit ce dernier en termes d’infrastructure.

Nous décrivons dans **le cinquième chapitre** l’architecture et la mise en œuvre de la partie back-end du projet.

Enfin, **Le dernier chapitre** concerne la partie front-end qui décrit l’interaction avec les diffé- rentes interfaces des plates-formes.

# 1

## Contexte général du projet

### Introduction

***Ce chapitre présente le contexte général du projet. Nous allons commencer tout d’abord par une présentation générale de l’organisme d’accueil. Ensuite nous allons présenter le projet en parlant de son contexte général, de la problématique pour finir par la solution proposée et conduite et la gestion du projet***

### 1.1 Présentation de MNS Consulting Group

Orange Bytes est une start-up kuwaitienne innovante à fort potentiel de croissance qui a été fondée en 2010 pour se positionner sur le secteur de la technologie, et octroyer à ses utilisateurs des solutions de qualité et surtout adaptées à leurs besoins spécifiques, son siège social est situé à Dubai dans l’U.A.E et elle a une filiale au Maroc qui a été créée en 2014.

Parmi les avantages du travail avec une start-up telle qu’Orange Bytes, on site :

* + - 1. **L’agilité :** Les start-up sont des entreprises jeunes et souples. Elles peuvent avoir un business plan et une stratégie opérationnelle bien installés, mais rien ne les empêche d’en changer en cours de route pour s’adapter à un changement de marché.
      2. **Bonnes perspectives d’évolution :** Lorsque’une personne commence à travailler dans une start-up dès sa création, elle pourra voir la progression de l’entreprise et elle aura aussi la chance de grandir avec elle et de monter en grade plus rapidement.
      3. **Un cadre de travail sympa et cool :** l’atmosphère d’une start-up est généralement très informelle et peu hiérarchique, l’employé sentira comme dans une famille. Aller voir un match de foot avec son patron sera considéré comme quelque chose de normal.
      4. **La variété des tâches :** Les équipes sont de petites tailles et les projets nécessitent souvent plusieurs compétences. Alors les développeurs sont amenés à travailler avec d’autres métiers. C’est enrichissant, vu qu’il permet de s’ouvrir sur les diverses activités de l’entreprise et de découvrir ce que font les autres en tant que développeurs.
      5. **L’environnement de travail :** L’environnement de travail est très dynamique. L’équipe est la plupart du temps jeune et pleine de ressources.

La figure ci-dessous montre l’organigramme de la société d’accueil :

3

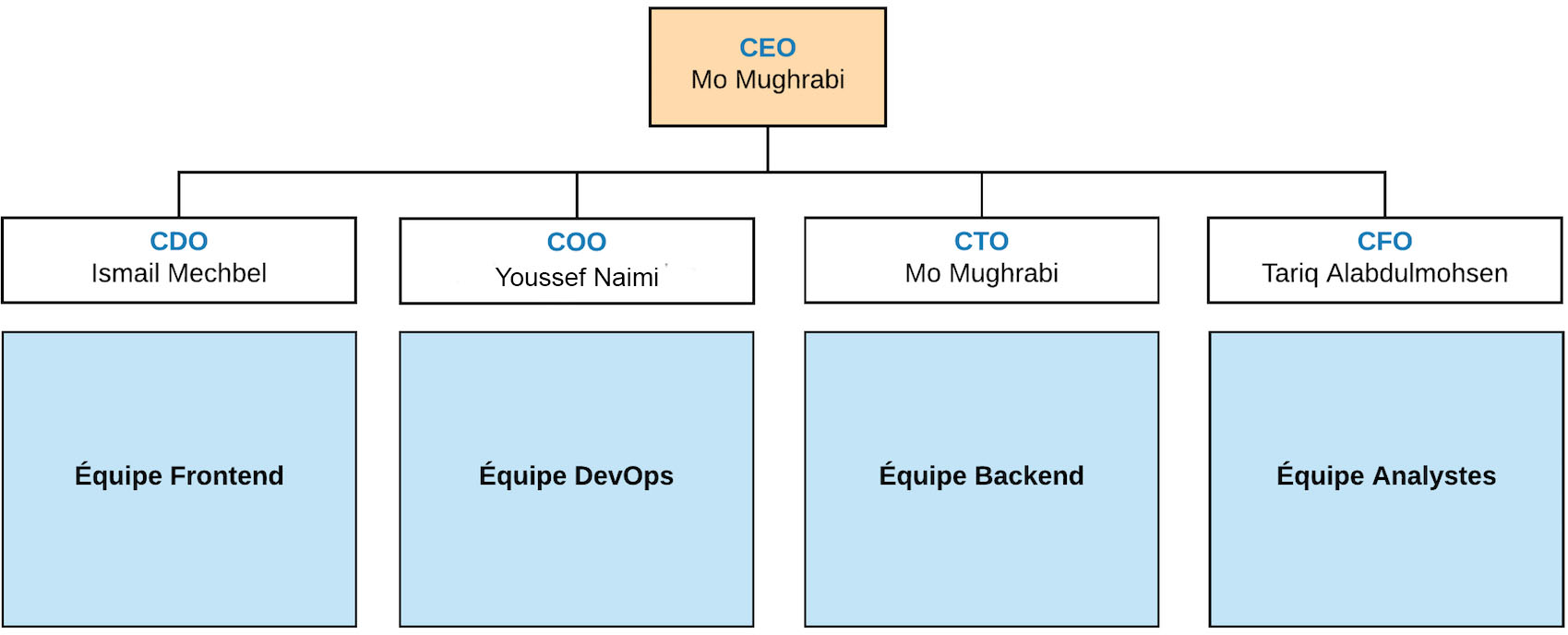


Figure 1.1: L’organigramme de la société d’accueil

### Présentation du projet

Les entreprises ont besoin de garder une trace de leurs activités, que ce soit de la comptabilité, des ressources humaines ou du flux des activités de l’entreprise. L’approche empirique consistait à tenir des registres sur lesquels étaient consignés l’ensemble de ces activités. Cette approche présente l’inconvénient de rapidement devenir difficile à gérer. En effet, avec le temps les papiers s’accumulent et deviennent ainsi ingérables.

Avec le développement de l’informatique, il est aujourd’hui plus facile de gérer toute ces problématiques avec plus de contrôle et moins d’efforts.

D’autre part, pour une entreprise commerciale, il est important d’avoir de la visibilité sur le marché, de pouvoir suivre ses ventes et de disposer de certaines statistiques pour connaitre et anticiper les besoins du marché afin d’optimiser ses revenues et de satisfaire ses clients. C’est précisément dans cette optique que s’inscrit notre projet de fin d’études.

### Problématique du projet

Strass Dakar est un magasin de vente de prêts à porter pour homme et femme de tout age.La majeur partie de ses produits proviennent de fournisseurs étrangers et l’établissement dispose de deux boutiques : Strass et Strass Kids qui partagent le même stock, ce qui est difficile à gérer . Le propriétaire voudrait disposer de chiffres exacts sur les ventes pour avoir une idée sur les tendances et tirer profit de ses déplacements à l’extérieur.

D’autre part, la boutique permet aux clients de retourner des articles défectueux pour des retouches qui peuvent parfois durer. Il ne faudrait pas que ces derniers se mélangent avec le reste du stock.

Strass dispose d’une page Facebook sur laquelle il fait la promotion de ses articles et permet aux clients de passer des commandes directement à partir de Facebook. Le problème qui se pose est que cette plateforme n’est pas connectée à son stock et que pour chaque commande il faudrait qu’un commerciale se charge de vérifier d’abord sa disponibilité avant de confirmer la commande ;

ce qui peut parfois prendre du temps et même entrainer des conflits dans les situations où il y’a beaucoup de commandes. En outre, le paiement dans ces cas se fait à la caisse ou à la livraison.

Le gérant voudrait aussi gérer les dépenses des boutiques et le personnel avec fiabilité et traçabilité.

### Solution proposée

Pour répondre aux besoins du clients, la solution adoptée consiste à mettre en œuvre une application permettant de digitaliser la gestion du stock, des ventes, des employés et des autres problématiques inhérentes à l’activité de la boutique. L’application ***Seytu*** devra en plus pouvoir être adaptable à des besoins plus spécifiques et être évolutive.

La première partie de ce projet consistera à mettre en place le module de gestion de stock et des employés, puis par la suite le module des ventes en accord avec les besoins du client Strass. La seconde partie sera axé sur la gestion des finances et des clients.

### Conduite et gestion du projet

Pour mener à bien un projet, il est nécessaire de se conformer à une méthode de gestion de projet qui a fait ses preuves. La gestion de projet est l’ensemble des activités visant à organiser le bon déroulement d’un projet et à atteindre les objectifs fixés dans les délais imparties.

En pratique pour qu’un projet soit réussi, il doit être adaptable à des modifications fréquentes, mais maitrisé et planifié. Le projet doit rester dynamique et équilibrer continuellement les contraintes techniques, de cout et de délai. Il s’agit donc finalement de trouver l’équilibre qualité-cout-délai.

Chez MNS Consulting, la méthode de gestion de projet utilisée est la méthode SCRUM de la famille des méthodes agiles. Ces derniers se veulent plus pragmatiques que les méthodes traditionnelles, impliquent au maximum le client et permettent une grande réactivité à ses demandes. Les méthodes agiles reposent sur un cycle de développement itératif, incrémental et adaptatif. A chaque itération une partie du projet aura été traitée et à la fin de l’itération une version autonome devra être livrer en production.

### Contraintes du projet

Les projets rencontrent souvent des contraintes au cours de leurs mise en œuvre. Ces contraintes imposent souvent des conditions non anticipées et peuvent nuire au bon déroulement du

projet. Elles peuvent provenir de facteurs externes ou internes. Il est donc primordial de bien cerner l’environnement du projet afin de d’identifier et d’anticiper les contraintes potentielles pour éviter un impact négatif sur le projet.

Pour ce projet, on peut naturellement dégager les contraintes générales : respecter les délais prévus

et respecter les exigences du client de façon stricte. On a aussi des contraintes d’ordre techniques :

* Partir sur la base d’une solution logicielle plus générique et l’adapter de façon spécifique au client Strass,
* La solution Seytu doit être fortement extensible et évolutif avec des besoins initiaux assez vagues et susceptibles de beaucoup changer au cours de la réalisation.
* Offrir une expérience utilisateur satisfaisant au client.

### Planification du projet :

La planification est une étape nécessaire au bon déroulement d’un projet. Elle permet de bien délimiter le projet et de pouvoir estimer les délais et ressources nécessaires à sa réalisation.

Pour planifier un projet, la première phase consiste à le découper en plusieurs étapes, d’en estimer les durées et d’identifier leurs enchainements, ensuite après cela, affecter les ressources nécessaires humaines comme financières. L’étape suivante consiste à passer à la réalisation proprement dite en se conformant au maximum au plan élaboré tout en prenant en considération qu’une planification exacte requiert une parfaite maitrise des outils et de l’environnement de développement.

Chez MNS Consulting, l’outil utilsé pour la gestion de projet et la suivi des taches est JIRA software qui est un système développé par Atlassian Software systems.Grace à son aspect collaboratif, les différents parties prenantes du projet travaillent en synergie focalisées sur leurs objectifs propres avec une grande facilité.Cet outil permet de gerer les le cycle de développement du projet avec les méthodes agiles.

Ci-dessous un aperçu de Jira.

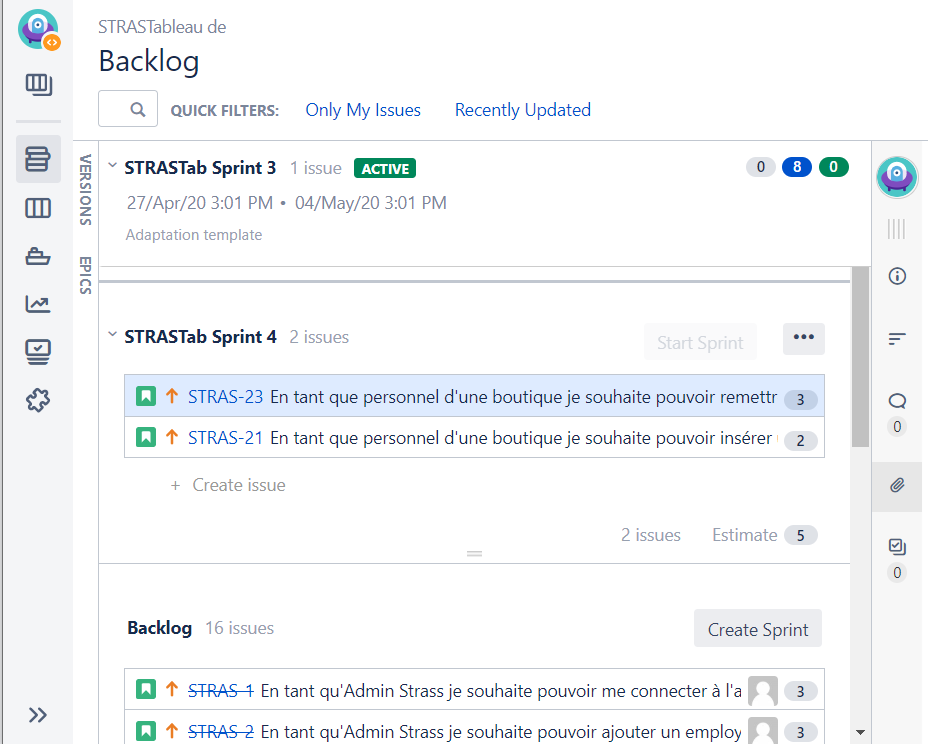


Figure 1 Aperçu Baclog Strass sous JIRA

### Conclusion :

Ce chapitre était consacré à la présentation de la structure d’accueil MNS Consulting mais aussi à la présentation de notre projet en le situant dans son contexte et à illustrer la problématique traitée. Nous avons aussi parler de la méthode de gestion de projet utilisée .Le chapitre suivant portera sur l’étude technique et fonctionnelle du projet .

### 

# 2

## Etude fonctionnelle - technique

### Introduction

***Ce chapitre est consacré à l’étude fonctionnelle et technique du projet. Il s’agira d’analyser et de comprendre les besoins du client, de cerner les caractéristiques du système à mettre en place et de proposer une solution appropriée aux besoins du client.***

### Spécification des besoins

Ce projet consiste à développer une solution logicielle permettant la gestion d’un établissement commerciale pour le compte d’un client dénommé Strass. La solution attendue se veut également extensible et adaptable à d’autres clients aux besoins proches. Les besoins formulés par le client peuvent se présenter comme suit :

**Gestion de stock :** le client veut digitaliser la gestion de son stock. Cela implique de connaitre l’état du stock à tout moment et d’avoir des statistiques sur les produits et leurs mouvements.

**Gestion des ventes :** pouvoir effectuer des ventes sur la plateforme et pouvoir les tracer.

**Gestion des clients :** avec une politique de fidélisation en place, garder l’historique des achats des clients.

**Gestion du personnel et des finances :** gérer les employés de l’établissement et leur permettre de travailler via la plateforme. Gérer le paiement des employés et les charges du magasin

### Analyse fonctionnelle

L’étude ou l’analyse fonctionnelle est un élément indispensable pour la bonne réalisation d’un projet, puisqu’elle a pour but de recenser et valoriser toutes les fonctionnalités inhérentes au système étudié. Cela permettra d’une part de cerner exhaustivement les caractéristiques intrinsèques du système, d’autre part, de choisir convenablement les mécanismes et techniques pour sa mise en œuvre.

Cette étude fonctionnelle, identifie dans un premier temps les acteurs qui interviennent à l’élaboration de l’ensemble des processus du système, puis étudier dans un deuxième temps les différents diagrammes UML.

#### Identification des acteurs

Un acteur désigne un rôle joué par une entité externe qui agit sur le système en échangeant de l’information en entrée et/ou en sortie. Il peut s’agit notamment d’une personne physique ou d’un programme. Les acteurs identifiés dans notre système sont :

* + - * **Gérant :** C’est un super utilisateur possédant toutes les permissions (Ajout, modification et suppression) sur tous les modèles sans exceptions via l’interface d’administration.
      * **Employé :** Un employé de l’établissement qui a ou non un compte de connexion sur la plateforme et qui peut effectuer un des actions restreintes.
      * **Client :** Un client de la boutique qui effectue soit des achats en boutique soit des achats en ligne.

#### Architecture de la solution

L’analyse méticuleuse des besoins spécifiés et la nature hétérogène du système nous a permis d’orienter le choix de l’architecture de la solution vers les microservices. En effet les différents besoins soulevés peuvent se regrouper en domaines qui n’ont pas forcément une grande liaison et peuvent donc se scinder en microservices dédiés pour chacun d’eux. Après une étude documentée, nous sommes arrivés à la décomposition suivante :

* + - * **Gestion du personnel et des finances :** microservice qui gère tout ce qui est en rapport avec le personnel et les finances.
      * **Gestion des clients**
      * **Gestion des produits et des ventes**

A ces microservices fonctionnels s’ajoutera d’autres microservices qui nous permettront d’optimiser la gestion de l’infrastructure globale en particulier la gestion des problématiques transverses comme la sécurité et la journalisation.

Ci-après, une modélisation de l’architecture :

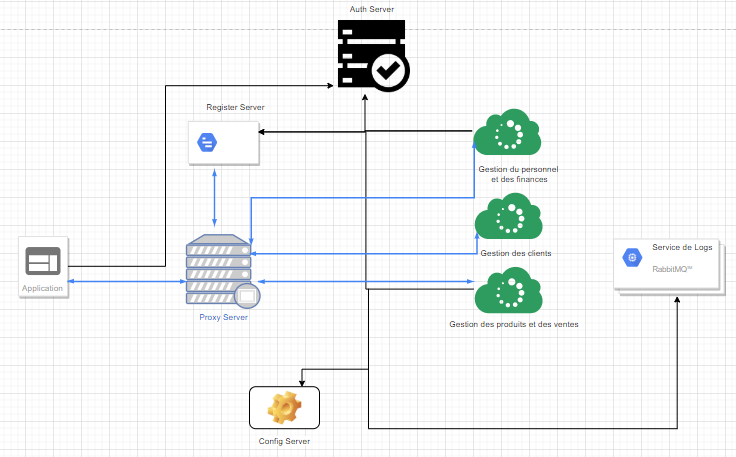


Figure 2- Architecture de la solution

#### Diagrammes de cas d’utilisation

C’est une description du système du point de vue utilisateur sous forme d’action et de réaction. Un cas d’utilisation modélise donc un service rendu par le système et permet d’exprimer un besoin d’un acteur du système. Il doit produire un résultat visible pour les acteurs.

Nous avons élaboré un diagramme de cas d’utilisation pour chaque module de l’application pour avoir un ensemble fonctionnel cohérant et spécifique.

##### 2.3.3.1 Cas d’utilisation module gestion de stock

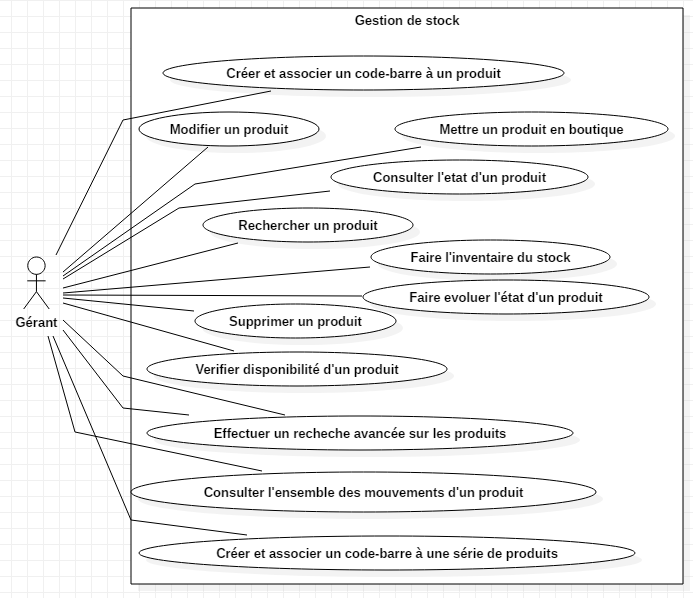


Figure 3 Cas d'utilisation gestion de stock

##### 2.3.3.2 Cas d’utilisation gestion des ventes :

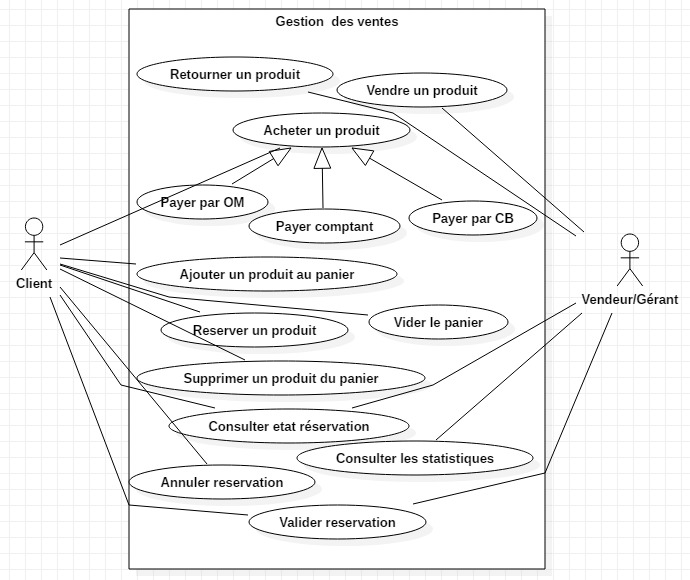


Figure 4 Cas d'utilisation gestion des ventes

##### 2.3.3.3 Cas d’utilisation gestion du personnel et des finances

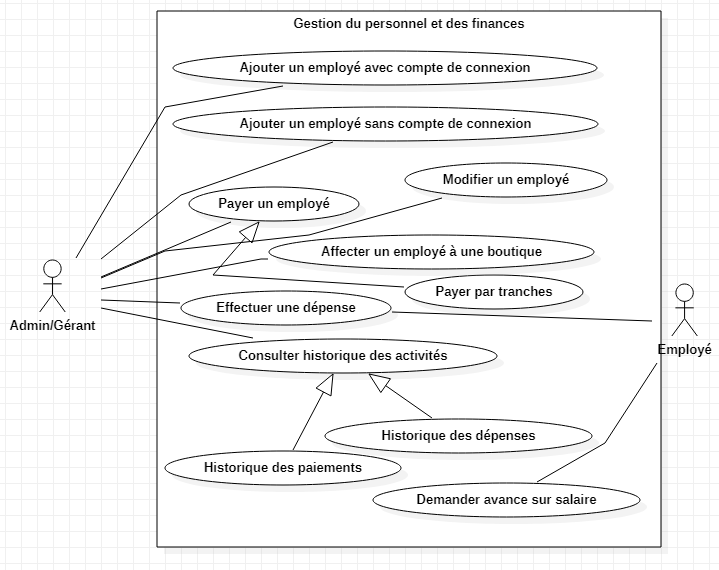


Figure 5 Cas d'utilisation gestion du personnel et des finances

##### 2.3.3.4 Cas d’utilisation gestion des clients

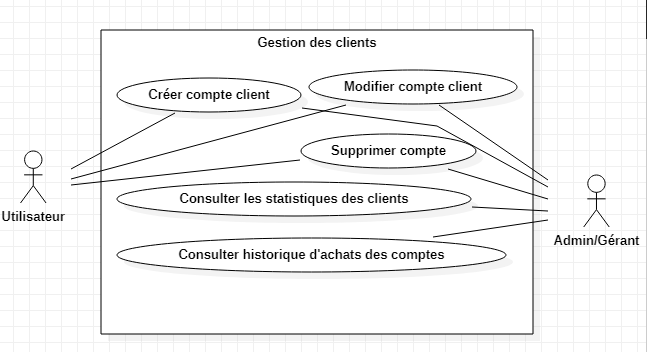


Figure 6 Cas d'utilisation gestion des clients

#### Diagrammes de classes

Le diagramme de classe modélise les concepts du domaine d’application dans le cadre le cadre de l’implémentation d’une application. Il permet de fournir une représentation abstraite des objets du systèmes qui vont interagir pour réaliser les cas d’utilisation. La description du diagramme de classe est centrée sur trois concepts : le concept d’objet, le concept de classe d’objets comprenant les attributs et les méthodes et les relations entre classes.

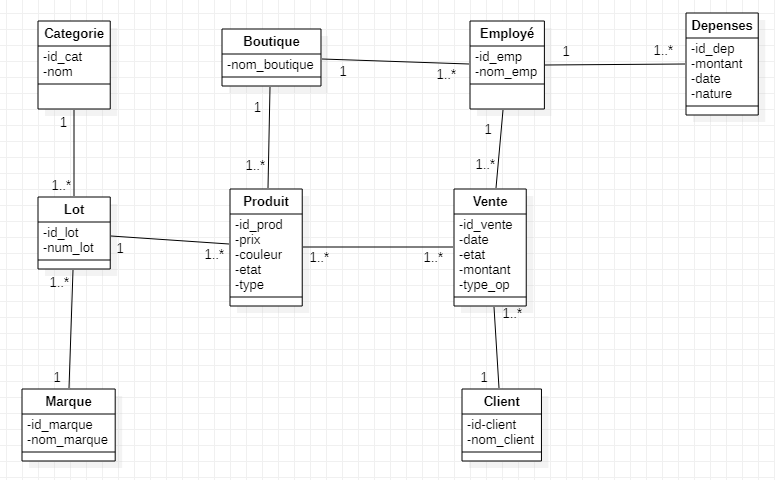


Figure 7 Diagramme de classe du système

### Etude technique du projet

Après avoir clairement défini et compris les besoins d’un projet, on doit chercher les moyens nécessaires pour sa réalisation à travers une étude technique méticuleuse afin de livrer un projet de qualité satisfaisant les besoins du client. Il sera question dans ce chapitre de justifier le choix de l’architecture de la solution évoqué lors de l’analyse fonctionnelle et d’une description ou présentation des différentes technologie choisies pour la réalisation de ce projet

#### Pourquoi les microservices ?

Le projet consiste à développer une solution de gestion d’un établissement commerciale. Il s’agit de la gestion des ventes, de la gestion du stock, de la gestion des clients, de la gestion des employés et de la gestion des finances. Il s’agira donc d’une application plutôt large.

D’un autre côté, les besoins du client ne sont pas fixes et évoluent tout le temps avec un ensemble de fonctionnalités pouvant être disponibles séparément et qui de ce fait pourront permettre au client de rapidement avoir un produit disponible et utilisable.

L’application est destinée à la gestion de toute l’établissement au quotidien, elle doit donc être robuste, tolérante aux pannes et hautement disponible et ceci à chaque instant.

On a donc des exigences bien précises et la solution livrée doit répondre à toute ces attentes. On doit donc trouver l’approche technique et l’architecture logicielle qui nous permettent d’y arriver

##### L’architecture monolithique :

L’approche monolithique consiste à développer une application en un seul bloc, dans laquelle est centralisé tous les besoins fonctionnels. Une application monolithique est conçue avec un ensemble de composants interconnectés et interdépendants qui travaillent en chainent pour fournir une réponse. Dans ce type d’architecture étroitement intégrée, chaque composant et ceux qui lui sont associés doivent être présents pour permettre l’exécution de l’application. L’architecture monolithique était jusqu’‘en 2011 le standard en matière de développement logicielle mais à partir de cette année, une nouvelle approche se basant sur l’architecture orientée service a été théorisé : le concept des microservices

##### Les microservices :

Le style architectural microservices est une approche pour développer une application unique comme une suite de petits services chacun fonctionnant indépendamment et communicants avec des mécanismes légers. La philosophie derrière cette architecture s’inspire en grande partie de celle d’UNIX qui prône : “ Ne faire qu’une seule chose et la faire bien.”

Avec les microservices, les services sont petits et conçus pour remplir une seule fonction ou un nombre minimal de fonction. Chaque service est élastique, résilient, composable, minimale et complet. L’organisation du projet doit prendre en compte l’automatisation, le déploiement et les tests.

##### Monolithique Vs Microservices :

La divergence majeure entre les microservices et l’architecture monolithique concerne l’infrastructure qui est centralisé pour les monolithiques et distribuée pour les microservices. Cette divergence induit d’autres différences sous-jacentes :

* Les architectures monolithiques sont simples à construire, à tester et à déployer. L’application peut facilement supporter la montée en charge en exécutant plusieurs nouvelles instances. Pour les microservices, la construction reste aussi relativement simple vu qu’elles sont minimales mais les tests d’intégration sont plus complexes à réaliser à cause du nombre de composants intervenants.
* Avec une base de code unique, les applications monolithiques peuvent facilement gérer les problématiques transversaux telles que la journalisation, la sécurité, les configurations etc. Ces problématiques sont en revanche plus complexes à gérer dans une architecture distribuée.
* Les microservices communiquent entre-elles à travers des API(Application Programing Interface).Cette communication introduit une certaine latence à l’exécution d’une requête, ce qui n’est pas le cas dans un système monolithique où les composants partagent la mémoire vive ce qui rend le traitement des requêtes plus rapide.
* La plus grande vulnérabilité de l’architecture monolithique est le couplage fort entre les composants de l’application. Cela affecte l’évolutivité et la maintenabilité de l’application mais aussi le déploiement continu car la moindre modification dans une partie de l’application pourrait nécessiter de revoir entièrement cette dernière puis de retester et de redéployer l’ensemble.
* Les applications monolithiques sont restreintes à une seule pile technologique, cela crée une limitation conséquente dans le sens où la technologie évolue très vite et certains langages sont plus performants que d’autres dans certains domaines, ce qui fait qu’on n’arrive pas à pleinement tirer profit de chacun d’entre eux selon nos besoins. Cette limitation est inexistante pour les microservices. Chaque microservice peut être développer avec une technologie différente.
* Les microservices s’adaptent très facilement à l’évolution ou au changement des besoins car les services sont indépendants les uns des autres. La montée en charge est optimisée avec les microservices. En effet, pour une grande application au sein de laquelle il y’a une partie spécifique qui connait un pic de requêtes, il est juste nécessaire d’exécuter plus d’instances du service concerné au lieu de plus d’instance d’une application entière.

#### Architecture Global - Applications

**Joodperfumes** est un projet qui est décomposé en plusieurs applications/services faiblement couplés, souvent spécialisés dans une seule tâche. Les processus indépendants communiquent les uns avec les autres dans un seul projet.

Pour s’attaquer au problème de la complexité on a adpoté cette architecture de modèles. Cette ar- chitecture nous a permis de décomposer ce qui pourrait être une application monolithique, en plusieurs services, chacun indépendant de l’autre .

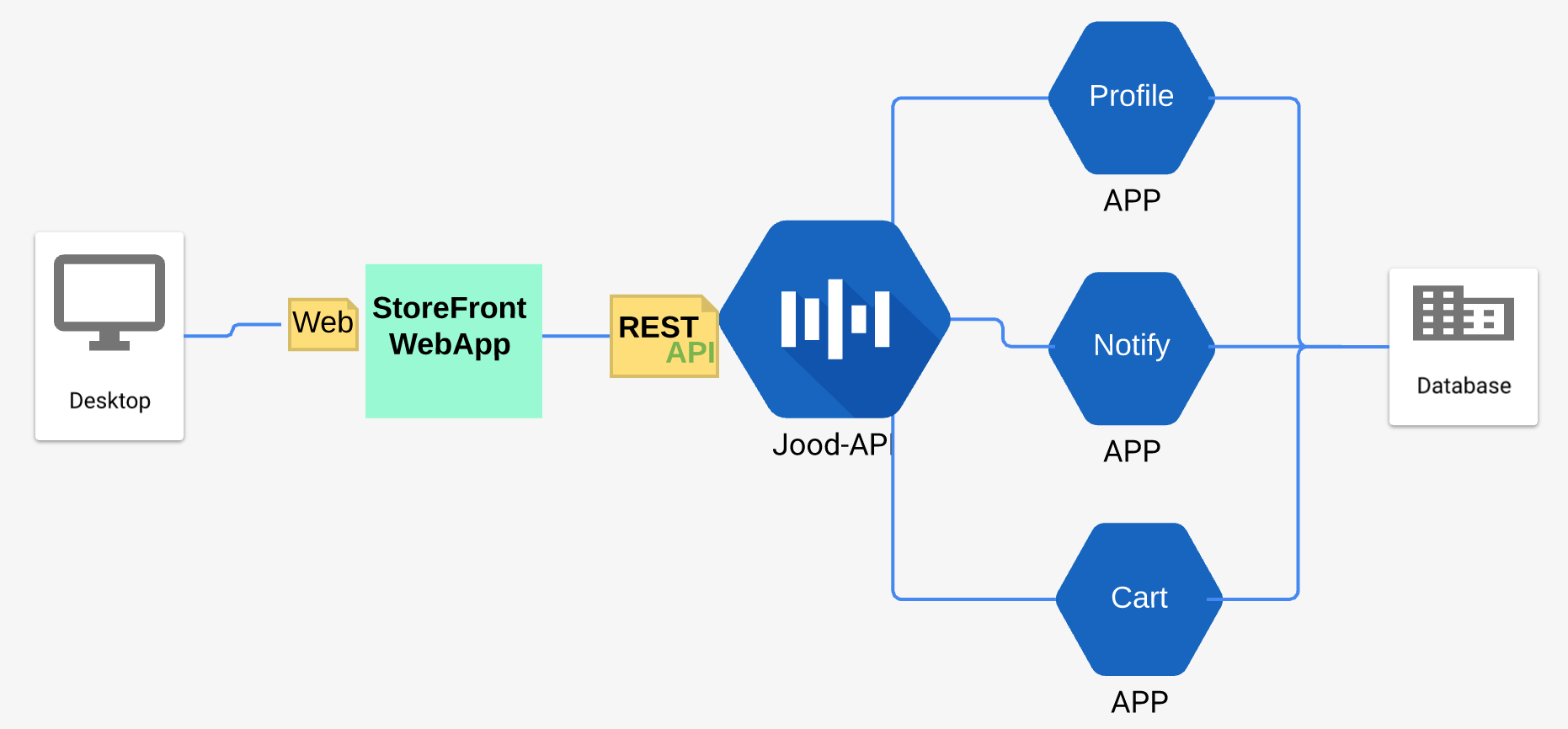


Figure 2.9: Exemple d’architecture microservices

#### 

### Conclusion

Cette étape été nécessaire pour la structuration de la couche métier, Ainsi nous avons raffiné les diagrammes de classe d’analyse, définit des modules et des composants.

Ensuite, nous avons traité l’étude technique du projet, au cours de laquelle nous avons détaillé les caractéristiques techniques de la solution escomptée qui sera utilisé dans la réalisation et mise en oeuvre qui fera l’objet du chapitre suivant.

# 3

## Technologies Utilisées

### Introduction

***Dans ce chapitre, je présenterai les principales technologies que nous utilisons avec Joodperfumes. Les technologies sont divisées par leur portée. Certaines sont spécifiques aux opérations et l’autre au développement, aux tests, au développement frontal et à la surveillance.. Le choix de l’environnement technique du travail est une étape primordiale avant d’entamer la phase de l’implémentation. Cette partie décrit l’ensemble des outils technologiques utilisés lors de développement du système.***

### Outils d’infrastructure Cloud

**AWS :** Amazon Web Services (AWS) est une division du groupe américain de commerce électronique Amazon.com, spécialisée dans les services de cloud computing à la demande pour les entreprises et particuliers .

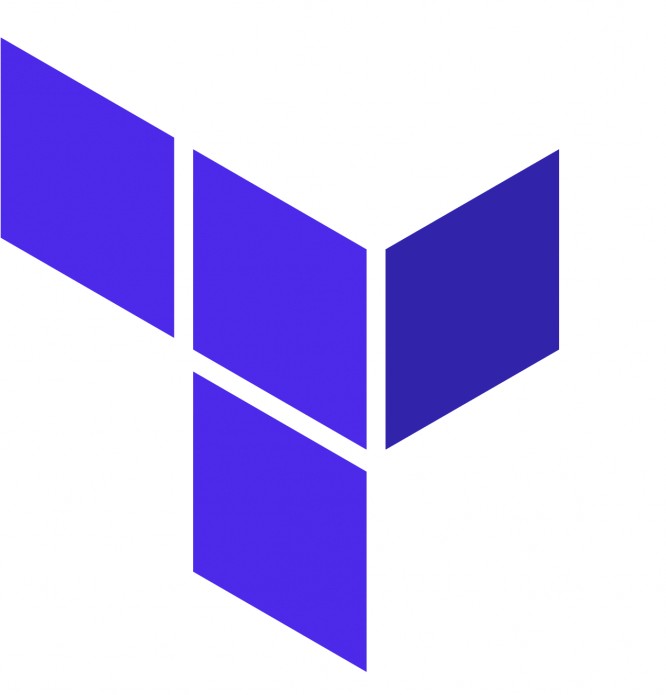
AWS propose plus de 90 services, comprenant le calcul, le sto- ckage, le réseau, la base de données, l’analyse de données, des services

applicatifs, du déploiement, de la gestion de système, de la gestion d’applications mobiles, des outils pour les développeurs et pour l’internet des objets.

**Cloudflare :** CloudFlare est une technologie d’optimisation qui utilise plusieurs techniques pour vous permettre d’optimiser les per- formances de votre site sans disposer de toute la gestion et de l’ad- ministration nécessaires à son accomplissement. Il réduit le temps de chargement de votre site Web, augmente sa sécurité et son efficacité et améliore ses performances globales.

**Terraform :** Terraform est un outil créé en 2014 par HashiCorp, la société qui a créé d’autres outils : Consul, Vagrant, Vault, Atlas, Packer et Nomad. C’est un outil open source, écrit en Go, avec une communauté active de plus de 1200 contributeurs, plus de 15 000 stars sur Github, et qui repose sur une architecture basée sur les plugins.

Terraform (TF) est un outil qui permet de construire, modifier et versionner une infrastructure.

Contrairement à ce que l’on peut lire sur Internet, la techno- logie n’est pas « plateforme agnostique », mais elle permet d’utiliser plusieurs providers dans un même template de configuration. Il existe en effet des plugins pour des providers de Cloud, des services d’hé- bergement, des SCM. . . Nous le verrons un peu plus tard dans ce chapitre.

Que fait l’outil ?

—Il assure la création et la cohérence d’infrastructure.

—Il permet d’appliquer des modifications incrémentales.

—On peut détruire des ressources si besoin.

—On peut prévisualiser les modifications avant de les appliquer.

Terraform permet de créer automatiquement son infrastructure, comme des machines virtuelles, des lambdas Amazon Web Services (AWS). . . le tout, simplement, en trois étapes :

La figure4.[3sc](#_bookmark74)hématise le Pipeline d’exécution de Terraform.

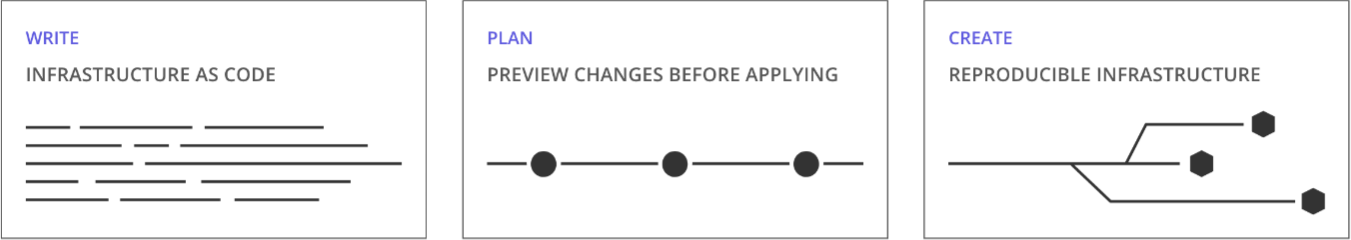
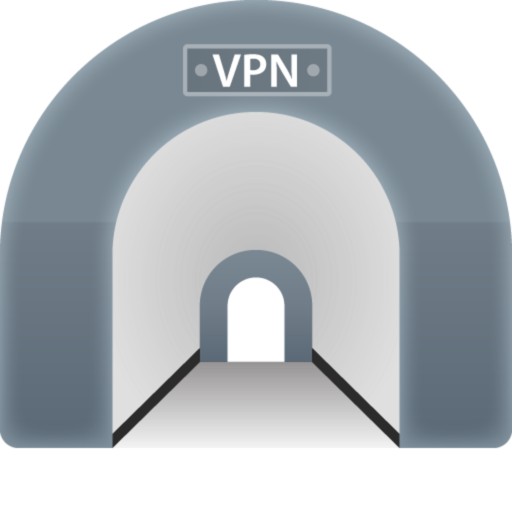


Figure 3.1: Pipeline d’exécution de Terraform

* + 1. Write : écriture de la définition de ses ressources dans des fichiers au format \*.tf ;
    2. Plan : un plan des ressources à créer/modifier supprimer est affiché, avant tout changement ; 3.Create : l’infrastructure voulue est mise en place, et reproductible dans tous les environnements

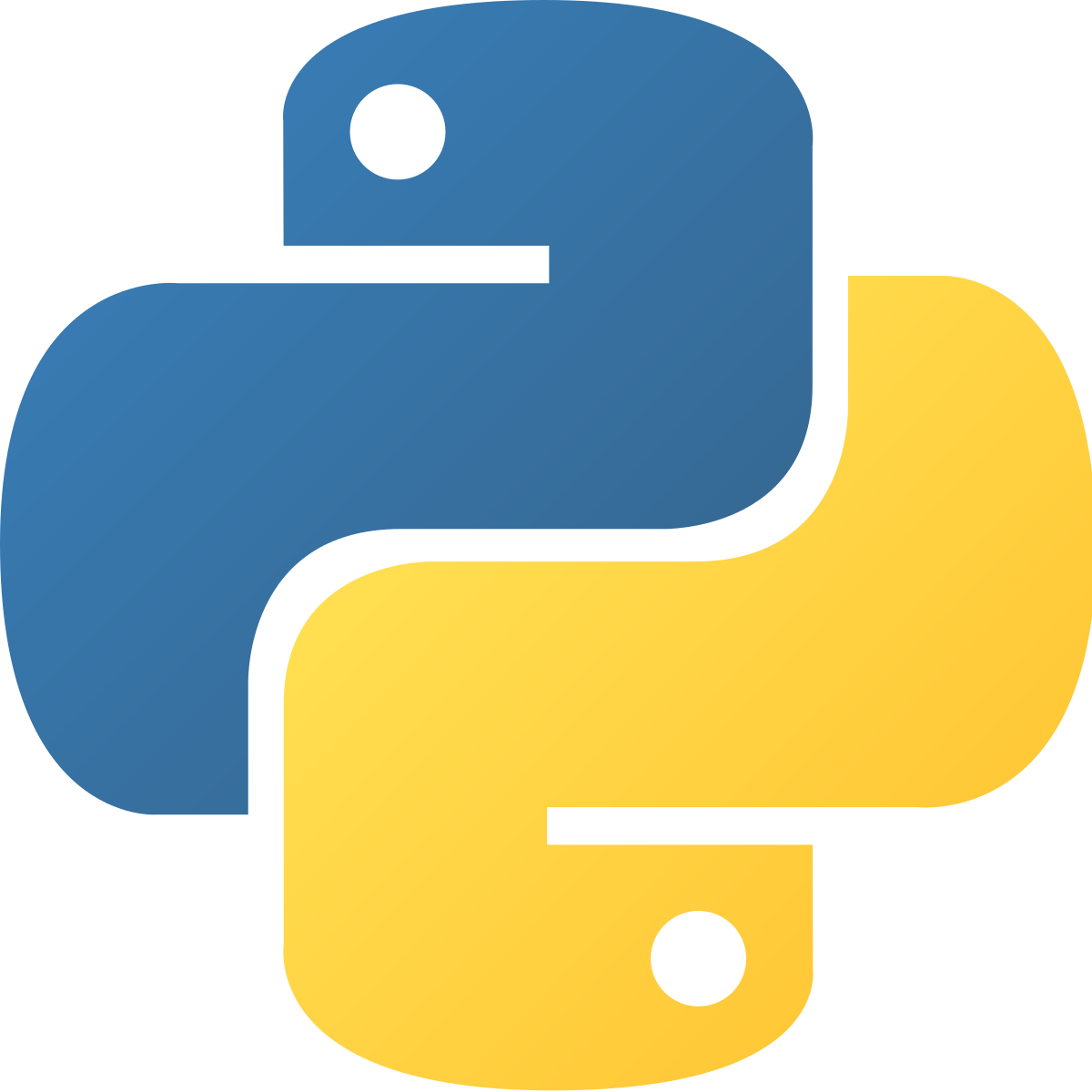
souh

**Tunnelblick** Tunnelblick est une interface utilisateur graphique gratuite et open source pour OpenVPN® sous macOS. Il per- met de contrôler facilement les connexions client et/ou serveur OpenVPN.

### Outils de développement -Backend

Le développement d’applications web et de sites Internet présente des points communs. D’où l’utilisation des Framework qui vient avec un ensemble cohérent de composants logiciels structurels, qui sert à créer les fondations ainsi que les grandes lignes d’une partie d’un logiciel (architecture). Notre projet se base totalement sur des technologies open-source.

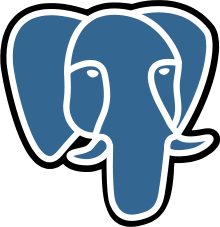
**Langage Python** Python est un langage de programmation orienté Object similaire au Perl et Ruby. Sa force majeure est qu’il propose des milliers de modules qui vous feront gagner un temps précieux durant le développement de vos applications.

Python est un langage interprété, c’est à dire qu’il n’est pas compilé. L’interpréteur Python exécute directement des instructions écrites dans un fichier source contenant du texte. Bien-sûr des séries d’optimisations permettent de garder une version entre le binaire et la source pour ne pas réinterpréter tout le texte à chaque exécution.

Les applications Python sont donc directement cross-platform puisqu’il suffit d’installer l’interpréteur Python sur un des très nombreux systèmes supportés pour que l’application puisse tourner.

**Django Framework** Django est un framework web Python de haut niveau qui encourage le développement rapide et propre. Gratuit et open source, Django vous permet d’éviter de réinventer la roue grâce à toutes les libraires disponibles en Python, mais aussi de tout ce que ce framework offre dès son installation.

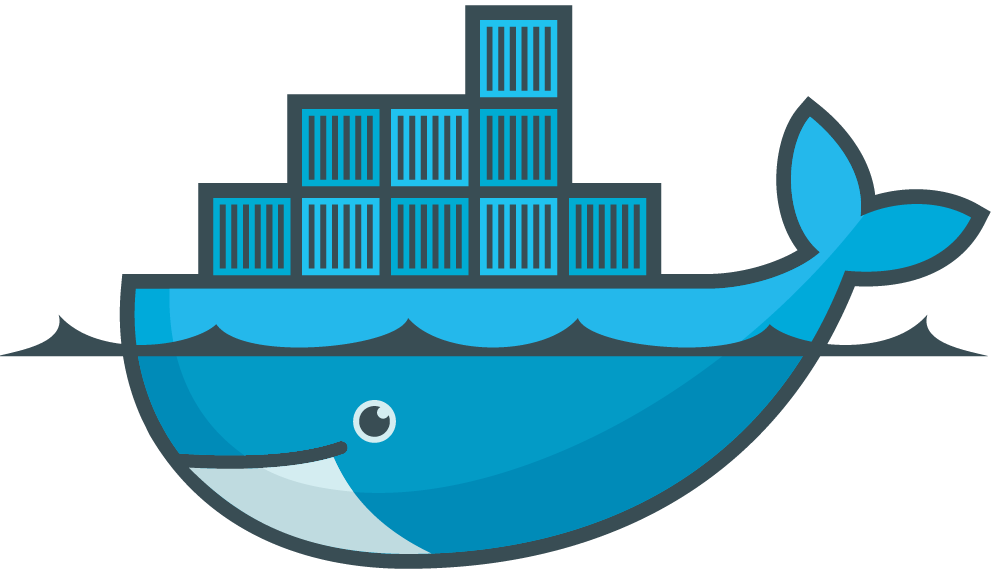
Django a été créé en 2005 et a beaucoup évolué depuis. Il dispose d’une bonne maturité, et d’une bonne stabilité grâce à son système de branche test, et stable. La documentation est claire quant aux versions et branches, ce qui est souvent oublié dans les documentations de logiciels informatiques. Les développeurs veulent souvent garder la documentation au niveau de la dernière version.

**SGBD : PostgreSQL** PostgreSQL est un système de gestion de bases de données relationnel robuste et puissant, aux fonctionnalités riches et avancées, capable de manipuler en toute fiabilité de gros volumes de données, mêmes dans des situations critiques.

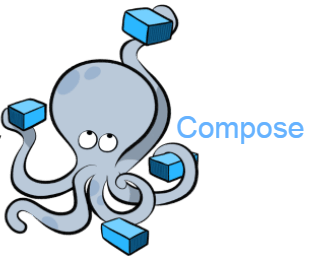
Il est extrêmement respectueux des standards, se conformant au

plus près à la norme ANSI-SQL 2008. Il supporte plus d’une douzaine de langages de programmation, dont Java, Perl, Python, Ruby, Tcl, C / C ++. Son propre PL / pgSQL est similaire à PL / SQL d’Oracle.

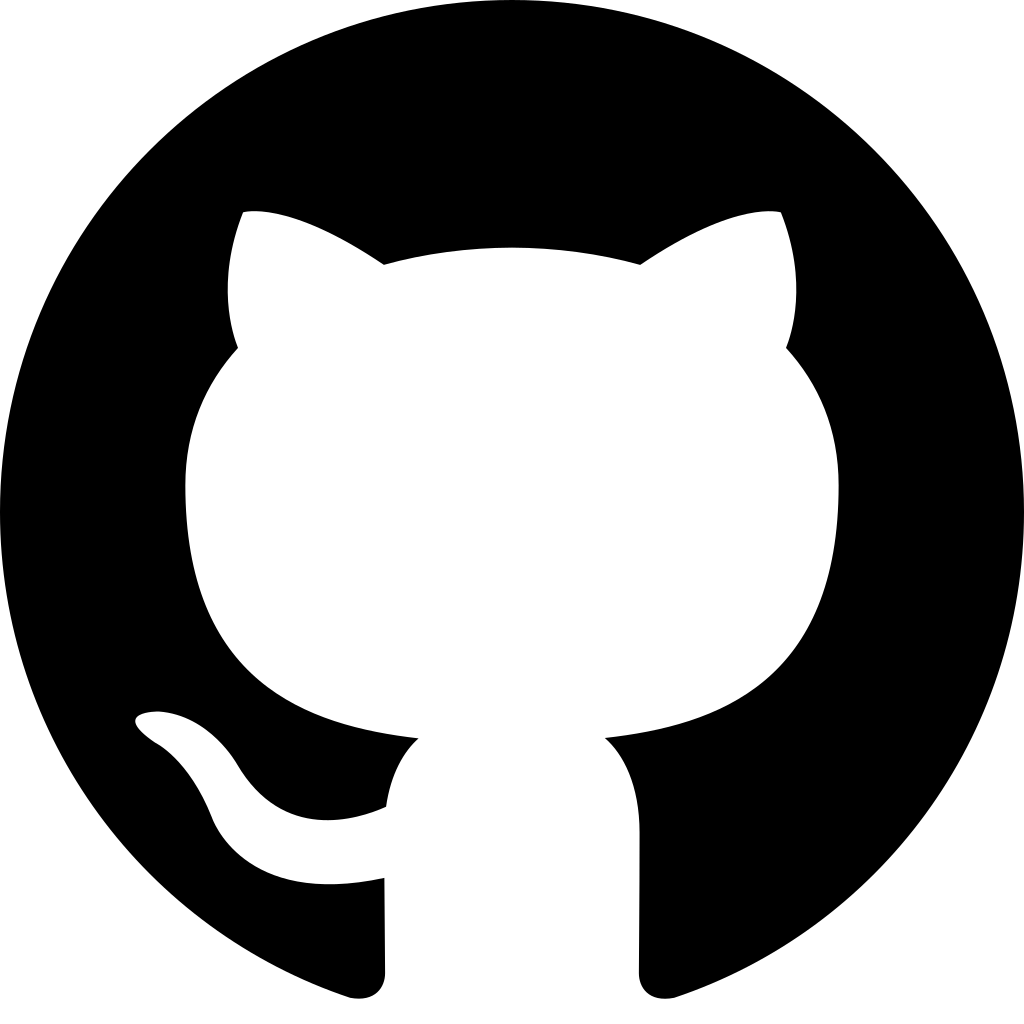
Open-source, il est développé par une communauté mondiale regroupant des milliers de dévelop- peurs et plusieurs dizaines d’entreprises.

**Docker** Docker est une plate-forme ouverte pour le développement, l’embarquement (shipping) et l’exécution d’applications. Docker per- met de séparer les applications de l’infrastructure afin qu’on puisse livrer rapidement un logiciel. Avec Docker, on peut gérer l’infrastruc-

ture de la même façon qu’on géré ses applications. En profitant des méthodologies de Docker pour l’embarquement, le test et le déploiement rapide du code, on peut réduire considérablement le délai entre l’écriture du code et son exécution en production.

**Compose** Compose est un outil permettant de définir et d’exécuter des applications Docker à conteneurs multiples. Avec Compose, nous pouvons utiliser un fichier YAML pour configurer les services de notre application. Ensuite, avec une seule commande, nous pouvons créer et démarrer tous les services de notre configuration. JOOD-CORE

fonctionne avec Docker Compose dans les processus de développement et de test .

**Git & GitHub** Github est un service web d’hébergement et de ges- tion de développement de logiciels, utilisant le logiciel de gestion de versions Git, amène chaque contributeur à télécharger les sources du projet et à proposer ensuite ses modifications à l’équipe du projet. Le

site offre aussi un logiciel de suivi de problèmes (issues tracking system). GitHub propose aussi l’inté- gration d’un grand nombre de services externes, tels que l’intégration continue (continous integration), la gestion de versions, badges, chat basés sur les projets, etc.

La figure3.[2](#_bookmark58)représente l’organisation Jood (de notre projet) sur GitHub.

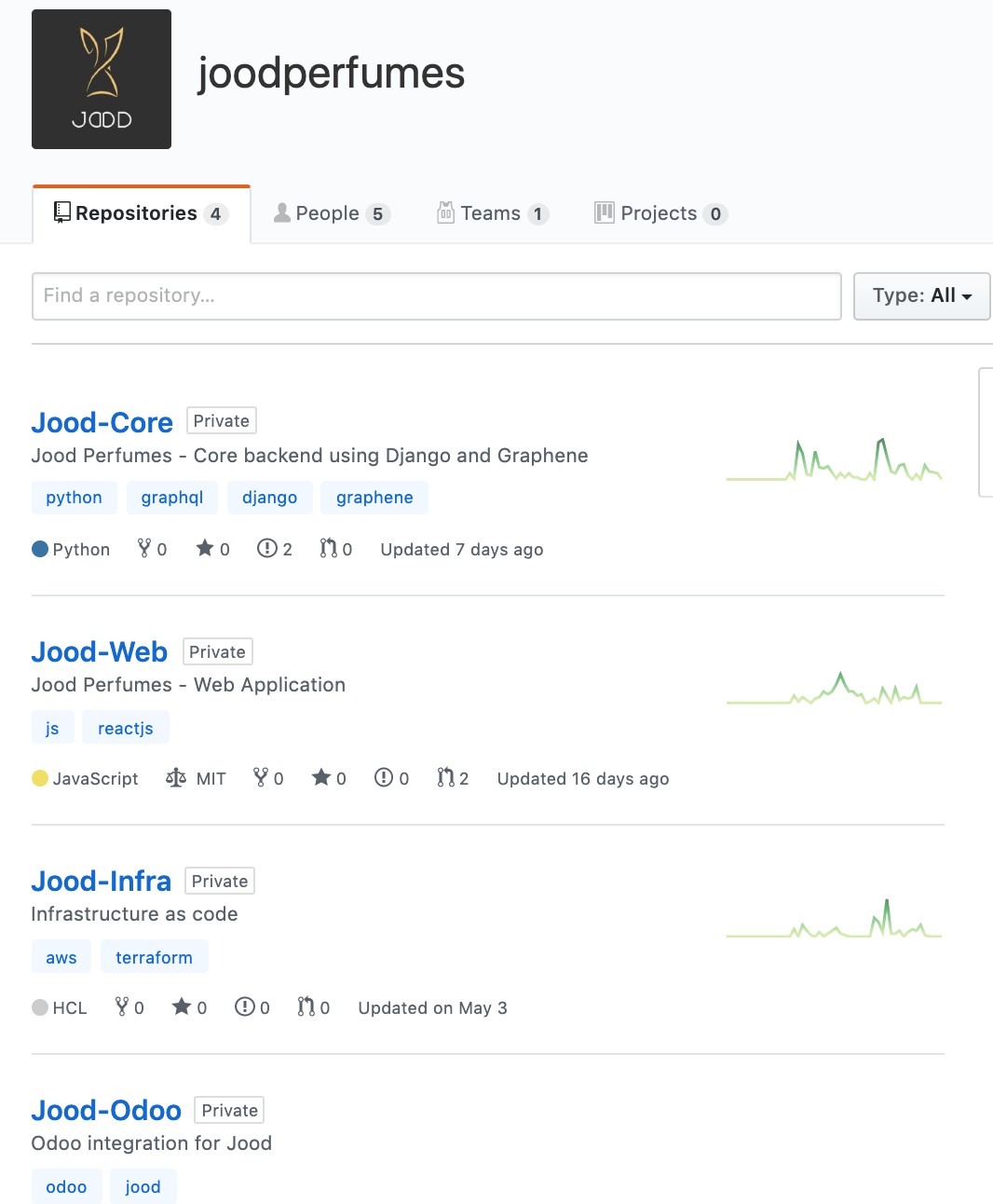
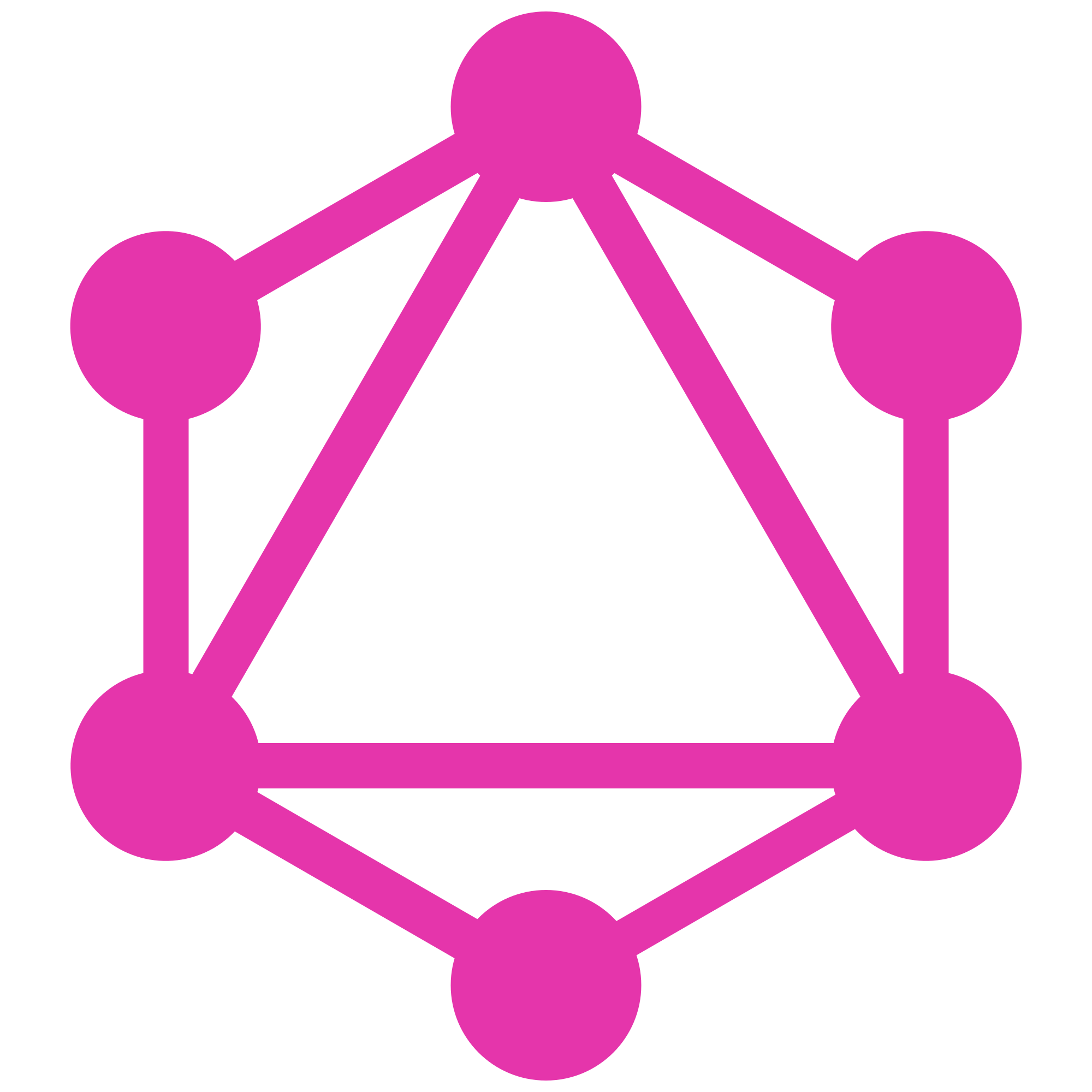


Figure 3.2: Aperçu de jood sur GitHub

**GraphQL-Graphene** GraphQL est un langage de requêtes déve- loppé en interne par Facebook en 2012 jusqu’à sa publication en 20152. Il propose une alternative aux API REST1 dont le stockage est éven- tuellement distribué. Il propose au client de formuler la structure de données dans la requête, tandis que cette même structure est retour-

née par le serveur. Fortement typé, ce langage évite les problèmes de retour de données insuffisants (under-fetching) ou surnuméraires (over-fetching).

GraphQL a été implémenté pour de nombreux langages, par exemple Graphene3 pour Python, Apollo4 et Relay5 pour Javascript.

Il est utilisé par certaines bases de données orientées graphe comme ArangoDB en tant que langage de requête6.

**Graphene** Graphene-Python est une bibliothèque permettant de créer facilement des API Gra- phQL en Python. Son objectif principal est de fournir une API simple mais évolutive facilitant la vie des développeurs.

**Celery** est une file d’attente de tâches / travaux asynchrone basée sur le transfert de messages à un courtier, tel que RabbitMQ . Il est axé sur le fonctionnement en temps réel, mais prend également en charge la planification. Les unités d’exécution, appelées tâches, sont exécutées simultanément sur un ou plusieurs serveurs de travail

utilisant le multitraitement, Eventlet ou gevent. Les tâches peuvent s’exécuter de manière asynchrone (en arrière-plan) ou synchrone (attendre d’être prêt).

Celery est un système Python Task-Queue qui gère la répartition des tâches sur les travailleurs via des threads ou des nœuds de réseau. Cela facilite la gestion des tâches asynchrones.

**RabbitMQ** RabbitMQ est une solution de messagerie orientée mes- sages, ou solution Message-Oriented Middleware (MOM). Un midd- leware est un logiciel tiers permettant de créer un réseau d’échange d’informations entre des applications. La technique d’échange d’infor- mations utilisée par RabbitMQ est l’échange de messages.

Le terme MQ, Message Queuing, présent dans RabbitMQ souligne cette technique et par ailleurs la principale fonctionnalité du produit. Il représente la connexion d’applications par le biais de messages routés grâce à un broker. On peut comparer le broker à La Poste, c’est-à-dire qu’il reçoit un message d’une application et le délivre à une autre. Nous utilisons RabbitMQ dans l’infrastructure JOOD parce qu’elle répond à ces exigences d’application modernes. Nous l’utilisons comme intermédiaire entre Celery et **Jood-Core** .

Jood-Core doit simplement envoyer des messages à RabbitMQ, et les employés de Celery les afficheront et planifieront l’exécution des tâches.

La figure3.3illustre un exemple d’Appel d’une Tache par l’application executer avec Celery et RabbitMQ.

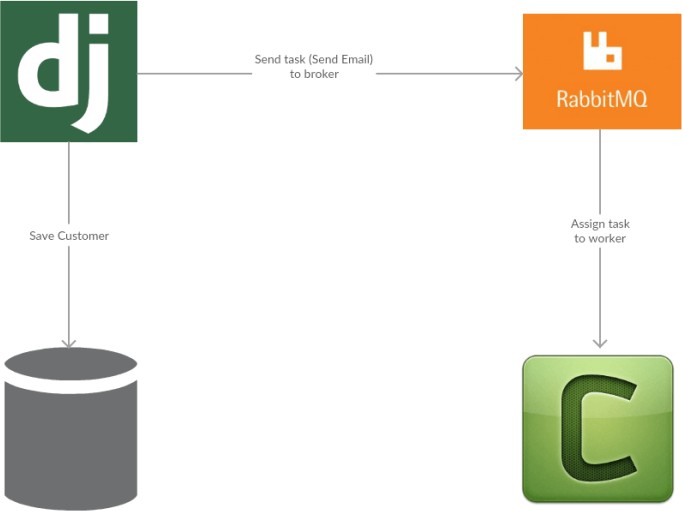


Figure 3.3: Appel d’une Tache celery

**XML-RPC** XML-RPC est un protocole RPC (remote procedure call), une spécification simple et une nsemble de codes qui permettent de s’exécuter dans des environnements différents defaire des appels de méthodes à travers un réseau.XML-RPC permet d’appeler une fonction sur un serveur distant de n’importe quel système(Windows, Mac OS X, GNU / Linux) et avec n’importe quel langage de programmation. Le serveur est lui-même sur n’importe quel système et est programmé dans n’importe quel langage.Cela permet de fournir un service Web utilisable par tout le monde sans restriction de système ou de langage.Les processus d’invocation à distance ont utilisé le protocole HTTP pour le transfert de données et la norme XML pour la structuration des données.XML-RPC est conçu pour permettre aux structures de données d’être transmis, exécutées et renvoyées très facilement.XML-RPC est une alternative aux services web WS- \*, ne SOAP.

On a utilisé XML-RPC dans JOOD-CORE pour communiquer avec JOOD-ODOO le serveurde JOOD dédié à ODOO

**Odoo** Odoo, anciennement OpenERP, est une société de logiciels open source fondée en 2004 qui propose une suite complète de modules de gestion d’entreprise entièrement intégrés.

Odoo est le programme de gestion d’entreprise le plus évolutif et le plus installé au monde avec des applications qui répondent à tous les besoins d’une entreprise, de la gestion de la relation client à la création de sites Web, en passant par le commerce électronique. commerce, par la production, la gestion des stocks, la comptabilité, les ERP, etc., parfaitement intégrés. C’est la première fois qu’un éditeur de logiciel atteint un tel niveau de fonctionnalité.

La figure3.[4](#_bookmark60)représente les principales fonctionalites de ERP Odoo



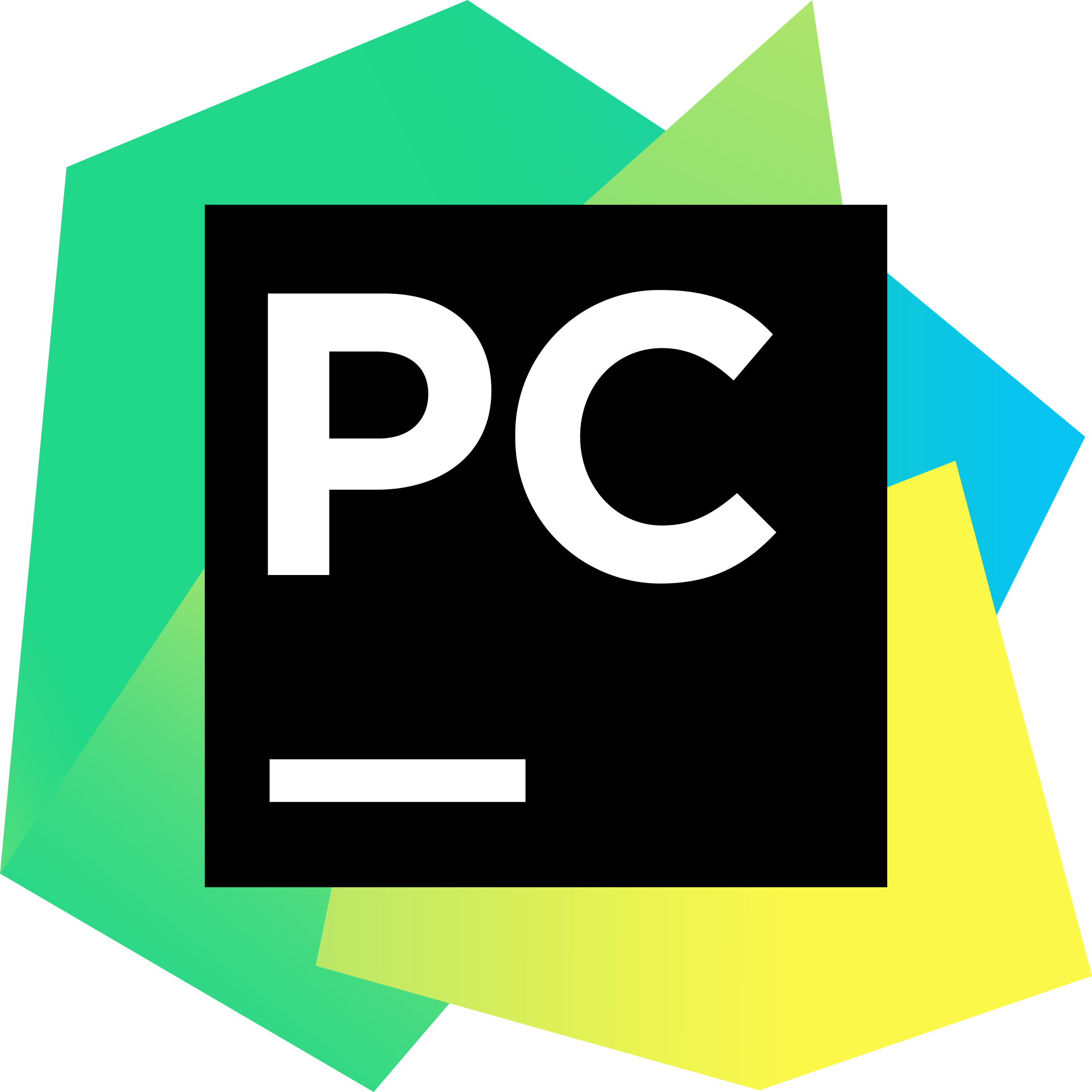
Figure 3.4: Les principales fonctionnalités de ERP Odoo

##### Pourquoi Odoo ?

* Solution ERP Open Source : le code source est disponible à tout moment, votre ERP n’est donc pas une boîte noire inaccessible ;
* il s’agit d’un ERP flexible : une mise en oeuvre progressive en fonction de l’évolution de vos besoins est donc tout à fait possible ;
* La solution ERP est très facilement interopérable avec tout logiciel et / ou service tiers (notam- ment via le connecteur générique développé par Camptocamp) ;
* S’appuie sur un éditeur solide de plus de 600 employés, une communauté dynamique et structurée (offrant plus de 20 000 modules de communauté) et un réseau mondial de plus de 1 000 partenaires certifiés.

**KNETPAY** est la solution de passerelle KNET Payment Gateway pour les marchands koweïtiens et les sociétés de services disposant de capacités de commerce électronique. Le service prendra en charge les transactions par carte bancaire de marque KNETco sur le Web.KNET présentera les cartes de crédit à l’avenir Le service KNET Payment

Gateway (PG) fournira une facilité de paiement pour les services marchands ou les biens. Les principales caractéristiques du service KNETPG sont lessuivantes : oPrise en charge des cartes de débit de marque KNETco sur une page depaiement sécurisée hébergée sur KNET oAutorisation en ligne des transactions de paiement via une intégration deplug-in marchand-gateway

**IDE : PYCharm** On attend principalement d’un IDE qu’il re- groupe dans une même interface les fonctionnalités indispensables à la réalisation et au suivi du projet tout au long de son cycle de vie. On y retrouve donc bien évidemment un éditeur performant simplifiant l’écriture de code, mais aussi idéalement l’intégration d’un débogueur,

l’interfaçage avec un outil de gestion de versions, l’interfaçage avec un outil de gestion de tâches, la prise en charge des tests unitaires ou encore des outils de déploiement. PyCharm intègre tout ceci de façon intuitive et efficace. Développé par l’équipe de JetBrains, c’est une équipe qui met dans ses produits tout leur savoir-faire et leur passion pour le développement informatique. Tout en étant très orienté professionnel.

### Outils de Tests

Tester les applications est devenu un ensemble de compétences standard requis par toutdévelop- peur compétent. La communauté Python accepte les tests et même la bibliothèque standard Python possède de bons outils intégrés pour prendre en charge les tests. Dans le grand écosystème Python, il existe de nombreux outils de test.

Les tests ne font pas exception. Pour les monolithes, il existe des tests unitaires, des tests decomposants et des tests d’intégration. Les limites sont claires, la façon d’écrire des testsest également claire.

**PYTEST – UNIT TESTING** PyTest est un outil de test Python robuste qui peut être utilisé pour tous les types et niveaux de tests logiciels. PyTest peut être utilisé par les équipes de développement, les équipes d’assurance qualité, les groupes de test indépendants, les utilisateurs du logiciel TDD et les projets opensource. En fait, des

projets partout sur Internet sont passés de test unitaire , notamment Mozilla et Dropbox. Pourquoi ? Parce que PyTest offre des fonctionnalités puissantes telles que la réécriture «assert», un modèle de plug-intiers et un modèle de montage puissant et simple, inégalé dans aucun autre framework de test.Dans **OBytes**, chaque développeur est à la fois un testeur et un développeur. Lors de la création d’une nouvelle fonctionnalité pour Jood, nous nous assurons que les tests sont écrits en premier, puis nous développons la fonctionnalité.

**VCRPY – SERVICE MOCKING** VCR.py simplifie et accélère les tests des requêtes HTTP. La première fois que nous exécutons du code contenu dans un gestionnaire de contexte VCR.py ou une fonctiondécorée, VCR.py enregistre toutes les interactions HTTP ayant lieu par le biais des biblio- thèques. Il prend en charge, sérialise et écrit dans un fichier plat (au format yamlpar défaut). ). Ce fichier plat s’appelle une cassette.Lorsque le code correspondant est exécuté à nouveau, VCR.py lit les requêtes et les réponses sérialisées à partir du fichier de cassette susmentionné, intercepte les requêtes

HTTP qu’il reconnaît à partir de l’exécution du test d’origine et renvoie les réponses correspondant à ces requêtes. Cela signifie que les demandes ne généreront pas de trafic HTTP, ce qui présente plusieurs avantages, notamment :

* La possibilité de travailler hors ligne
* Tests complètement déterministes
* Augmentation de la vitesse d’exécution du test

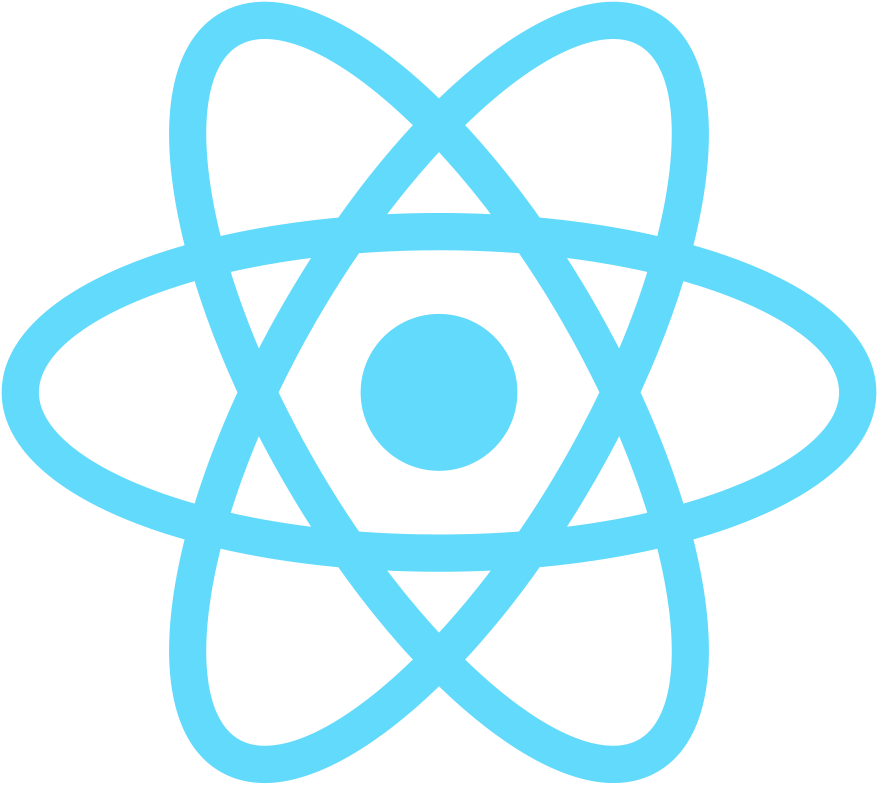
Si le module que nous testons modifie constamment son API, il suffit de supprimer nos fichiers de cassettes existants et de réexécuter nos tests. VCR.py détectera l’absence de fichier de cassette et enregistrera à nouveau toutes les interactions HTTP, qui les actualisera pour correspondre à la nouvelle API.Nous utilisons vcr.py pour simuler des appels http a une externe API.

**FAKER – DATA GENERATION** Lors de l’écriture de tests unitaires, nous rencontrons une situation dans laquelle nous devons générer des données de test ou utiliser des données factices dans nos tests. Si nous avons déjà des données quelque part dans une base de données, une solution que nous pourrions utiliser consiste à générer un vidage de ces données et à les utiliser dans nos tests (par exemple, les fixtures).Cependant, nous pourrions également utiliser un paquet tel que faker pour générer très facilement de fausses données lorsque nous en aurons besoin.Le seul but de Faker est de créer de fausses données semi-aléatoires. Faker peut créer de faux noms, adresses, agents utilisateurs de navigateur, domaines, paragraphes et bien plus encore.

**CODECOV – CODE COVERAGE** CODECOV décrit le degré de test du code source d’un programme. Nous utilisons CodeCov pour mesurer la couverture de code du notre application . Il surveille le projet , notant quelles parties du code ont été exécutées, puis analyse la source pour identifier le code qui aurait pu être exécuté mais ne l’a pas été.La mesure de la couverture est généralement utilisée pour évaluer l’efficacité des tests. Il peut indiquer quelles parties du code sont soumises à des tests et les quelles ne le sont pas.JOOD Core code doit avoir une couverture de 98% pour être fusionné à la branche principale.

**CONFTEST – Definir des Fixtures** Utiliser PyTest est génial et le support pour les montages de test est assez impressionnant. Cependant, afin de partager nos appareils sur l’ensemble du mo- dule,PyTest suggère de définir tous les appareils dans un seul fichier conftest.py. Ce n’est paspratique si nous avons un grand nombre de fixtures. Pour une meilleure organisation et une meilleure lisibilité, nous préférerions définir nos fixtures sur plusieurs fichiers bien nommés .

### Outils de development Front-end

**React** aussi appelé (React.js ou ReactJS) est une bibliothèque Ja- vaScript libre développée par Facebook depuis 2013. Le but principal de cette bibliothèque est de faciliter la création d’application web mo- nopage, via la création de composants dépendant d’un état et générant une page (ou portion) HTML à chaque changement d’état. React est

une bibliothèque qui ne gère que l’interface de l’application, considéré comme la vue dans le modèle MVC. Elle peut ainsi être utilisée avec une autre bibliothèque ou un framework MVC comme An- gularJS. La bibliothèque se démarque de ses concurrents par sa flexibilité et ses performances, en travaillant avec un DOM virtuel et en ne mettant à jour le rendu dans le navigateur qu’en cas de nécessité.

La bibliothèque est utilisée notamment par Netflix (pour gagner 50 % de performance), Yahoo, Airbnb, Sony, Atlassian ainsi que par les équipes de Facebook, appliquant le dogfooding sur le réseau social éponyme, Instagram ou encore WhatsApp. À la fin de 2015, WordPress.com annonce Calypso, une interface pour les administrateurs WordPress, développée en JavaScript avec node.js et React

##### GatsbyJS

**Les sites Web statiques** sont des un sites dont le contenu est fixe. Techniquement, il est composé d’une simple liste de fichiers HTML, qui affiche les mêmes informations quel que soit le visiteur.

Contrairement aux sites dynamiques, il ne requiert ni programmation back-end ni base de données. Pour faciliter leur création, de nombreux générateurs de sites statiques sont disponibles : Jekyll, Hugo, Hexo, etc. La plupart du temps, le contenu est géré via des fichiers (idéalement en format Markdown) statiques ou via une API de contenu. Ensuite, le générateur requête le contenu, l’injecte dans les templates définis par le développeur et génère un ensemble de fichiers HTML, CSS et JavaScript.

**Les Progressive Web Apps** (PWA) sont des applications web, reposant fortement sur l’uti- lisation de JavaScript, qui se veulent être fiables, rapides et engageantes. Étant donné qu’elles rendent la navigation plus rapide et qu’elles offrent une expérience utilisateur bien meilleure, les PWA sont devenues la manière par défaut de construire des interfaces web. Pour cette raison, de nombreux fra- meworks front-end ont fait leur apparition au cours des dernières années : Angular, React, et plus récemment, Vue.

**Gatsby** : quand les sites statiques rencontrent les Progressive Web Apps. Les sites Web statiques et PWA ont tous les deux de solides avantages, ce qui nous incite à trouver un moyen de les utiliser tous les deux dans le même projet. Heureusement, quelques outils ont réussi à faire le pont entre les deux, notamment celui dont on entend le plus parler depuis quelques mois : **Gatsby**

**Gatsby** est un générateur de site statique (SSG) basé sur React. Il s’agit d’un projet open Source (GitHub) né en en mai 2015 auquel ont contribués plus de 1350 développeurs en 7000 commits à ce jour. Début 2018 Kyle Mathews crée Gatsby Inc pour soutenir le développement du projet.

##### ECMAScript 6 / JavaScript

**ECMAScript** ECMAScript 2015 est la version actuelle pour le langage ECMAScript. C’est le standard qui définit l’implémentation de JavaScript, notamment pour SpiderMonkey, le moteur utilisé par Firefox et les autres applications Mozilla. Cette version est appelée avec le nom de code « ES.next », « Harmony » ou « ECMAScript

6 ». Les brouillons de spécification (aussi appelés drafts) se trouvent sur le wiki officiel d’ECMA. Le premier draft basé sur ECMAScript 5.1, a été publié le 12 juillet 2011 sous le nom « ES.next ». À partir d’août 2014, ECMAScript 2015 n’a plus accepté de nouvelles fonctionnalités et a été stabilisé. Il a été officiellement approuvé et fut publié le 17 juin 2015 par l’assemblée générale ECMA.

**JavaScript** est un langage de programmation de scripts prin- cipalement employé dans les pages web interactives mais aussi pour les serveurs, avec l’utilisation (par exemple) de Node.js.

C’est un langage orienté objet à prototype, c’est-à-dire que les bases du langage et ses principales interfaces sont fournies par des ob-

jets qui ne sont pas des instances de classes, mais qui sont chacun équipés de constructeurs permettant de créer leurs propriétés, et notamment une propriété de prototypage qui permet d’en créer des objets héritiers personnalisés. En outre, les fonctions sont des objets de première classe. Le langage supporte le paradigme objet, impératif et fonctionnel. JavaScript est le langage possédant le plus large écosystème grâce à son gestionnaire de dépendances npm, avec environ 500 000 paquets en août 2017.

**NodeJS** est une plateforme logicielle libre et événementielle en Ja- vaScript orientée vers les applications réseau qui doivent pouvoir mon- ter en charge. Elle utilise la machine virtuelle V8 et implémenté sous licence MIT les spécifications CommonJS. Parmi les modules natifs de Node.js, on retrouve http qui permet le développement de serveur

HTTP. Il est donc possible de se passer de serveurs web tels que Nginx ou Apache lors du déploiement de sites et d’applications web développés avec Node.js. Concrètement, Node.js est un environnement bas niveau permettant l’exécution de JavaScript côté serveur. Node.js est utilisé notamment comme plateforme de serveur Web, elle est utilisée par Groupon, Vivaldi, SAP, LinkedIn, Microsoft, Yahoo !, Walmart, Rakuten, Sage et PayPal.

**Flux** est l’architecture applicative utilisée par Facebook pour la construction d’applications Web côté client. Il complète les composants de vue composables de React en utilisant un flux de données unidirectionnel. Il s’agit plus d’un modèle que d’un cadre formel et vous pouvez commencer à utiliser Flux immédiatement sans beaucoup de nouveau code.

* **Les actions** sont des objets simples avec une propriété de type et des données. Par exemple, une action peut être type : IncrementCount, données\_local : delta : 1
* **Le Stores** contiennent l’état et la logique de l’application. La meilleure abstraction est de penser aux stores comme à la gestion d’un domaine particulier de l’application
* **Le Dispatcher** agit comme un hub central. Le répartiteur traite les actions (par exemple, les interactions utilisateur) et invoque les callbacks que les stores ont enregistrés.
* **Les vues** sont très courantes dans la plupart des modèles d’interface graphique MVC. Ils écoutent les changements dans les Stores et se rendent de manière appropriée. Les vues peuvent également ajouter de nouvelles actions au dispatcher, par exemple lors d’interactions entre utilisateurs.

**Redux** est un conteneur du “state” pour les applications JavaScript. Il vous aide à écrire des applications qui se comportent de manière cohérente, fonctionnent dans différents environnements (client, serveur et natif) et sont faciles à tester. En plus de cela, il fournit une grande expérience de développement, telle que l’édition de code en temps réel

combinée à un débogueur temporel. Redux peut être utilisé avec React ou avec toute autre bibliothèque de vues. Il est minuscule (2 Ko, dépendances comprises), mais dispose d’un vaste écosystème d’addons disponibles.

**JSX** React peut s’écrire avec la “véritable syntaxe” JavaScript mais cela est moins facile à écrire et encore moins à lire. JSX apporte un véritable souplesse dans l’écriture, dans la lecture et la compréhension. Babel propose un preset pour JSX ce qui permet de transpiler le JSX en code JavaScript classique et exécutable par le navigateur.

JSX est une extension de syntaxe de type XML à ECMAScript sans sémantique définie. Il n’est pas destiné à être implémenté par les moteurs ou les navigateurs. Ce n’est pas une proposition d’intégrer JSX dans la spécification ECMAScript elle-même. Il est destiné à être utilisé par divers préprocesseurs (transpilers) pour transformer ces jetons en ECMAScript standard.

**pourquoi JSX ?** React comprend le fait que la logique de rendu est intrinsèquement couplée à une autre logique d’interface utilisateur : mode de gestion des événements, évolution de l’état dans le temps et mode de préparation des données pour l’affichage.

Au lieu de séparer artificiellement les technologies en plaçant les balises et la logique dans des fichiers séparés, React dissocie les problèmes avec des unités faiblement couplées appelées «composants»

qui contiennent les deux. React ne nécessite pas l’utilisation de JSX, mais la plupart des gens le trouvent utile comme aide visuelle lorsqu’on travaille avec une interface utilisateur dans le code JavaScript. Cela permet également à React d’afficher des messages d’erreur et d’avertissement plus utiles.

# 4

## L’infrastructure cloud de joodpefumes

### Introduction

***Dans ce chapitre, je présenterai le Workflow d’infrastructure et d’exploitation de JoodPerfumes, la première section sera consacrée à une introduction sur le cloud com- puting. Dans la deuxième partie j’expliquerai les technologies utilisées pour la gérer et l’organisation des réseaux, services, clusters, aux fournisseurs de cloud que nous uti- lisons et à la sécurisation de l’infrastructure de Joodperfumes.Enfin, je traiterai de la dernière section l’architecture cloud de Joodperfumes .***

### Qu’est-ce que le cloud computing

Pour simplifier, **le cloud** computing est la fourniture de services informatiques (notamment des serveurs, du stockage, des bases de données, la gestion réseau, des logiciels, des outils d’analyse, l’intel- ligence artificielle) via Internet **le cloud** dans le but d’offrir une innovation plus rapide, des ressources flexibles et des économies d’échelle. En règle générale, vous payez uniquement les services cloud que vous utilisez (réduisant ainsi vos coûts d’exploitation), gérez votre infrastructure plus efficacement et adaptez l’échelle des services en fonction des besoins de votre entreprise.

#### Principaux avantages du cloud computing

Le cloud computing est radicalement différent de l’approche traditionnelle que les entreprises adoptent en matière de ressources informatiques. Voici sept raisons courantes pour lesquelles les orga- nisations optent pour des services de cloud computing :

* + - * **Coût** : Le cloud computing élimine la nécessité d’investir dans du matériel et des logiciels, et de configurer et de gérer des centres de données sur site : racks de serveurs, alimentation électrique permanente pour l’alimentation et le refroidissement, experts informatiques pour la gestion de l’infrastructure. La facture est vite salée.
      * **Vitesse** : La plupart des services de cloud computing sont fournis en libre-service et à la demande. D’énormes ressources de calcul peuvent donc être mises en œuvre en quelques minutes et en quelques clics, offrant ainsi aux entreprises un haut niveau de flexibilité et les dégageant de la pression liée à la planification de la capacité.
      * **Mise à l’échelle mondiale** : La mise à l’échelle élastique est un des avantages des services de cloud computing. En termes de cloud, cela veut dire qu’il est possible de mettre en œuvre

la quantité nécessaire de ressources informatiques, par exemple plus ou moins de puissance de calcul, de stockage ou de bande passante, au moment où elles sont nécessaires, là où elles sont nécessaires.

* + - * **Productivité** : Les centres de données sur site nécessitent en général la manipulation de matériel, la mise à jour des logiciels et d’autres corvées informatiques qui prennent beaucoup de temps. Le cloud computing supprime la plupart de ces tâches et les équipes informatiques peuvent donc passer plus de temps à travailler à la concrétisation des objectifs de l’entreprise.
      * **Performances** : Les plus grands services de cloud computing s’exécutent sur un réseau de centres de données sécurisés, dont le matériel est régulièrement mis à niveau pour assurer des performances rapides et efficaces. Ceci offre plusieurs avantages par rapport à un centre de don- nées classique, y compris un temps de latence réseau réduit pour les applications et de plus grandes économies d’échelle.
      * **Fiabilité** : Le cloud computing simplifie la sauvegarde des données, la récupération d’urgence et la continuité des activités. Il rend ces activités moins coûteuses, car les données peuvent être mises en miroir sur plusieurs sites redondants au sein du réseau du fournisseur.
      * **Sécurité** : De nombreux fournisseurs de cloud offrent un vaste éventail de stratégies, technologies et contrôles qui renforcent globalement votre situation de sécurité, contribuant ainsi à protéger vos données, vos applications et votre infrastructure contre des menaces potentielles.

#### Types de cloud computing

Tous les clouds ne sont pas identiques et aucun type de cloud computing ne convient à tout le monde. Plusieurs modèles, types et services différents ont évolué pour vous aider à trouver la solution adaptée à vos besoins.

Vous devez commencer par déterminer le type de déploiement cloud ou d’architecture de cloud computing sur lequel vos services cloud seront implémentés. Il existe trois modes de déploiement de services cloud : le cloud public, le cloud privé et le cloud hybride.

**Cloud public :** Public clouds are owned and operated by a third-party cloud service providers, which deliver their computing resources, like servers and storage, over the Internet. Microsoft Azure is an example of a public cloud. With a public cloud, all hardware, software, and other supporting infrastructure is owned and managed by the cloud provider. You access these services and manage your account using a web browser.

**Cloud privé :** Le cloud privé est l’ensemble des ressources de cloud computing utilisées de façon exclusive par une entreprise ou une organisation. Le cloud privé peut se trouver physiquement dans le centre de données local de l’entreprise. Certaines entreprises paient également des fournisseurs de services pour qu’ils hébergent leur cloud privé. Le cloud privé est un cloud dans lequel les services et l’infrastructure se trouvent sur un réseau privé.

**Cloud hybride :** Le cloud hybride regroupe des clouds publics et privés, liés par une technologie leur permettant de partager des données et des applications. En permettant que les données et applications se déplacent entre des clouds privé et public, un cloud hybride offre à votre entreprise une plus grande flexibilité, davantage d’options de déploiement et une optimisation de votre infrastructure, de votre sécurité et de votre conformité existantes.

#### Types de services cloud : IaaS, PaaS, SaaS

La plupart des services de cloud computing peuvent être classés en quatre grandes catégories : IaaS (infrastructure as a service), PaaS (platform as a service), serverless et SaaS (software as a service). On les appelle parfois « pile » de cloud computing, car elles s’empilent les unes sur les autres. Si vous

savez en quoi elles consistent et en quoi elles sont différentes, vous pourrez plus facilement atteindre vos objectifs.

**Infrastructure as a service (IaaS) :** La catégorie la plus basique des services de cloud computing. Avec l’IaaS, vous louez une infrastructure informatique (serveurs, machines virtuelles, stockage, réseaux, systèmes d’exploitation) auprès d’un fournisseur de services cloud, avec un paiement à l’utilisation

**Platform as a service (PaaS) :** L’expression plateforme en tant que service (PaaS, Platform-as- a-Service) qualifie les services de cloud computing qui offrent un environnement à la demande pour développer, tester, fournir et gérer des applications logicielles. PaaS est conçu pour permettre aux développeurs de créer rapidement des applications web ou mobiles sans avoir à se préoccuper de la configuration ou de la gestion de l’infrastructure de serveurs, de stockage, de réseau et de bases de données nécessaire au développement.

**Software as a service (SaaS) :** Le logiciel en tant que service (SaaS, Software-as-a-Service) est une méthode de diffusion d’applications logicielles via Internet, à la demande et en général sur abon- nement. Avec le SaaS, les fournisseurs de services cloud hébergent et gèrent les applications logicielles et l’infrastructure sous-jacente, et gèrent la maintenance, par exemple la mise à niveau des logiciels et l’application des correctifs de sécurité. Les utilisateurs se connectent à l’application via Internet, en général par l’intermédiaire d’un navigateur web sur leur téléphone, leur tablette ou leur PC.

### le choix de AWS

Amazon Web Services (AWS) et Google Cloud sont des plates-formes pour fournir des services de cloud computing aux entreprises et aux particuliers. Nous avons choisi AWS parce qu’il est plus efficace que Google Cloud dans plusieurs services que nous allons maintenant mentionner :

La première catégorie concerne la façon dont leurs machines virtuelles (instances) sont traitées dans la technologie Google Cloud Virtual Machine est KVM, tandis que la technologie derrière les machines virtuelles AWS EC2 est Xen.

Stockage / Disque : le type de stockage et les disques utilisés par un fournisseur de cloud jouent un rôle très important car ils ont un impact direct sur les performances, tels que le débit attendu (IO), le nombre maximal d’E / S par volume / instance et possibilité d’augmenter la capacité sur de courtes périodes. Lorsque vous comparez Google et AWS, il existe deux principaux types d’options de stockage : le stockage en masse et le stockage d’objets.

La capacité réseau disponible sur les instances de Google Compute Engine fonctionne légèrement différemment, car elle dépend du nombre de processeurs de vos machines virtuelles. Chaque noyau est soumis à une limite de 2 Gbits / seconde (Gbps) pour des performances optimales. Chaque noyau supplémentaire augmente le plafond du réseau, jusqu’à un maximum théorique de 16 Gbps pour chaque machine virtuelle.

Les instances Amazon EC2 ont une bande passante maximale de 25 Gbps, mais cela ne s’applique qu’aux instances plus grandes. Nombre maximal d’instances standard à 10 Gbps / seconde.

la figure suivante montre les Tendances du Cloud Computing sur Stack Overflow dans lesquels ils analysent des modèles basés sur le % de questions d’un mois.

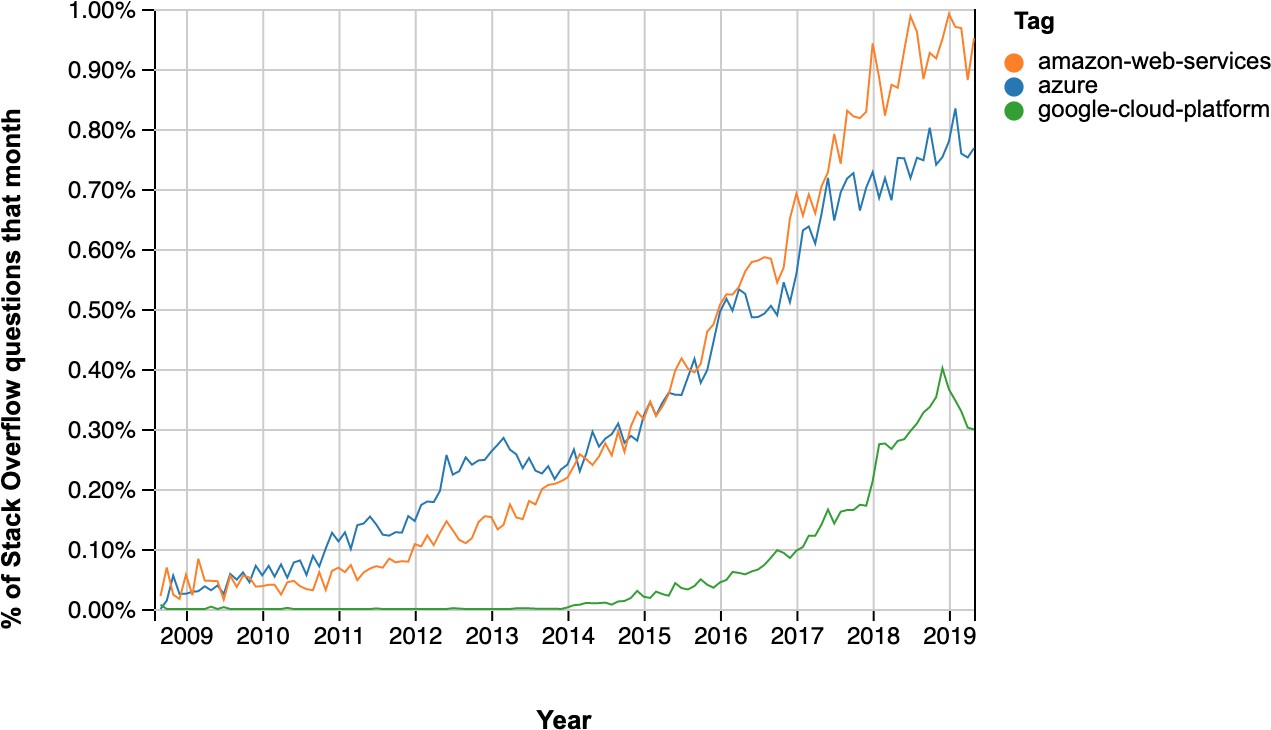


Figure 4.1: Tendances du Cloud Computing sur Stack Overflow

Nous avons aussi opté pour AWS car dans l’entreprise ils ont déjà travaillé avec google cloud mais malheureusement ils ont eu beaucoup de problèmes soit avec les serveurs qui tombent en panne soit avec la performance qui est faible

### Infrastructure en tant que code

Parfois appelée "infrastructure programmable," l’infrastructure en tant que code **IaC** aborde la configuration d’infrastructure exactement de la même manière que la programmation de logiciel. En effet, elle commence par estomper les limites entre la rédaction d’applications et la création des environnements où celles-ci sont exécutées. Les applications peuvent contenir des scripts qui créent et orchestrent leur propres machines virtuelles (VM). C’est une partie fondamentale du **Cloud Compu- ting** et essentielle aux **DevOps**.

L’infrastructure en tant que code permet une gestion programmable de machines virtuelles, ce qui élimine le besoin en configuration manuelle et en mises à jour des composantes matérielles. Cela rend cette infrastructure hautement "élastique", c’est-à-dire reproductible et évolutive. Un opérateur peut déployer et gérer 1 machine ou 1 000 machines avec le même ensemble de codes. La vitesse, l’économie de coûts et la réduction des risques sont les conséquences directes de l’infrastructure en tant que code.

L’infrastructure en tant que code supporte **IaaS** en permettant l’accès à des machines virtuelles et à des outils de gestion s’appuyant sur des logiciels conviviaux, regroupés et vendus au sein d’un service.

Actuellement, toutes les ressources d’infrastructure sont gérées à l’aide de terraform, le code est stocké dans un référentiel GitHub.

### Terraform - Concepts

Dans cette section, nous allons présenter les principaux concepts qui composent Terraform. Les fi- chiers de configuration s’écrivent en HCL (HashiCorp Configuration Language). Le principe est d’écrire des ressources.

Les ressources peuvent être écrites en JSON également, mais il est recommandé de les écrire en

HCL.

La figure4.[2](#_bookmark73)represente le compsants dune ressource avec terraform :

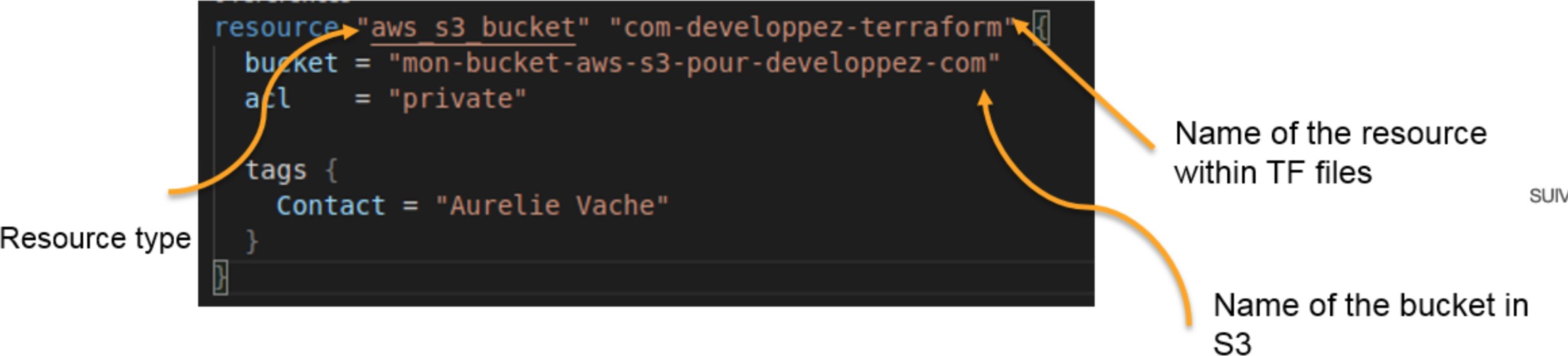


Figure 4.2: HashiCorp Configuration Language

L’exemple ci-dessous montre que, en quelques lignes de code, on va pouvoir déployer/créer un bucket S3 sur AWS :

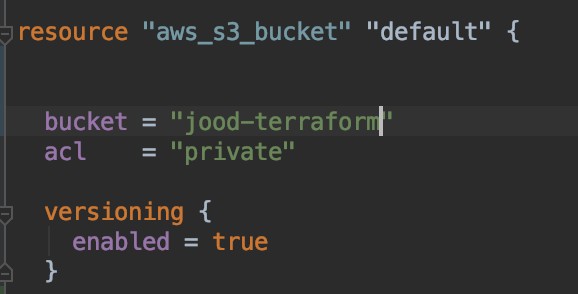
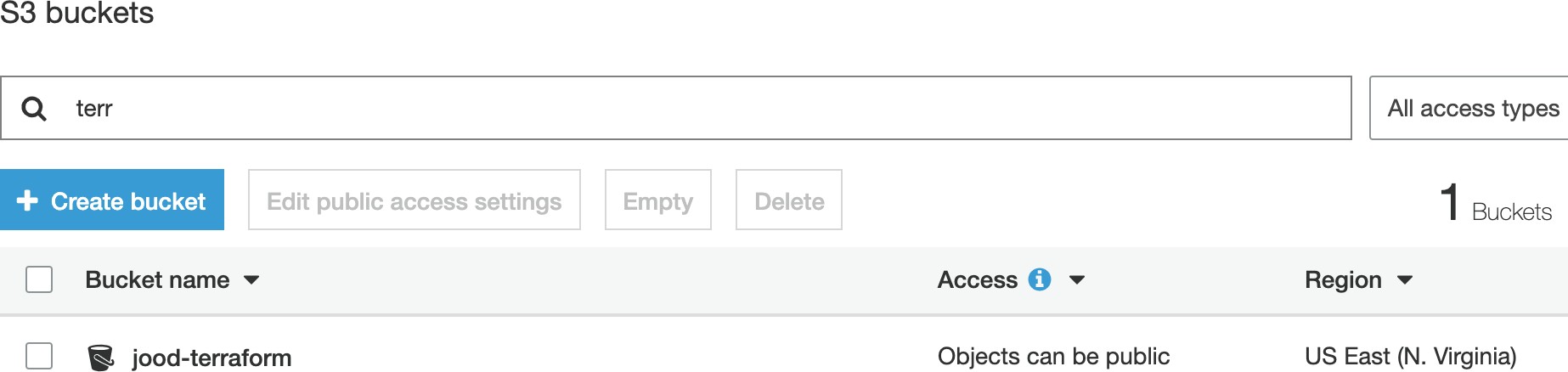


Figure 4.3: Creation de S3 bucket avec terraform La figure4.[4](#_bookmark75)represente le Résultat Terraform code sur AWS :

Figure 4.4: Résultat sur AWS

#### Terraform - Fournisseur (Provider)

Un fournisseur (provider) est responsable du cycle de vie/du CRUD (create, read, update, delete) d’une ressource : sa création, sa lecture, sa mise à jour et sa suppression.

Plus de 125 fournisseurs (providers) sont actuellement supportés :

* AWS, GCP, Azure, OpenStack. . . ;
* Heroku, OVH, 1&1. . . ;
* Consul, Chef, Vault. . . ;
* Consul, Chef, Vault. . . ;
* GitLab, BitBucket, GitHub. . . ;
* MySQL, PostgreSQL. . . ;
* mais encore RabbitMQ, DNSimple, CloudFlare. . .

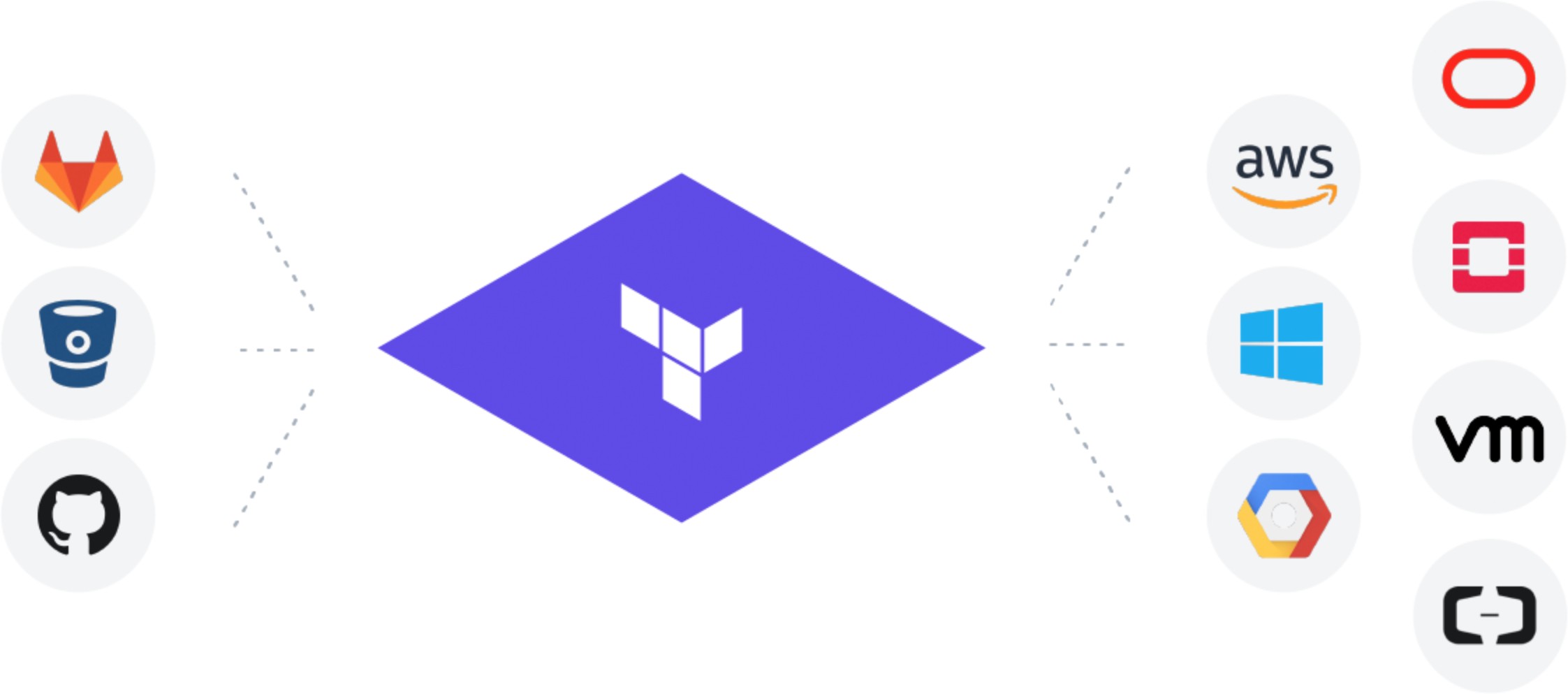


Figure 4.5: Exemples de fournisseurs Terraform

#### Terraform - Variables

Afin de faire du code propre et qui se réutilise, il est recommandé d’initialiser des variables et de les utiliser dans les autres fichiers .tf.

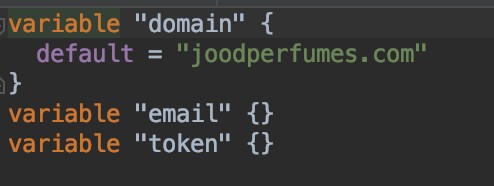


Figure 4.6: Variables avec Terraform

#### Terraform - Modules

Les modules sont utilisés pour créer des composants réutilisables, améliorer l’organisation et traiter les éléments de l’infrastructure comme une boite noire.

C’est un groupe de ressources qui prennent en entrée des paramètres et retournent en sortie des outputs. Dans le même fichier, fichier.tf, dans lequel vous avez défini votre fournisseur, vous pouvez ensuite définir vos modules :



Figure 4.7: Modules avec Terraform

#### Terraform - Outputs

Les modules peuvent produire des outputs que l’on pourra utiliser dans d’autres ressources.

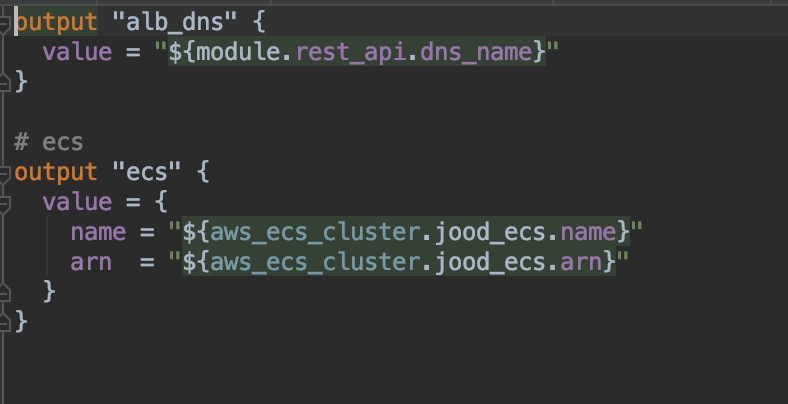


Figure 4.8: Outputs avec Terraform

La figure4.[11](#_bookmark87)represente l’utilisation d’une output dans une autre ressource avec terraform :

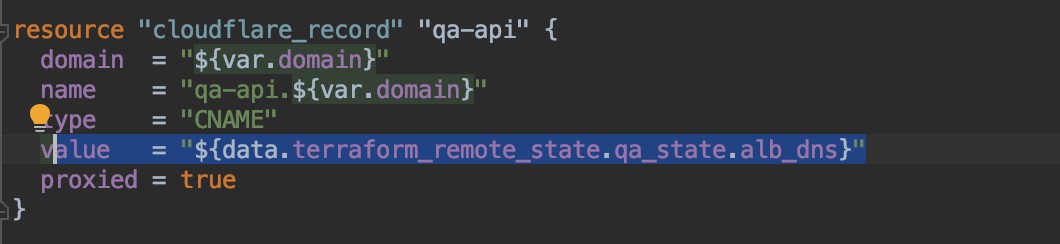


Figure 4.9: Appel d’une Output avec Terraform

#### Terraform - Sources de données (Data Sources)

Les sources de données (Data Sources) servent à récupérer une donnée/ressource existante dans votre infrastructure.

Utiliser une source de données est une bonne pratique. En effet, il est bien de privilégier son utilisation plutôt que de coder en dur des variables ou autre donnée qui peut être récupérée !



Figure 4.10: Data sources avec Terraform

Et ensuite vous pouvez appeler cette source de données simplement dans vos fichiers .tf comme

ceci :



Figure 4.11: Appel d’une source de donnees avec Terraform

#### Terraform - Etat (State)

Un état (state) est un snapshot de votre infrastructure depuis la dernière fois que vous avez exécuté la commande terraform apply.

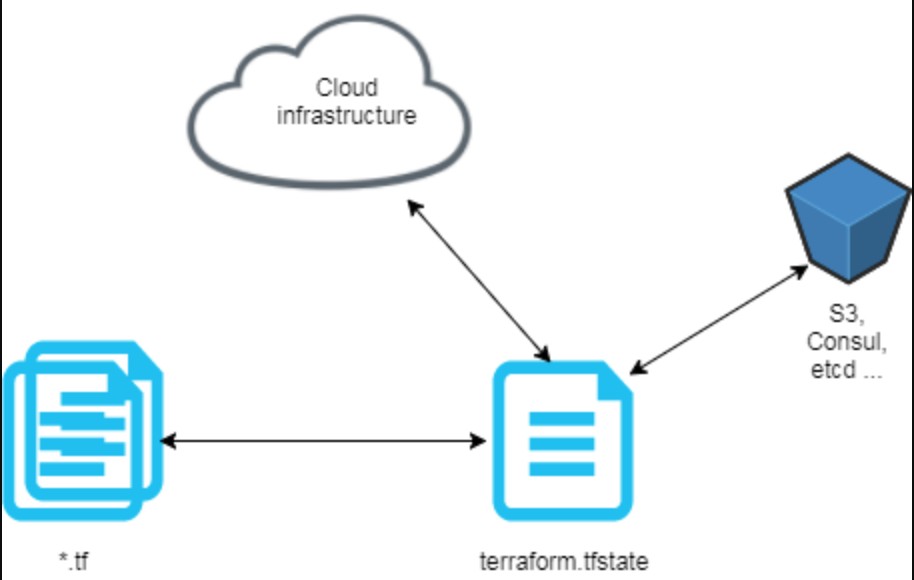


Figure 4.12: état (state) avec Terraform

Terraform utilise un stockage local pour créer les plans et effectuer les changements sur votre infrastructure, mais il est possible de stocker cet état dans une solution cloud.

Pour configurer le stockage de cet état de manière distante, il suffit de définir un backend.

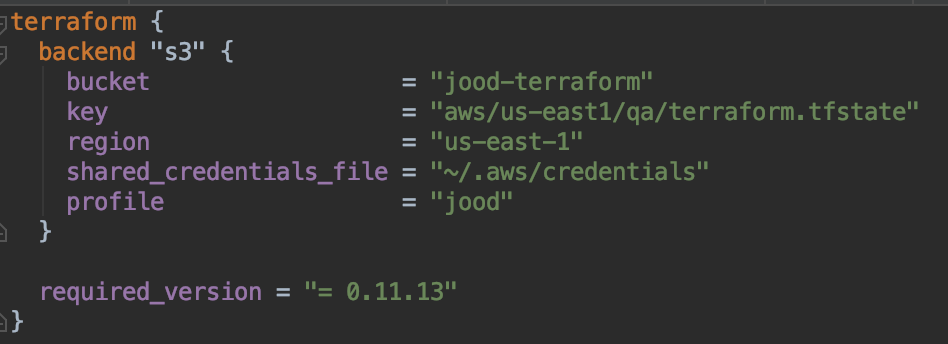


Figure 4.13: le stockage d’un etat avec Terraform

#### Terraform - CLI

Terraform a une CLI (Command-Line Interface) facile à utiliser et composée de plusieurs com- mandes : nous allons en voir quelques-unes, pas toutes, uniquement celles que j’utilise le plus au quotidien.

* + - * **terraform init :** La commande init va initialiser votre répertoire de travail qui contient vos fichiers de configuration au format .tf.

C’est la première commande à exécuter pour une nouvelle configuration, ou après avoir fait un checkout d’une configuration existante, depuis votre dépôt git par exemple.

La commande init va :

* + - * + télécharger et installer les fournisseurs (providers) ;
        + initialiser le backend (si défini) ;
        + télécharger et installer les modules (si définis).
      * **terraform plan :** La commande plan permet de créer un plan d’exécution. Terraform va dé- terminer quelles actions il doit entreprendre afin d’avoir les ressources listées dans les fichiers de configuration par rapport à ce qui est actuellement en place sur l’environnement/le fournisseur cible.

Cette commande n’effectue concrètement rien sur votre infrastructure.

* + - * **terraform apply :** La commande apply, comme son nom l’indique, permet d’appliquer les changements à effectuer sur l’infrastructure. C’est cette commande qui va créer nos ressources.

### Infrastructure de Jood

L’infrastructure Joodperfumes.com (ou Jood) est entièrement gérée par Dockers. Chaque com- posant est censé fonctionner dans un environnement Docker et géré par AWS ECS Fargate . L’infra- structure est actuellement hébergée sur Aamzone web service (AWS) alors que le domaine DNS utilise Cloudflare. Les détails sur l’automatisation et les configurations d’infrastructure sont suivis dans Jood- infra, fonctionnant actuellement dans une zone unique avec la configuration en place pour s’étendre à d’autres zones et / ou fournisseurs de cloud

#### Jood-Services de calcul et de mise en réseau

* + - * **Amazon VPC** : Amazon Virtual Private Cloud nous permet de lancer et ressembler les res- sources de Jood sur un réseau virtuel sur l’infrastructure du jood on a deux réseaux :
        + admin vpc : Ce réseau virtuel rassemble les composants communs entre la zone de production et la zone de staging tell que bastion host et le vpn
        + staging vpc : Ce réseau contient les principals elements de jood (Docker conteneurs , Ins- tances EC2) ainsi que les outils pour les gerer.

|  |  |
| --- | --- |
| **Réseaux** | **Cidr proposé** |
| qa-jood-useast1-vpc | 10.241.0.0/16 |
| adm-jood-useast1-vpc | 10.241.0.0/16 |

Table 4.1: Les réseaux de l’infrastructure Jood

* + - * **Amazon EC2** : Amazon Elastic Compute Cloud est un service Web qui fournit une capacité de calcul sécurisée et redimensionnable dans le cloud. il est conçu pour faciliter l’accès aux ressources de cloud computing à l’échelle du Web les instances ec2 dans joodperfumes

|  |  |  |
| --- | --- | --- |
| **Instances** | **Role** | **Réseau** |
| Odoo | Odoo est le système ERP utilise avec notre boutique | adm-jood-useast1-vpc |
| RabbitMq | RabbitMQ est le courtier de messages utilise par Jood. | qa-jood-useast1-vpc |
| VPN | Créer des connexions sécurisées avec les réseaux adm et qa (staging) basées sur le cryptage du protocole OpenVPN | adm-jood-useast1-vpc |
| Bastion | Instance qui fournit un point d’entrée et/ou de sortie unique vers tout les instances de jood infra par ssh | qa-jood-useast1-vpc |

Table 4.2: Les Instances EC2 de l’infrastructure Jood

* + - * **Amazon ECS** : Amazon Elastic Container Service est un service de gestion de conteneurs hautement évolutif et rapide, qui facilite l’exécution, l’arrêt et la gestion des conteneurs Docker sur un cluster d’instances Amazon EC2. Joodperfumes utilse ECS Fargate qui est un moteur de calcul conçu pour Amazon ECS qui nous permet d’exécuter les conteneurs sans avoir à gérer des serveurs ou clusters.

|  |  |  |
| --- | --- | --- |
| **Conteneur** | **Role** | **Réseau** |
| Jood-Core | API de joodperfumes | qa-jood-useast1-vpc |
| Celery | Celery est une file d’attente de tâches / travaux basée sur la transmission de messages distribuée exécutées en arrière-plan | qa-jood-useast1-vpc |
| Flower | Flower est un outil Web pour surveiller et administrer les Clustrer de Celery. | qa-jood-useast1-vpc |

Table 4.3: Les Conteneurs Docker de l’infrastructure Jood

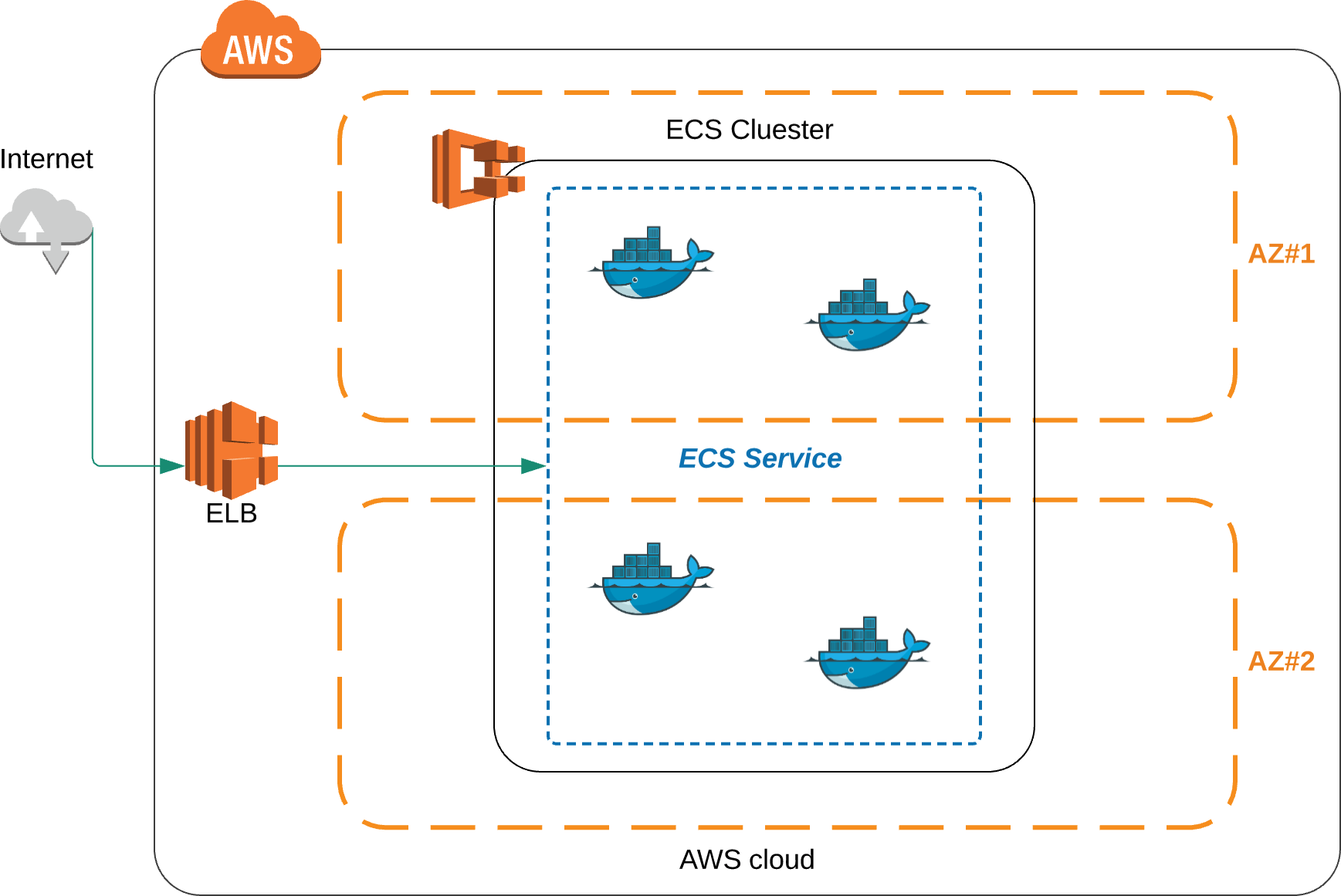


Figure 4.14: la gestion de conteneurs avec AWS Fargate

* + - * **Elastic Load Balancing** : (ELB) répartit automatiquement le trafic entrant sur nos intances Ec2 (rabbitmq) et les conteneurs (flower,jood-core, Celery) dans la zone de staging.

Elb sert aussi a détecter nos cibles non saines et cesser d’y acheminer du trafic et répartir la charge sur les autres cibles saines.

|  |  |
| --- | --- |
| **ELB** | **Type** |
| adm-jood-useast1-bastion-elb | privee |
| qa-jood-useast1-core-api-elb | public |
| qa-jood-useast1-flower-elb | privee |
| qa-jood-useast1-rabbitmq-elb | privee |

Table 4.4: Jood-Elb

* + - * **Amazon ECR** : Amazon Elastic Container Registry est un registre de conteneurs Docker en- tièrement géré par AWS , il permet de stocker, gérer et déployer facilement nos images Docker (conteneurs) a partir de github repository. c’est une version publiable du code source sur aws .

#### Jood-Services de stockage et de livraison de contenu

* + - * **Amazon Relational Database Service** : Le service de base de données Amazon Relational (Amazon RDS) est un service Web qui facilite la configuration, l’utilisation et la mise à l’échelle d’une base de données relationnelle dans le cloud.

Jood infra contient 2 instances RDS l’une pour gerer la base de données de plateforme et l’autre pour la base de données de l’ERP Odoo

|  |  |
| --- | --- |
| **Base de donnees** | **Type** |
| qa-jood-useast1-core | PostgreSQL |
| qa-jood-useast1-odoo | PostgreSQL |

Table 4.5: Les base de donnees RDS du Jood

* + - * **Amazon S3** : Amazon Simple Storage Service est un service de stockage dans le cloud sur internet. Vous pouvez utiliser Amazon S3 pour stocker et récupérer toute quantité de données à tout moment, où que vous soyez sur le Web. le service s3 utilsé dans jood infra pour :
        + Charger des objets (photos, vidéos de produits) ;
        + Stocker des fichiers statiques (html, css, js) ;
        + Le dernier état (snapshot) de l’infrastructure ; (instances, vpc, sécurité ...)
        + Charger les (logs,settings,cache) ...
      * **Amazon CloudFront** : Amazon CloudFront accélère la distribution du contenu Web statique et dynamique, tel que les fichiers .html, .css, .php, images et fichiers multimédias.

Lorsque les utilisateurs demandent le contenu du notre boutique , CloudFront le diffuse via un réseau mondial d’emplacements périphériques offrant une faible latence et des performances élevées.

#### Jood-Services de sécurité et d’identité

* + - * **AWS Identity and Access Management** : AWS (IAM) est un service Web permettant de contrôler en toute sécurité l’accès aux services AWS. Avec IAM, nous pouvons gérer de manière centralisée les utilisateurs, les informations d’identification de sécurité telles que les clés d’accès et les autorisations qui contrôlent les ressources AWS auxquelles les utilisateurs et les applications peuvent accéder.
      * **AWS Security Group** : Aws security group agit en tant que pare-feu virtuel pour que une instance contrôle le trafic entrant et sortant. Lorsque vous lancez une instance dans un VPC, nous pouvons affecter jusqu’à cinq groupes de sécurité à l’instance. Les groupes de sécurité agissent au niveau de l’instance et non du sous-réseau.

la figure4.15[.](#_bookmark102) présente la sg (security groupe ) du service jood-core qui autorise tout le trafic provient du load balancer.

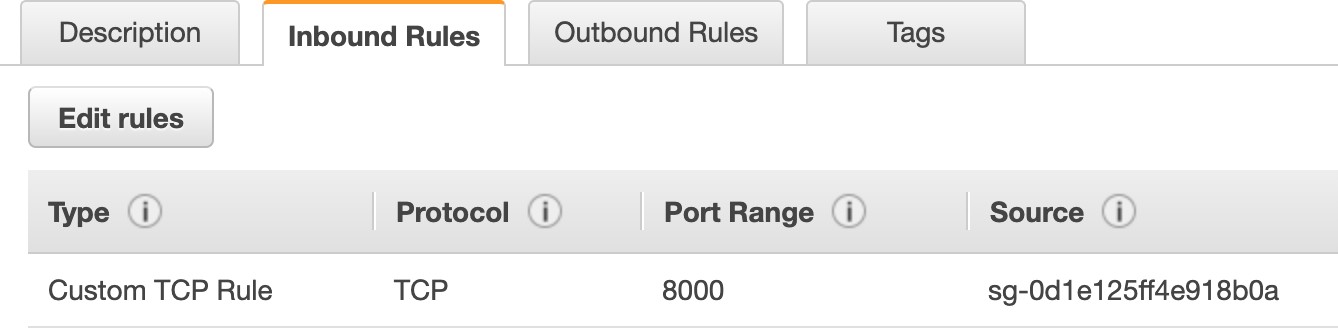


Figure 4.15: Security groupe du service Jood-core

la figure4.16[.](#_bookmark103) presente la sg du load blancer qui autorise tout le trafic provient d’Internet via le port 80 (http) et 443 (https).

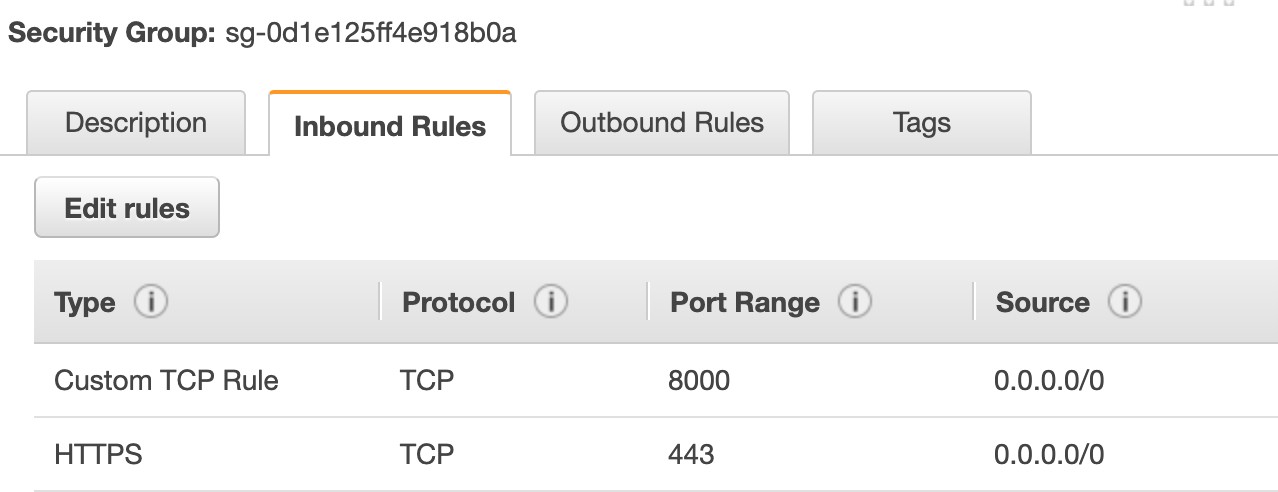


Figure 4.16: Security groupe du Load Balancer

#### Jood-Services de Management

* + - * **AWS Auto Scaling :** Avec AWS Auto Scaling, nous pouvons configurer rapidement une mise à l’échelle dynamique et prédictive pour nos ressources AWS évolutives et pour mettre à l’échelle les instances EC2 et les conteneur docker.
      * **Amazon CloudWatch :** Amazon CloudWatch fournit une solution de surveillance fiable, évolutive et flexible que nous avons utiliser pour collecter et suivre des métriques, qui sont des variables que nous pouvons mesurer pour nos ressources et nos applications.

#### Jood-Outils de développement

* + - * **AWS CodeBuild :** AWS CodeBuild est un service qui compile le code source, exécute des tests unitaires et produit des artefacts prêts à être déployés.
      * **AWS CodePipeline :** AWS CodePipeline est un service de distribution continue qui nous permet de modéliser, de visualiser et d’automatiser les étapes nécessaires à la publication de notre plateforme.
      * **AWS CodeDeploy :** AWS CodeDeploy est un service de déploiement qui nous permet d’au- tomatiser le déploiement sur les instances et de mettre à jour les repositories en fonction des besoins.

Le diagramme4.17[.](#_bookmark106) montre un exemple de processus de publication utilisant CodePipeline.

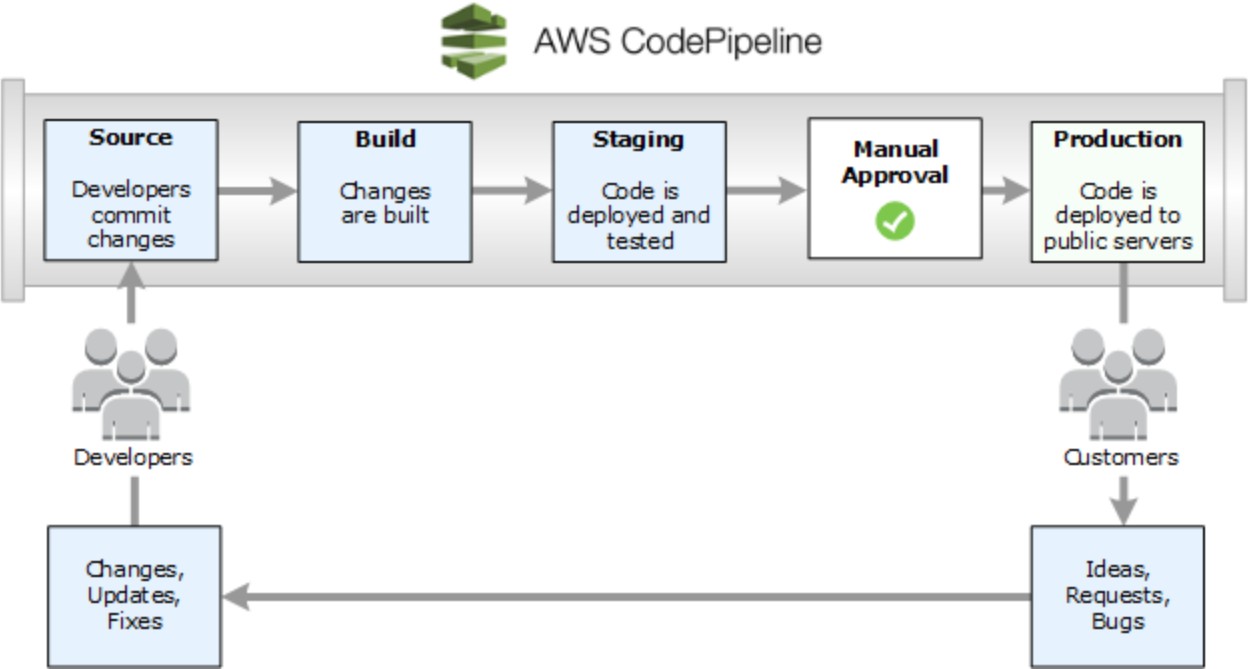


Figure 4.17: Processus de publication avec CodePipelin

**Cycle de déploiement du Jood** Les cycles de déploiement dans Jood-infra sont structurés comme suit :

—Chaque validation des modifications apportées avec commit dans GitHub déclenche un Trigger de terraform par AWS CodePipeline Webhooks ,ce dernière faire appel au CodeBuild pour la compilation du code source et l’exécution des tests unitaire .

—Chaque fusion à la branche master déclenchera un flux de travail avec 2 plans de travail, s’appli- quera à l’environement d’administration , S’appliquera aussi a la production .

La figure4.18représente le Cycle de déploiement du Jood :

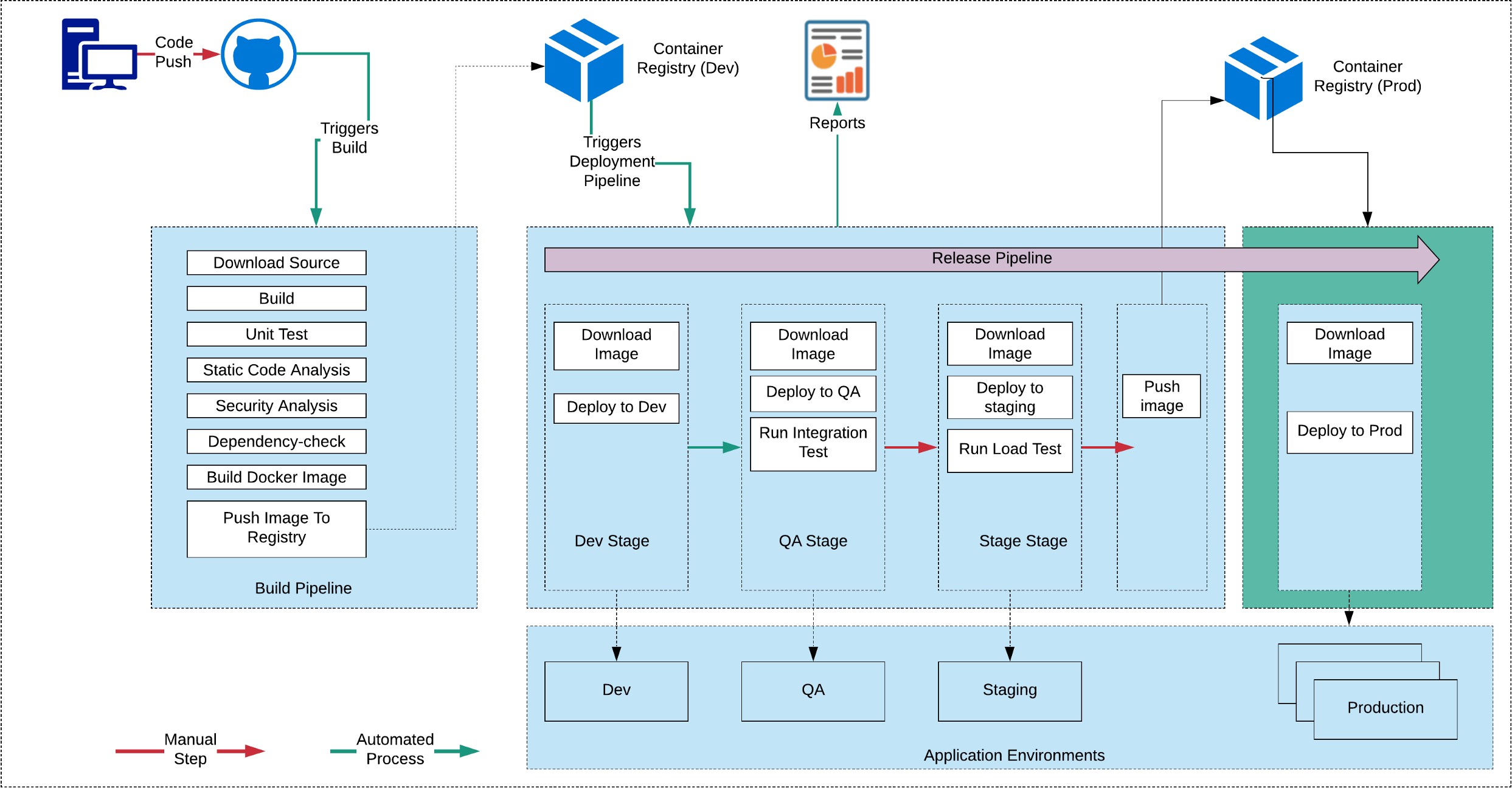


Figure 4.18: Intégration-Déploiement Continue de Jood

#### La sécurisation de l’infrastructiure de Jood

**Problématique** Toute personne sur Internet semble recommander d’exécuter vos instances EC2 dans une boîte aux lettres fermée, d’installer une instance NAT puis de configurer une DMZ avec au moins un serveur bastion. Brisez la vie privée et capturez les sessions des utilisateurs avec des scripts faits maison !

**Solution proposé** Il est pratique courante d’utiliser les boîtes de saut SSH et le groupe de sécurité pour restreindre l’accès SSH aux instances dans AWS. Cette méthode fonctionne très bien,

parmi les choix sur aws est le bastion hôte (instance)

**Un hôte bastion** est un serveur dont le but est de fournir un accès à un réseau privé à partir d’un réseau externe, tel qu’Internet. En raison de son exposition à une attaque potentielle, un hôte de bastion doit minimiser le risque de pénétration. Par exemple, vous pouvez utiliser un hôte bastion pour atténuer le risque d’autoriser les connexions SSH d’un réseau externe à des instances Linux lancées dans vos réseaux privés depuis votre vpc.

**L’architecture de la solution** Dans cette section, je présente l’architecture de cette solution sur notre projet et explique comment nous avons configuré l’hôte bastion pour enregistrer les sessions SSH. Comme nous l’avons déjà expliqué dans ce chapitre, Amazon VPC vous permet de lancer des ressources AWS sur un réseau privé virtuel que vous avez défini. L’hôte du bastion fonctionne sur une instance Amazon EC2 qui est généralement située dans un sous-réseau public de votre commande par correspondance Amazon. Les instances Linux sont situées dans un sous-réseau qui n’est pas accessible

au public, et elles sont configurées avec un groupe de sécurité qui permet l’accès SSH depuis le groupe de sécurité attaché à l’instance EC2 sous-jacente qui exécute l’hôte bastion. Les utilisateurs de l’hôte Bastion se connectent à l’hôte Bastion pour se connecter aux instances Linux, comme le montre le schéma suivant4.19

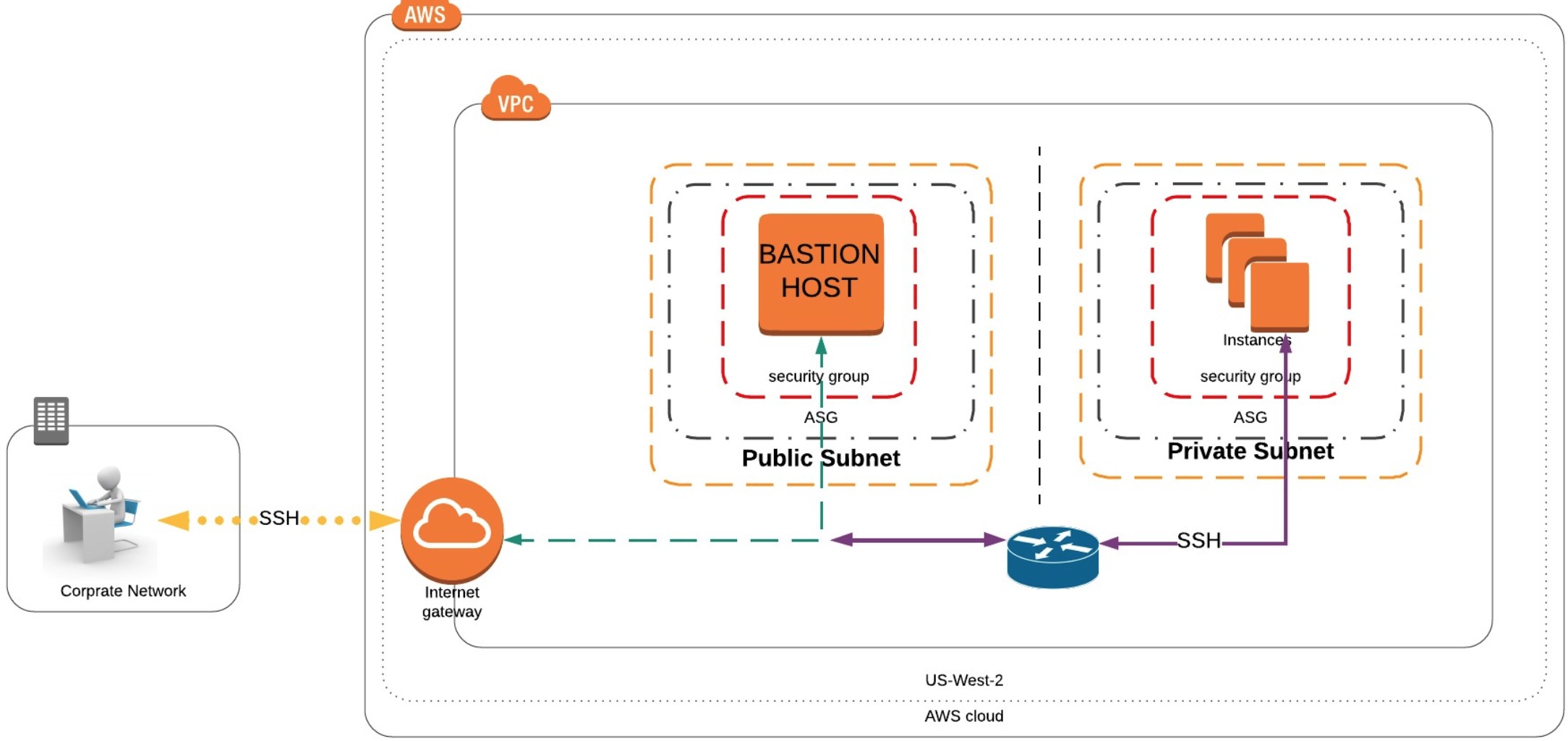


Figure 4.19: Sessions SSH établies par l’intermédiaire d’un hôte Bastion

Bastion host dans Jood infra est une instance qui se trouve dans l’admin vpc. Une fois que la connectivité à distance a été établie avec l’hôte bastion, il agit alors comme un serveur ’jump’, vous permettant d’utiliser SSH pour vous connecter à d’autres instances (dans des sous-réseaux privés/pu- blics) plus profondes dans notre adm et qa VPC qui sont liées avec une connexion de peering VPC entre les deux. .le bastion sert essentiellement de pont vers d’autres instances du réseau. Toutes les instances ont le port 22 restreint pour être ouvert uniquement à l’instance du bastion.

Afin d’améliorer cette solution, il est parfois très utile de pouvoir accéder aux instances directe- ment et de manière sécurisée. Pour ce faire, nous avons utilisé OpenVPN combiné avec quelques astuces AWS pour créer un serveur bastion VPN résilient. et pour mettre tous les services (conteneurs docker, instances...) derrière le vpn

**Qu’est-ce qu’un VPN ?** Pensez à un réseau privé virtuel comme une couche sous votre connexion réseau. En connectant votre appareil au VPN, les données sont envoyées et reçues en privé via une adresse IP différente. La connexion à un VPN offre une multitude d’avantages, tels que la fonctionnalité, la sécurité (cryptage des données). En termes très simples, un hôte VPN est le seul hôte ou ordinateur autorisé à accéder au réseau public. Il s’agit généralement d’une puissante boîte de serveur avec une sécurité réseau élevée parce que c’est le seul hôte qui est autorisé l’accès public (donc il dirige notre instance bastion vers un réseau privé) Les administrateurs système peuvent utiliser cette machine pour se connecter au réseau jood dans le cloud

Cela signifie que vous ne pouvons pas ssh aux instances du Jood sans passer par l’hôte bastion et vous ne pouvons pas y accéder si nous sommes pas connecté à l’hote VPN. comme le montre le schéma suivant : figure4.20

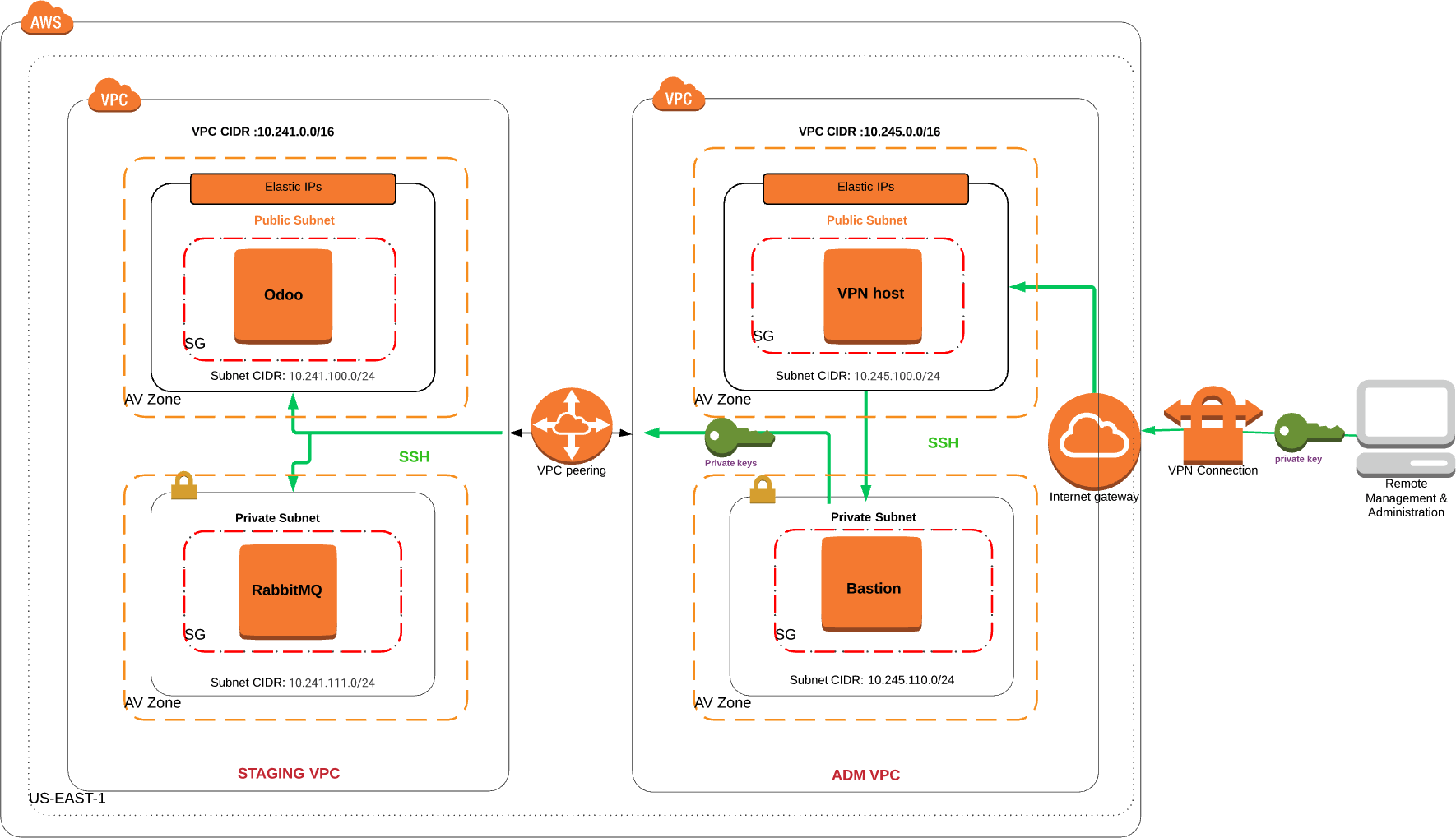
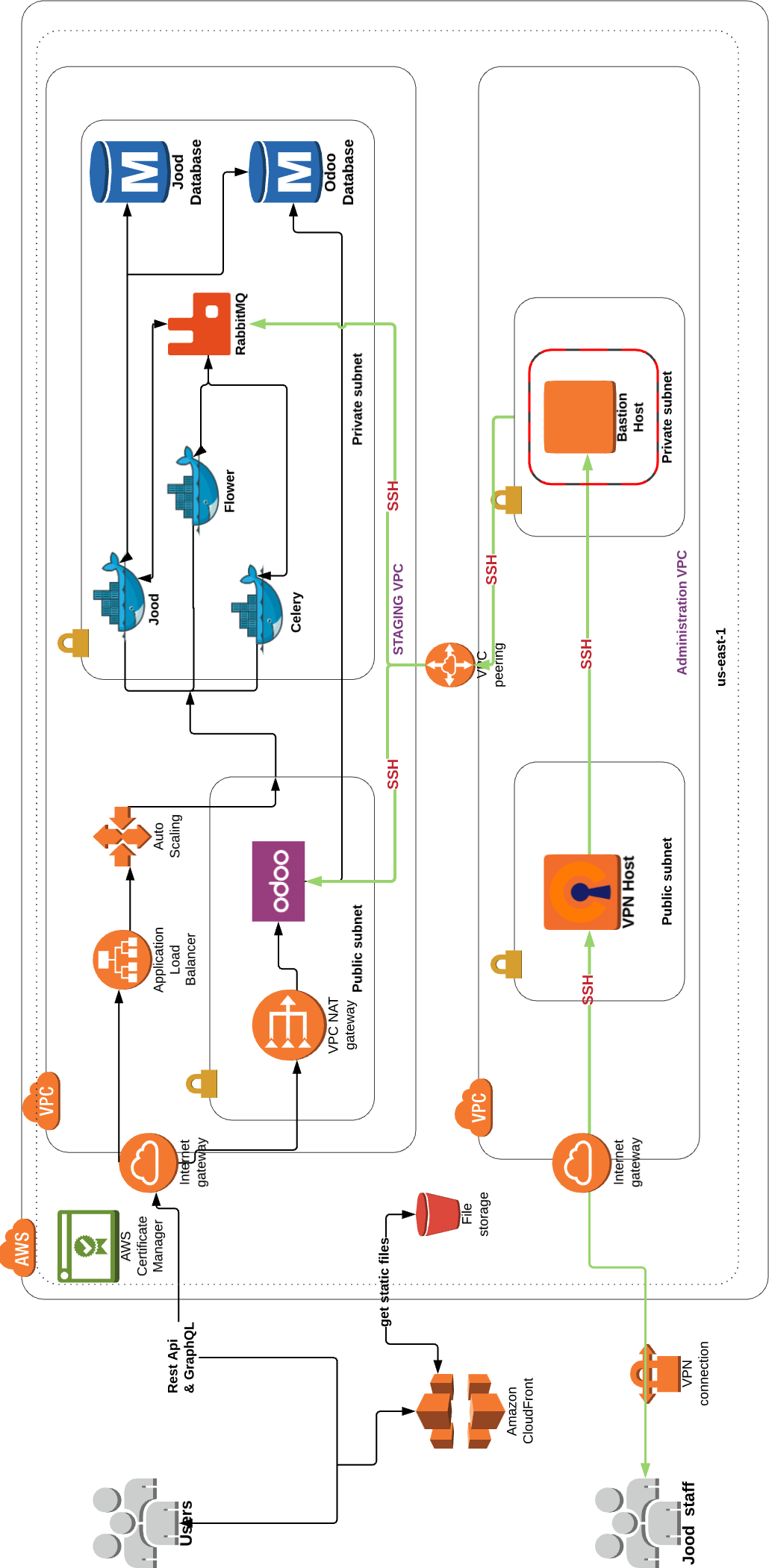


Figure 4.20: La configuration des hôtes VPN et Bastion dans jood

#### L’architecture global

La figure4.[21](#_bookmark112)représente l’architecture cloud du Jood sur AWS :

Figure 4.21: L’architecture cloud du Jood sur AWS



### Conclusion

Durant ce chapitre, nous avons traité la conception et l’implémentation de l’environment Cloud pour les différents services du Joodperfumes

# 5

## Jood - développement backend

***les référentiels du Joodperfumes sont séparés en 3 catégories : référentiels de micro- modules backend, référentiels d’opérations et référentiels d’interface utilisateur. Dans ce chapitre, nous expliquerons l’architecture backend du projet Joodperfumes. D’abord nous commencerons par détailler la structure des repertoire sous Jood-core ,l’utilisation des différents fichiers ensuite nous présenterons quelques outils utilisés pour l’administration de l’application JOOD-ODOO et leur objectif ansi que quelques captures de notre projet.***

### Architecture backend du Jood

JOOD a différentes applications, chacune ayant une responsabilité unique mais formant ensemble une seule et même plateforme solide et cohérente.

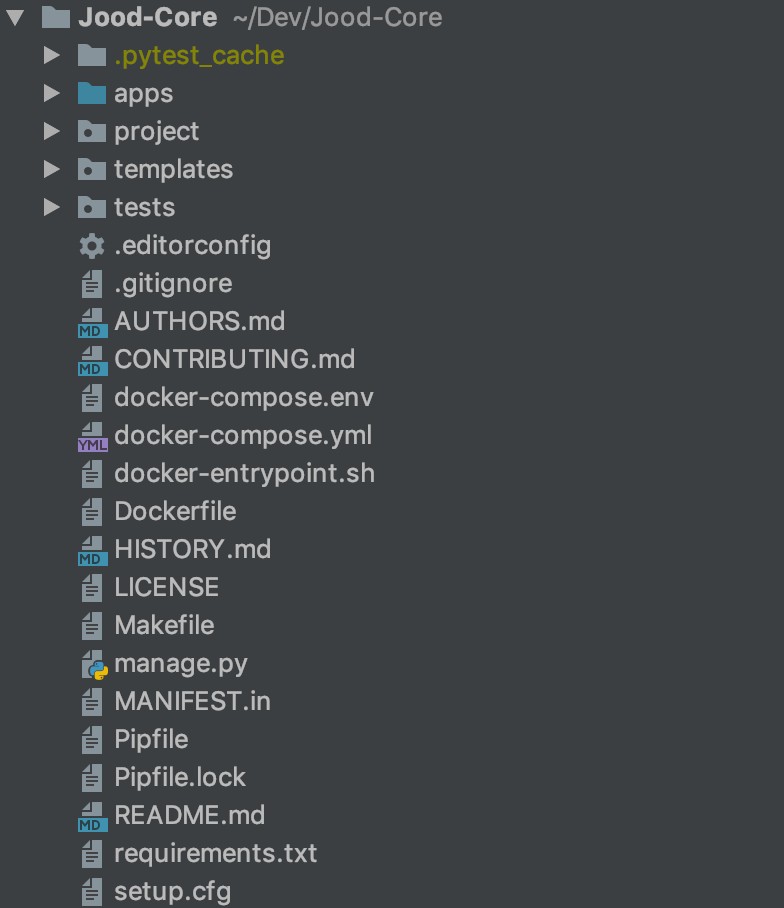


Figure 5.1: Arhitecture du Jood-Core

#### Répertoire Pytest\_cache

pytest-cache fait deux choses :

* + - * un mécanisme par lequel d’autres plugins peuvent obtenir / définir des valeurs via config.cache.get|set. Ceci est utilisé par pytest-pep8 et pytest-flakes par exemple pour stocker l’heure de la dernière vérification afin d’éviter de revérifier les fichiers tout le temps.
      * sauvegarder les échecs de test afin que vous puissiez exécuter –lf pour n’exécuter que les derniers échecs et –ff pour exécuter les derniers échecs en premier, puis le reste des tests

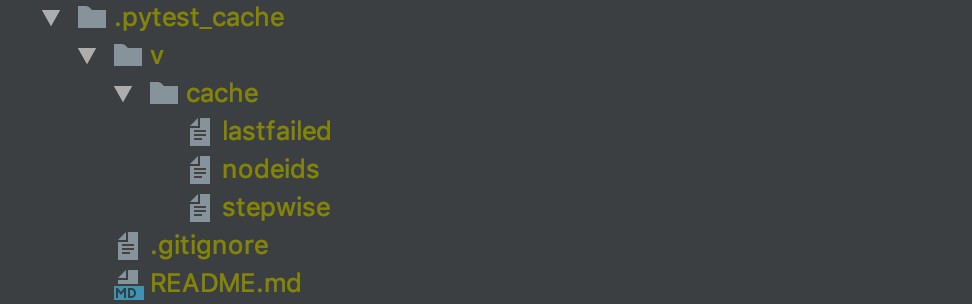


Figure 5.2: Le Répertoire du Pytest\_cahe

#### Répertoire Project

* **Celery.py** contient la configuration du service CELERY TASK
* **S3\_storage.py** fichier de configuration pour le système de stockage S3
* **Production.py** contient la configuration spécifiée en production (paramètres spécifiques à l’en- vironnement, caches)
* **Local.py** contient la configuration dédié à l’environnement local ( Réglages spécifiques à l’envi- ronnement, caches)
* **Views.py**c’est ficher des contrôleurs de l’application, Il contient la fonction qui nous retourne la page Index.html
* **Urls.py** c’est le fichier des routes , le lien entre les contrôleurs ( views ) et les vues ( templates )
* **Wsgi.py** spécifie Comment déployer avec WSGI (Web Server Gateway Interface)

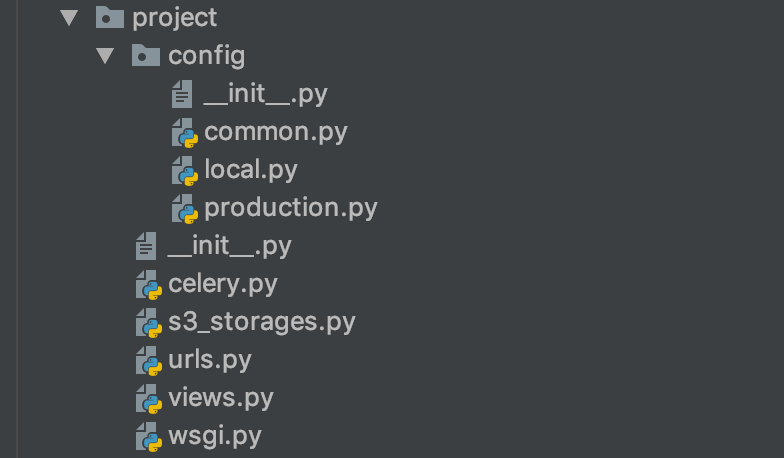


Figure 5.3: le répertoire d’arborescence du projet

#### Répertoire Templates

C’est le répertoire des Vues de backend, il contient seulement index.html page de démarrage de l’application vue que les templates et leurs traitement se présentent dans Jood-Web

#### Répertoire apps

Il contient les répertoires de l’objectif principale du projet, distribués aux modules suivants appelés applications. Le terme application décrit un paquet Python qui fournit un certain ensemble de fonctionnalités. Les applications peuvent être réutilisées dans différents projets

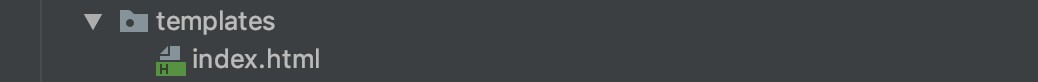


Figure 5.4: Le répertoire de templates

* + - * **Cart** C’est le module qui gère les paniers et ces articles, contient les classes Cart, CartItemet et les services Web graphql
      * **Profile** c’est le module qui contient les classes User, et Site, il a la même forme de Django-profile, il contient les contrôleurs d’enregistrement et d’authentification et les services Web Graphql Généralement, un utilisateur peut s’authentifier avec un email / mot de passe

Pour enregistrer un utilisateur avec un email et un mot de passe, nous faisons un POST pour

/ utilisateurs / en incluant ces informations : Email, mot de passe, prénom, nom, numéro de téléphone

* + - * **Product** C’est le module qui gère le stock et la catégorie de produits, ainsi que les dépôts de ces produits, et fournit aux utilisateurs toutes les informations sur les produits.

c’est le module qui contient les classes Product, Site, Category, Warehouse, GiftWrapping et GiftWrappingSite, ainsi que les services Web graphql.

* + - * **Order** Ce module gère le cycle de vie d’une commande, de la création à la facturation.

le module qui contient les classes Order, OrderItem et Shipping Address et les services Web Graphql, ainsi qu’un service Web Rest pour initialiser le paiement d’une commande.

Il gère également la facturation après le parment d’une commande

* + - * **graphql\_core** c’est le répertoire qui contient la configuration générale de graphql ainsi que les fonctions utilitaires de graphql (fonction de décodage et d’encodage, ...) ainsi que les classes mères de mutations ...
      * **graphql\_utils** contient le schéma de toutes les requêtes graphql dans l’application

Il s’agit d’un contrat conclu entre e front-end et le back-end. Par conséquent, son maintien au centre permet aux deux parties du développement d’évoluer sans sortir du cahier des charges. Cela facilite également la parallélisation du travail, car le client peut dès le départ avoir une connaissance complète de l’API, à l’aide d’un simple service de simulation, qui peut ensuite être facilement remplacé par ce dernier. serveur.

* + - * **Notify** c’est le module qui gère les notifications envoyé aux utilisateurs après un paiement d’un ordre ou d’une inscription au site et aussi les promotions

Le service devrait :

Autoriser l’utilisateur à définir ses préférences de notification. Autoriser l’utilisateur à définir sa méthode de notification préférée

Fournir une interface configurable aux modèles de courrier électronique et aux fournisseurs.

Gérez le fil de nouvelles des utilisateurs en fonction des canaux de notification et des canaux pub / subs

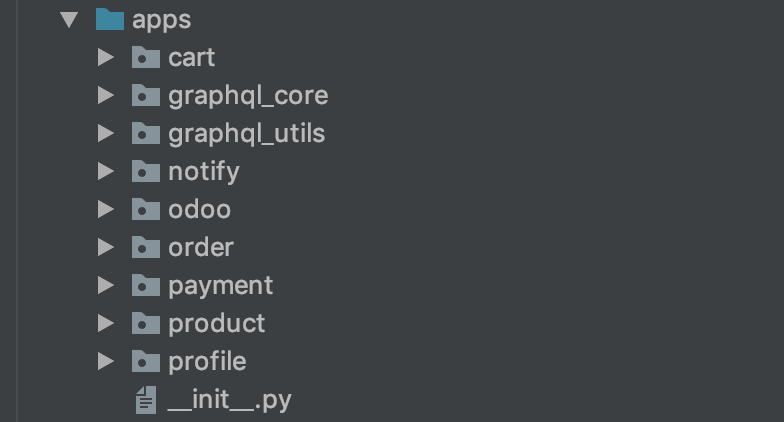


Figure 5.5: Le répertoire des modules de projet

#### Répertoire Tests

C’est le répertoire qui contient les tests de chaque module dans le répertoire apps.

Ceci est le répertoire qui contient les tests pour chaque module du répertoire apps. Dans notre projet, nous testons par module et répartis en 4 types de tests :

* **Test Queries** sert a tester les requêtes de recherche dans la base de données en utilisant graphql.
* **Test API** ce sont les tests API externes utilisés dans le module comme knet paiment, odoo item **Test de Mutation** ce sont les tests des demandes de création et de modification, et la suppression dans la base de données en utilisant Graphql

Pour démarrer les tests, tapez simplement la commande "make test" pour lancer tous les tests.

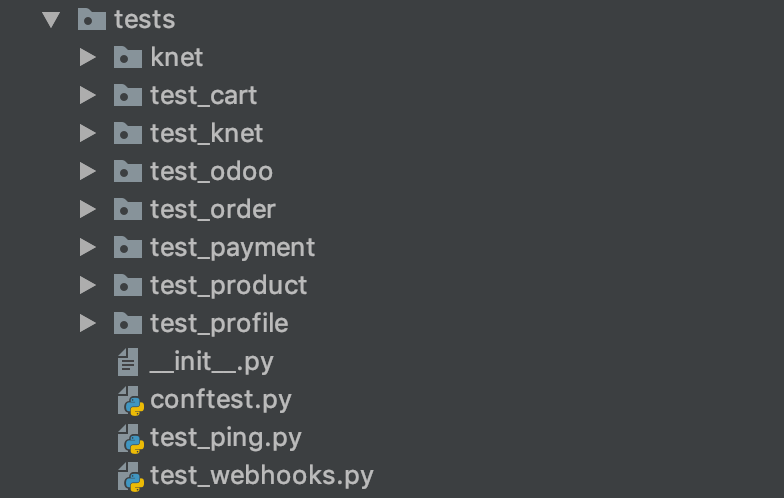


Figure 5.6: Le répertoire des tests

#### Les fichiers Docker

Il y a 4 fichiers docker et docker dans notre application :

* **Dockerfile :** c’est le fichier qui définit les dépendances nécessaires à la construction de l’image JOOD-CORE
* **Docker-compose.env :** contient les variables d’environnement telles que DATABASE\_URL
* **Docker-compose.yaml :** contient les services docker utilisé par l’image JOOD-CORE
* **Docker-entrypoint.sh :** contient les commandes nécessaires à exécuter lors du démarrage de l’application
* **Makefile :** contient des commandes spécifiques pour notre Conteneur Docker , telles que make tests ...

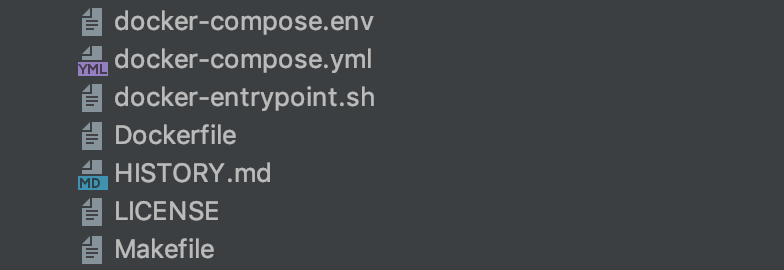


Figure 5.7: Le fichiers docker de projet

#### Autre Fichiers

* **manage.py :**fichier principale de django c’est le point d’entrée de projet
* **Pipfile :** contient les versions des package ,docker utilise ce fichier pour construire l’image et installer toutes les dépendances de l’image
* **Pipfile.lock :**contient les information de chaque dépendance du projet comme le lien de télé- chargement), requirements.txt (le fichier qui contient les exigences du projet

### Interaction avec l’ERP Odoo

Pour une meilleure gestion des stocks, des ventes, des ressources humaines et des clients, la société a décidé d’utiliser l’ERP Odoo et de la relier avec la plate-forme JOOD afin de synchroniser les données des commandes, les produits et les clients après chaque opération de création, de modifier ou de supprimer la base de données du site à l’aide du XML. Protocole-RPC pour connecter Jood-Core à Jood-Odoo et CELERY afin que les tâches XML-RPC soient synchronisées avec les opérations JOOD- CORE et doivent se trouver à l’arrière-plan du site dans l’objective de ne pas ralentir le fonctionnement, c’est-à-dire que les erreurs de ces tâches n’influencent pas sur le fonctionnement normal du site.

les figures au dessous montrent quelques fonctions implémentées sur odoo

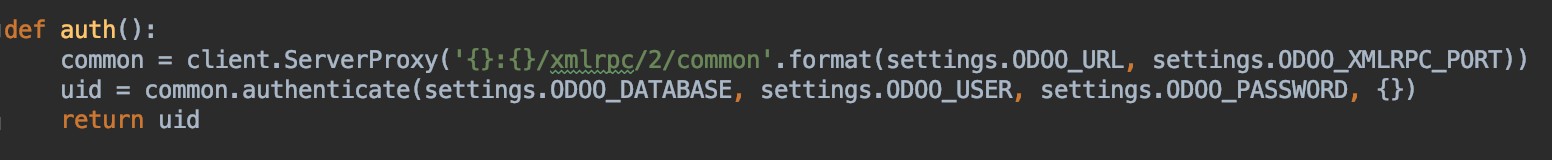


Figure 5.8: Connexion Jood-Core au serveur Jood-Odoo avec XRPC



Figure 5.9: Obtenir les modèles de Odoo



Figure 5.10: Creation d’un ulisateur sur Odoo

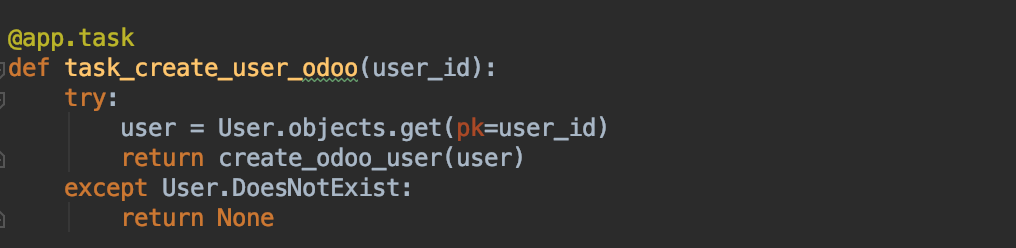


Figure 5.11: création d’un utilisateur sous Odoo avec celery

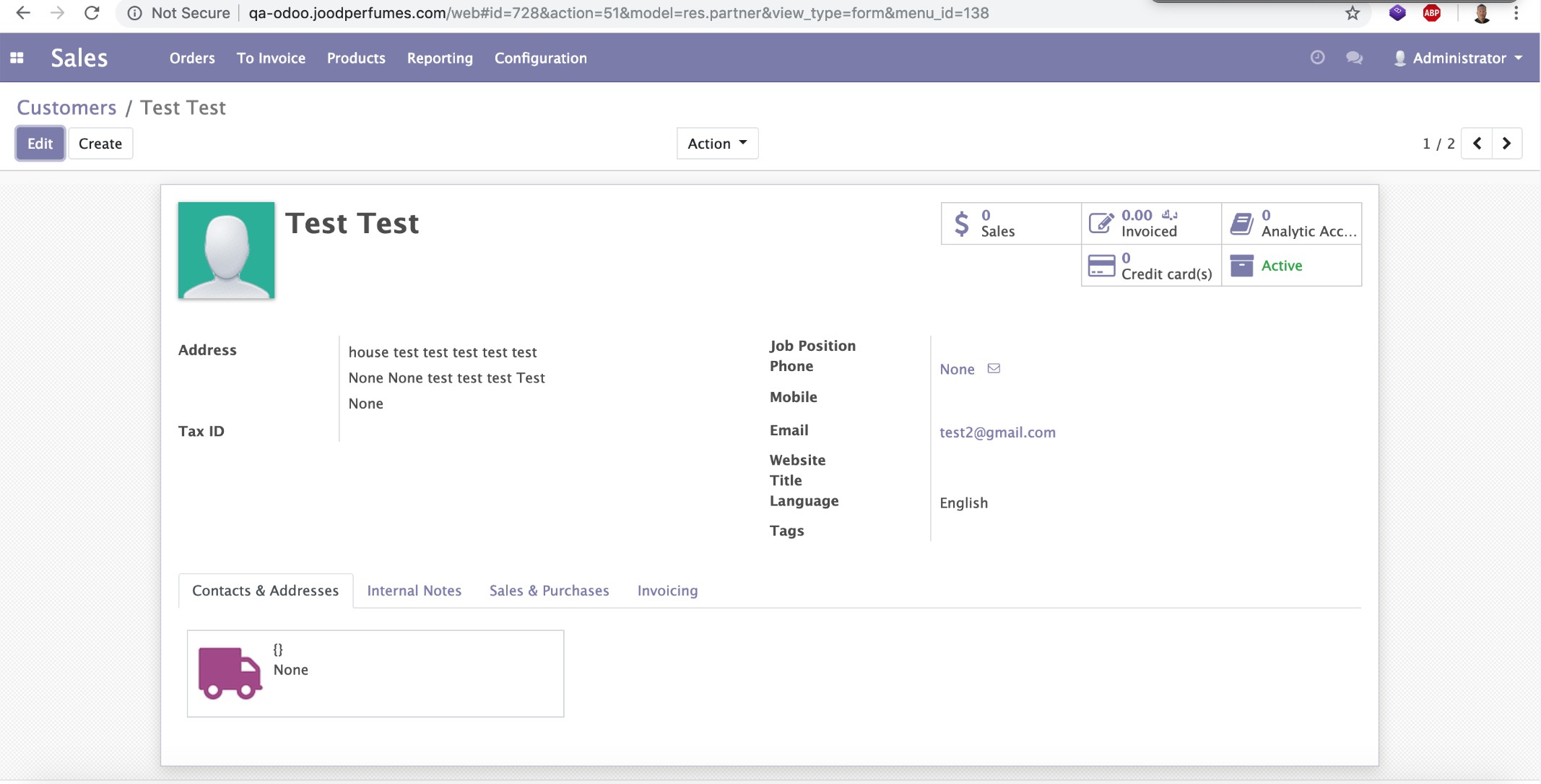


Figure 5.12: La création d’un ordre pour un utilisateur dans Odoo

### L’API de Jood

**GRAPHQL API :** Pour servir API graphql dans notre projet, nous avons suivi les étapes suivantes pour chaque classe : On prend la class Product comme exemple

**Création de types et de schémas** Dans GraphQL, un Type est un objet pouvant contenir plusieurs champs. Chaque champ est calculé via des résolveurs, qui renvoie une valeur. Une collection de types s’appelle un schéma. Chaque schéma a un type spécial appelé requête pour obtenir des données du serveur et une mutation pour envoyer des données au serveur.



Figure 5.13: Exemple d’un ProductType avec graphene-django

Dans l’extrait de code ci-dessus, ProductType a été créé à l’aide de DjangoObjectType - un type personnalisé disponible dans Graphene Django.

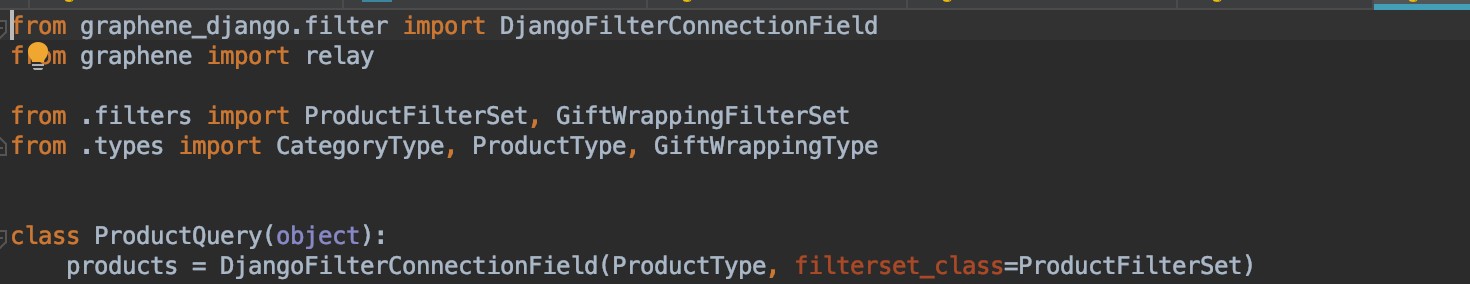


Figure 5.14: Exemple de ProductQuery avec graphene-django

Dans l’extrait de code ci-dessus, la requête de products spécial a été créée avec un résolveur pour les liens de champ, qui renvoie tous les liens

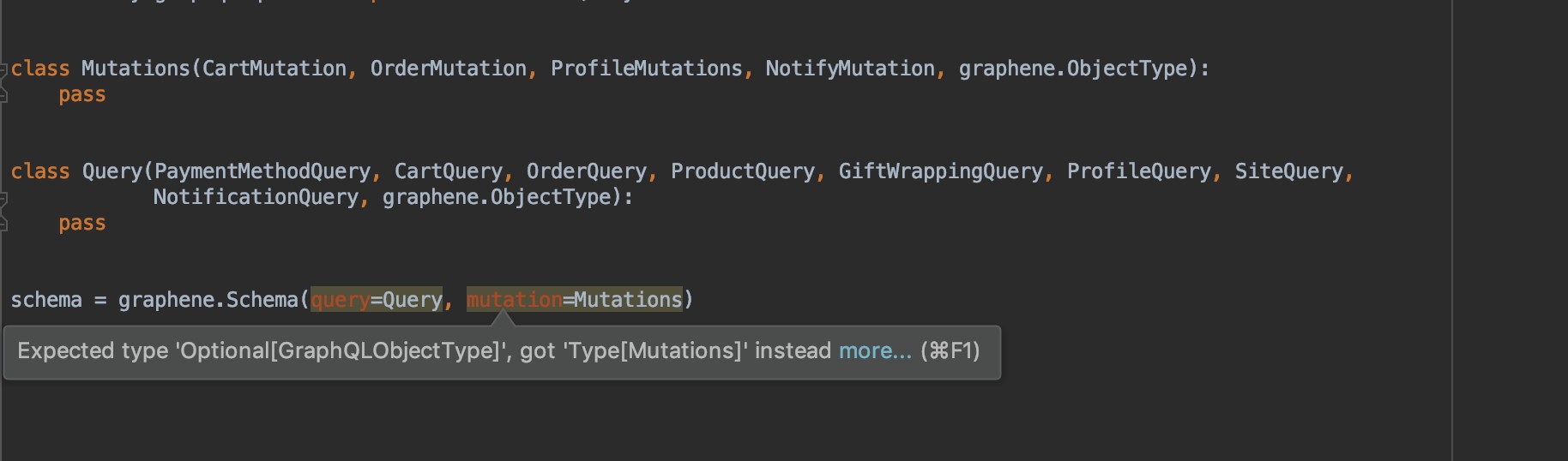


Figure 5.15: Déclaration des requêtes et des mutations dans le schéma du Joodperfumes

**GraphiQL** est un IDE graphique intégré au navigateur. En d’autres termes, un terrain de jeu. Notez que vous devez désactiver la protection Django CSRF.



Figure 5.16: L’ajout de graphiql au urlpattern de projet

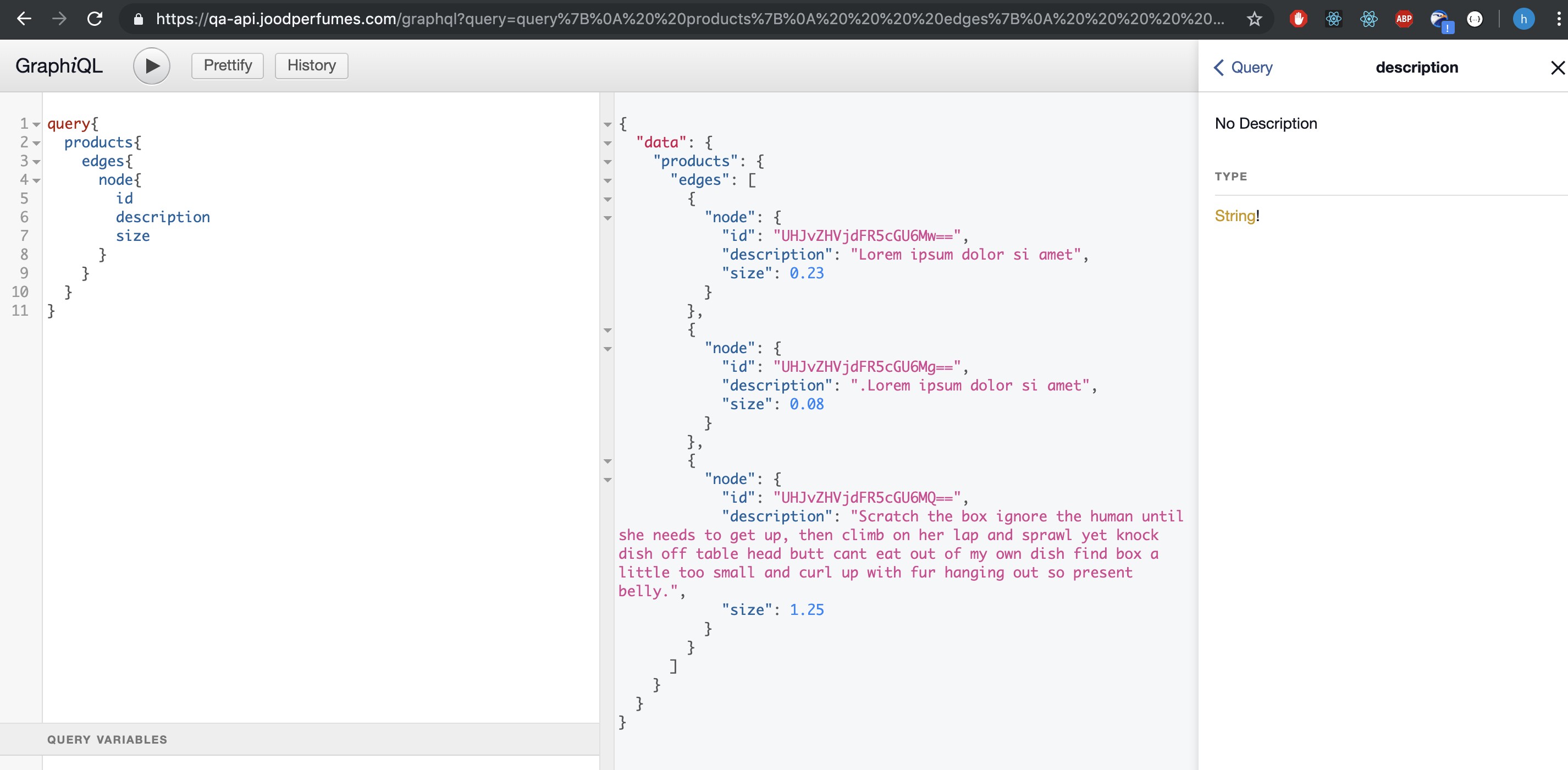


Figure 5.17: La Vue de Graphiql api Jood

Sur la droite se trouve l’explorateur de documentation, des informations sur notre schéma créées automatiquement par introspection. Sur la gauche, pour créer votre requête, tapez ce qui suit :



Figure 5.18: La requête products dans graphiql Vous devriez voir un lien de réponse ceci :

Figure 5.19: Le résultat de la requête products dans graphiql

### Conclusion

Dans ce chapitre, on a pu présenter le fonctionnement de la côté backend, qui a servir une API graphql et aussi api Rest , consommable par le Front end de l’application, de tout en mettant en place les différents outils techniques et logicielles qui nous a permis de concevoir les différents modules de projet.

# 6

## Jood - développement Front-end

### Introduction

***Nous présenterons dans cette section un aperçu général sur le fonctionnement de la plateforme en utilisant quelques captures d’écran. Toutefois, il faut mentionner que la plateforme est encore en version Beta et que plusieurs de ses aspects ne sont pas encore implementés dans l’UI, soit parce qu’il sont encore en phase de tests dans le Back-end ou bien qu’ils n’ont pas une priorité elevée...***

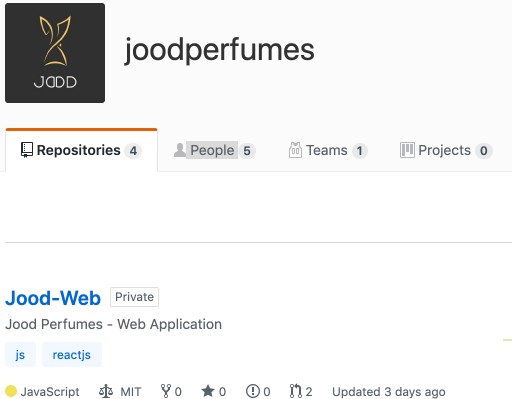


Figure 6.1: Le référentiel du front-end sur github

### Organisation du projet

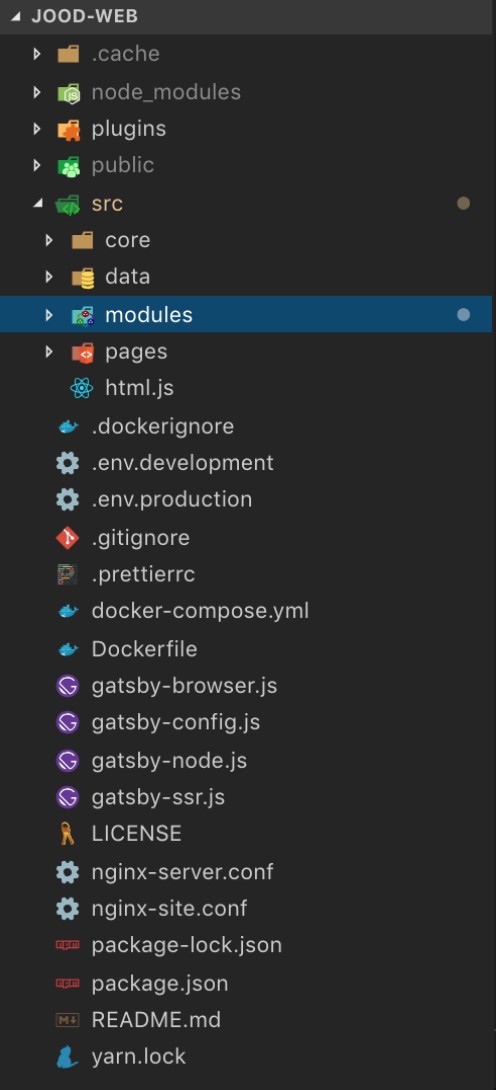


Figure 6.2: La structure du Jood-Web

##### Dossiers

* + - **cache :** Généré automatiquement. Ce dossier est un cache interne créé automatiquement par Gatsby. Les fichiers contenus dans ce dossier ne sont pas destinés à être modifiés. Devrait être ajouté au fichier .gitignore s’il n’a pas déjà été ajouté.
    - **public :**généré automatiquement. La sortie du processus de construction sera exposée dans ce dossier. Devrait être ajouté au fichier .gitignore s’il n’a pas déjà été ajouté.
    - **node\_module, /package.json : :** les packages modules des dépendances des framework javas- cript pour le projet frontend.
    - **plugins :** Ce dossier héberge tous les plugins spécifiques au projet («locaux») qui ne sont pas publiés sous forme de paquet npm. Consultez la documentation du plugin pour plus de détails.
    - **src :** Ce répertoire contiendra tout le code lié à ce que vous verrez sur le frontend de votre site (ce que vous voyez dans le navigateur), comme tête de votre site, ou un modèle de page. « Src » est une convention pour « code source »

**pages :**Les composants sous /src/pages deviennent automatiquement des pages avec des chemins basés sur leur nom de fichier

**data :**Ce répertoire contient les fichiers comportant les données statiques de l’application tel que countries.js et currencies.js.

**core :** Ce répertoire contient les fournisseurs de service pour l’application sous le répertoire

/provider (tel que : apolloClient, AuthManager...etc) ainsi que la gestion du state avec Redux dans la totalité de l’application

**modules :** Ce répertoire contient tous les composants partiels dans l’application, catégorisé par fonctionnalité (tel que : auth : gestion d’authentification, order : gestion des commandes, giftWrapping. .etc) chaque module peut contenir plusieur sous répertoire tel que /hoc, /actions,

/component, /container

**html.js :** Pour la configuration personnalisée de .cache/default\_html.js par défaut

##### Fichiers

* + - * **gatsby-browser.js :** Ce fichier est l’endroit où Gatsby s’attend à trouver un usage quel- conque des API de navigateur Gatsby (le cas échéant). Ceux-ci permettent la personnalisa- tion / extension des paramètres par défaut de Gatsby affectant le navigateur.
      * **gatsby-config.js :** Il s’agit du fichier de configuration principal d’un site Gatsby. C’est là que vous pouvez spécifier des informations sur votre site (métadonnées), telles que le titre et la description du site, les plugins Gatsby que vous souhaitez inclure, etc. Consultez la documentation de configuration pour plus de détails.
      * **gatsby-node.js :**ce fichier est l’emplacement où Gatsby s’attend à trouver un usage quel- conque des API de nœud Gatsby (le cas échéant). Ceux-ci permettent la personnalisation / extension des paramètres par défaut de Gatsby qui affectent des éléments du processus de construction du site.
      * **gatsby-ssr.js :** ce fichier est l’endroit où Gatsby s’attend à trouver un usage quelconque des API de rendu côté serveur de Gatsby (le cas échéant). Celles-ci permettent la per- sonnalisation des paramètres par défaut de Gatsby ayant une incidence sur le rendu côté serveur.

### Workflow de JoodPerfumes

Cette partie représente un scénario d’utilisation du projet par un utilisateur ordinaire dans lequel je vais expliquer plus le fonctionnement interne de chaque partie

#### Interfaces d’utilisateur principale

Après ouverture du projet en local en accédant à **http ://localhost :8000/** ou via **https ://qa- web.joodperfumes.com/**la page d’accueil est par défaut initialisé par les données adapté pour country= kuwait https :../kw/

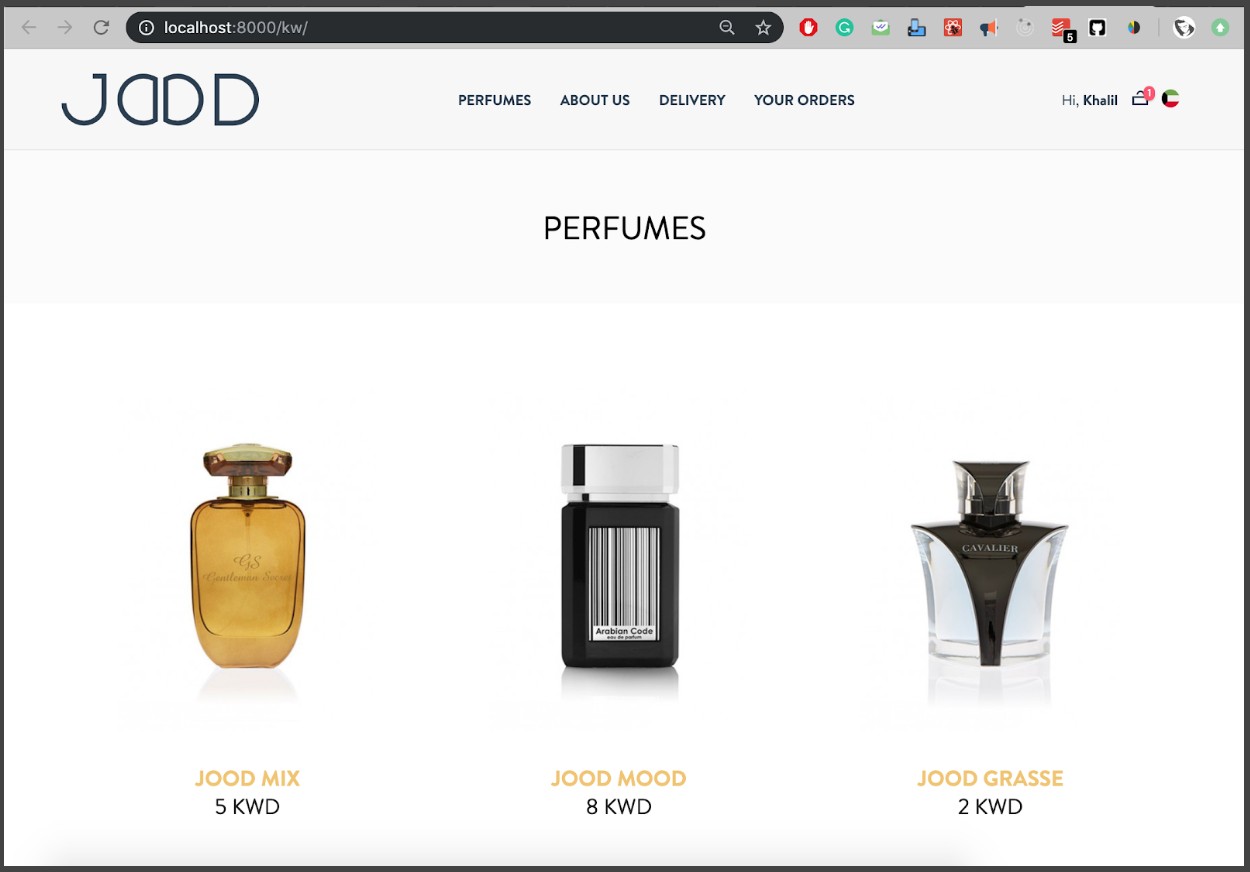
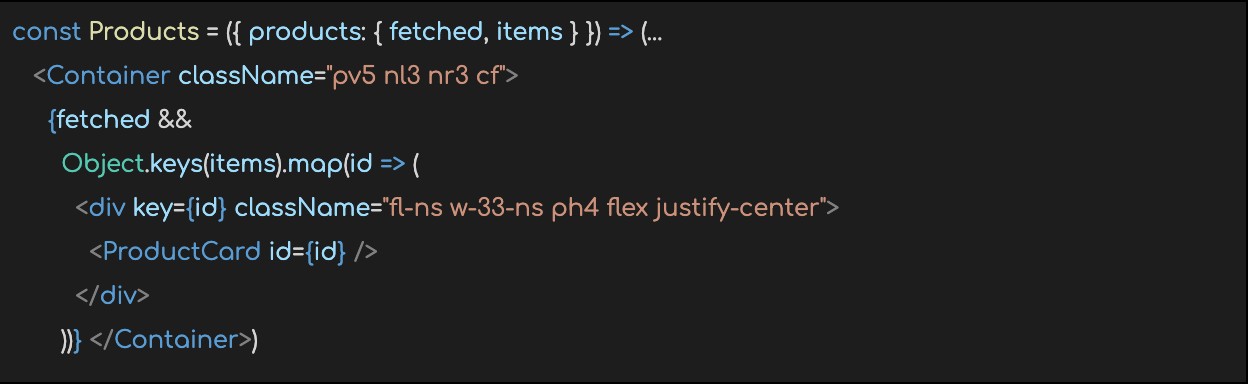


Figure 6.3: L’interface principale du Joodperfumes

La page est composé de plusieurs Components, dont Products est le plus principale : ce compo- sant liste tous les produits venant d’un composant d’ordre supérieur (higher order component, référencé aussi HOC) appelé **withProducts** Qui à son tour est basé sur une requête **GraphQL query**



WithProducts est un composant d’ordre supérieur, localisé sous ‘src/modules/products/hoc/wi- thProducts‘

Sa seule mission est de récupérer les produits existants dans la base de données depuis l’interface GraphQL en utilisant la requête mentionnée précédemment

Ensuite ce composant transmis les produits récupérés au props à l’aide de la fonction mapState- ToProps

De l’autre côté, le composant qui a besoin des produits doit inclure le WithProduct dans son HOC (composant d’ordre supérieur) pour recevoir tous les produits comme props

**Scenario : Ajout des produits au panier** Un utilisateur peut naviguer dans les différents produits offerts par l’application, et ajouter les produits qu’il désire au panier, Pourtant l’ajout d’un produit au panier donne accès à deux scénarios possibles : ‘Ajout normal‘ ou ‘Ajout avec papier d’emballage cadeau‘

Dans le cas d’ajout d’un produit avec l’option ‘Gift Wrapping‘, l’utilisateur est demandé de choisir un papier cadeau via une fenêtre Pop-up

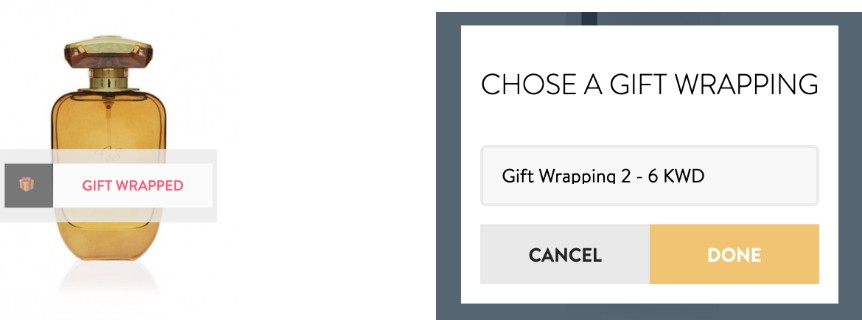


Figure 6.4: Le choix d’un papier cadeau (Gift Wrapping)

Après avoir sélectionné les produits qu’il désire commander, l’utilisateur a la possibilité de vi- sualiser son panier ‘card‘ en cliquant sur l’icône situé dans le header (figure suivante)

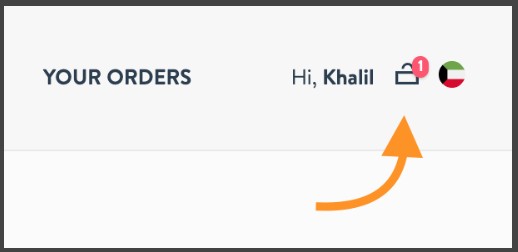
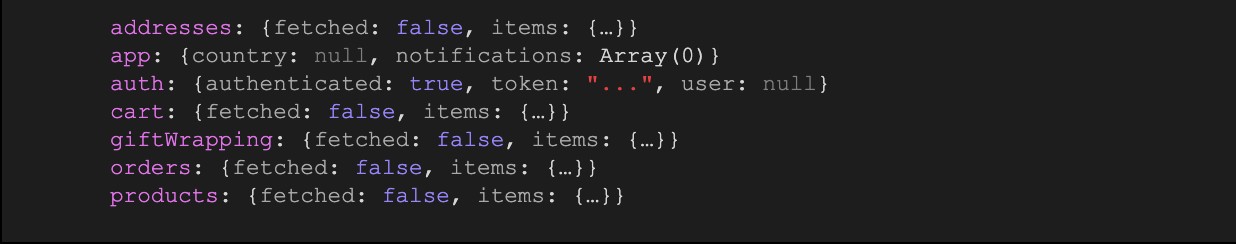
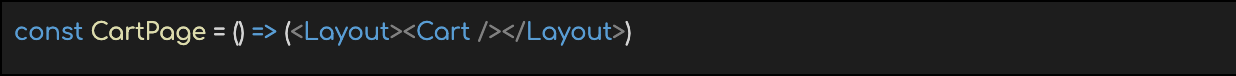


Figure 6.5: L’icon de panier du plateforme Jood

Chaque élément ajouté dans le panier modifie le state global de l’application selon le module



Dans cet exemple Cart contient toutes informations en relation avec le panier d’utilisateur et les produits qui l’intéressent. Toutes ces informations à propos du panier sont regroupés dans un component général appelé cartPage

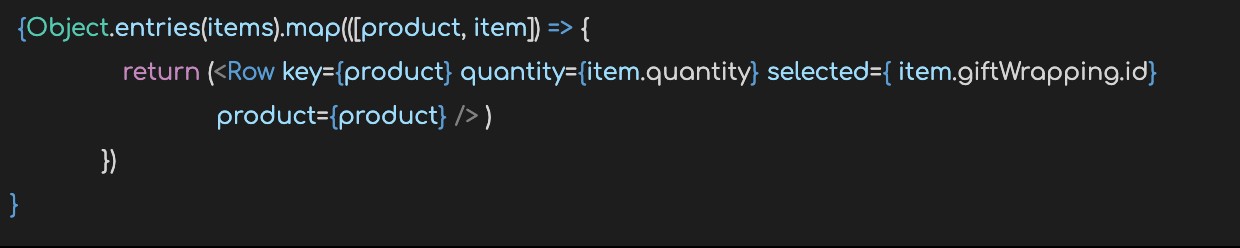


cartPage donne la main à l’utilisateur de faire des modifications à son panier, principalement : Modifier la quantité de chaque produit Ajouter ou supprimer les ‘Gift Wrapping‘ pour chaque produit Supprimer un produit du panier



Figure 6.6: Le contenu d’un panier dans Jood

Chaque élément du panier est visualisé grâce à un composant appelé Row, qui s’occupe de la gestion de toutes fonctionnalités spécifiques à ce produit



Items represente les elements du panier (cart), chaque element est un objet json :

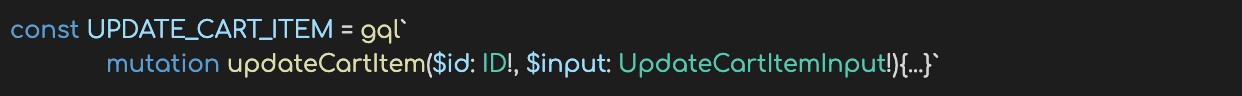


La gestion des éléments du panier, (appelé cardItem) est basé sur 3 composants HOC (Compo- sants d’ordre supérieur) principaux qui interagissent avec l’interface GraphQL :

* + - * **withAddCartItem :** ajout d’un élément au panier en exécutant la mutation addCartItem



* + - * **withUpdateCartItem :** modification d’un élément au panier avec la mutation update- CartItem



* + - * **withDeleteCartItem :** suppression d’un élément au panier en exécutant la mutation deleteCartItem



**Scenario : Procéder au paiement** Afin de pouvoir procéder au paiement, l’utilisateur doit accéder à la page de paiement via Go to Checkout (voir la figure précédente), ce qui entraîne le chargement des nouveaux composants (figure suivante).

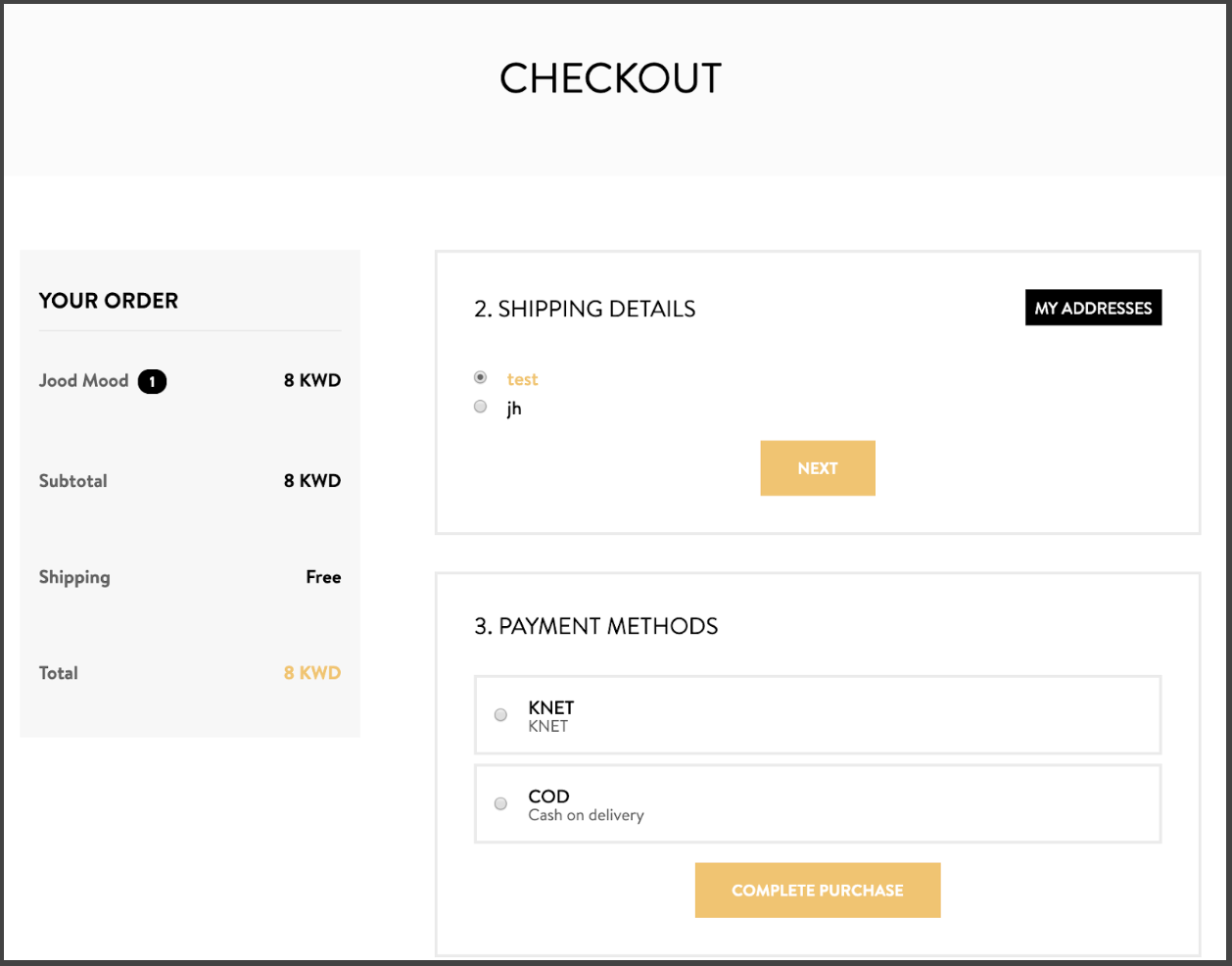


Figure 6.7: Le choix de la méthode de paiment

La gestion du module checkout est regroupé sous src/modules/order Principalement ce module gere :

* + - * Le choix de l’adresse de livraison textbfShipping Details qui permet à l’utilisateur de choisir une addresses existantes dans son profil ou bien ajouter une nouvelle adresse. (Le composant src/modules/order/components/ShippingDetails)
      * Le choix de la méthode de paiement Payment Methods ( en utilisant l’interface de paiement KNet ou l’option du paiement à la livraison COD). (Le composant src/modules/order/com- ponents/Payment)
      * Le calcul du montant de la commande

Après la saisie des informations nécessaires, et l’envoi de la requête l’utilisateur doit s’attendre à l’une des deux choses :

* + - * La redirection vers la plateforme de paiement en ligne KNet pour effectuer le paiement.
      * la redirection vers la page du succès de la commande

**Scenario : Ajout d’une nouvelle adresse** Quand procéder au paiement, le composant Ship- pingDetails fournit une manipulation à double sens, l’utilisateur peut choisir une des adresses lié à son compte, sinon, l’application lui donne la main d’ajouter d’une nouvelle adresse en cas de besoin à l’aide du formulaire suivant

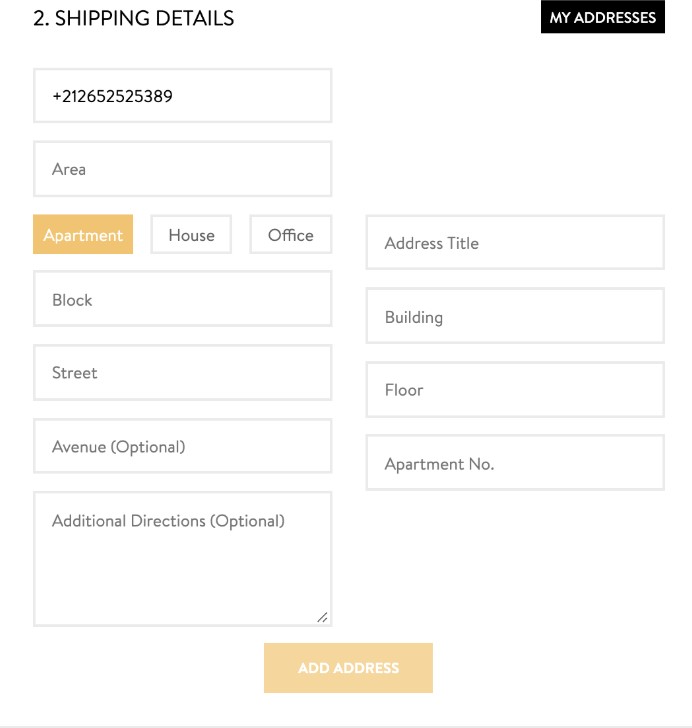


Figure 6.8: Ajout d’une nouvelle adresse

Les adresses différents selon le type de l’habitation (apartment, House, Office), par exemple : une adresse de type House ne contient pas les champs tel que [Building, Floor, Apartment No., Office]

La validation des formulaires est assurée par Yup qui est un validateur de schéma d’objet JavaS- cript et analyseur d’objet alors que le formulaire est assuré par Formik qui est une bibliothèque destinée pour la gestion de tous cas d’utilisation du formulaire.

En plus de ca, la gestion des adresses de l’utilisateur est disponible grâce au menu situé au header

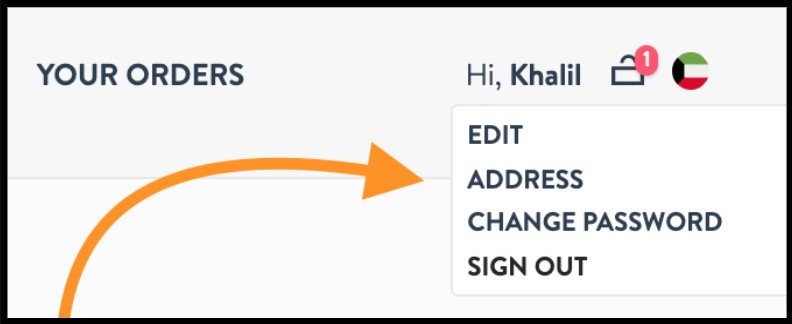


Figure 6.9: Menu de gestion (d’adresses, mot de passe, profile)

Ce qui redirige l’utilisateur vers le composant ListAddressPage :

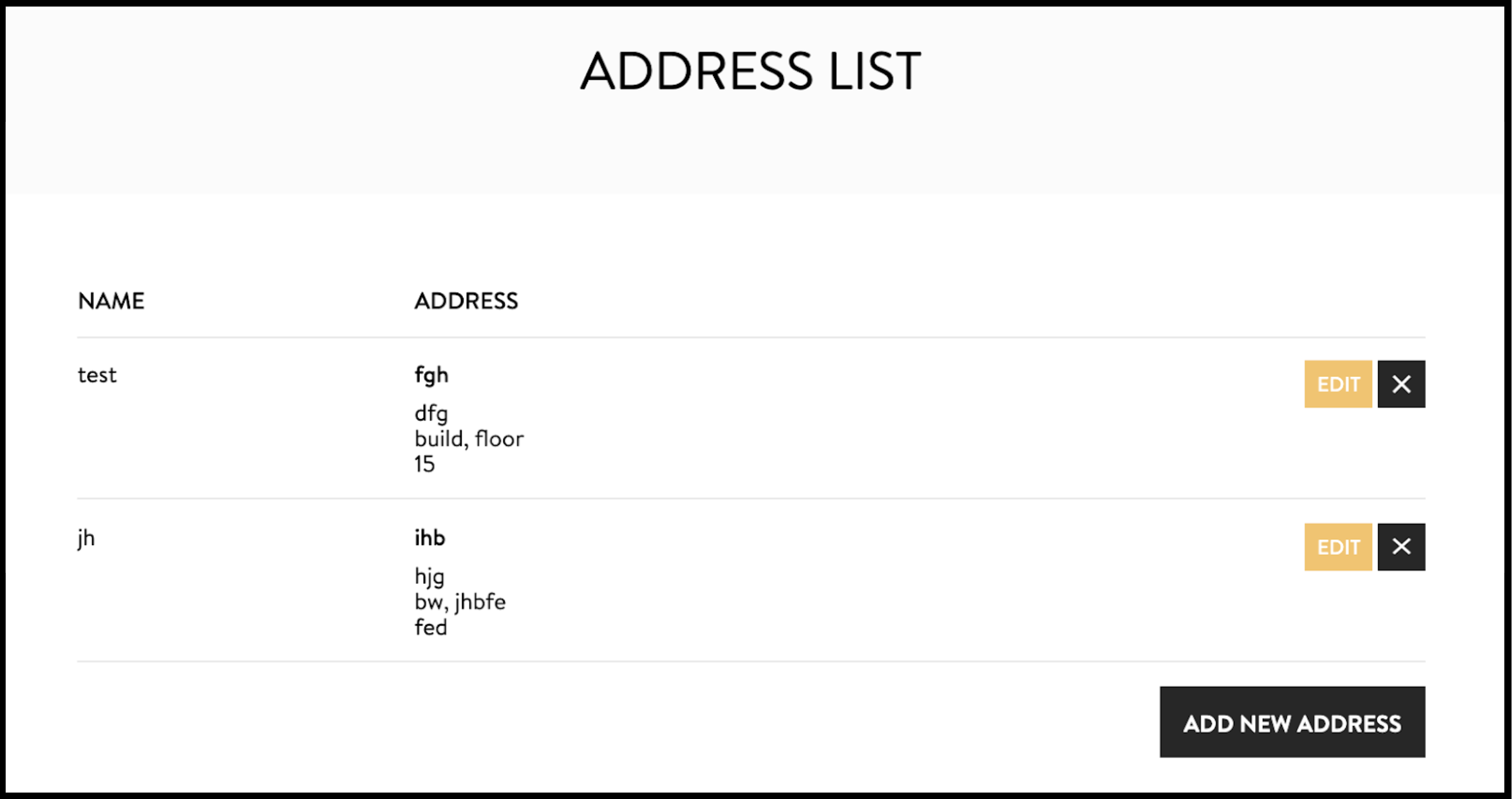


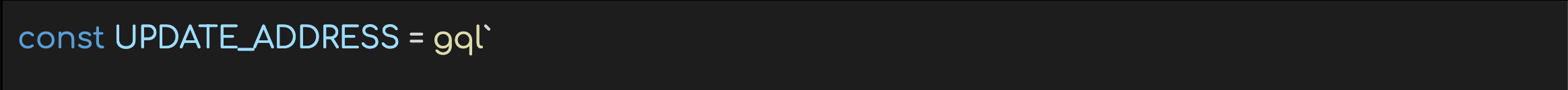
Figure 6.10: Interface de gestion d’adresses

ListAddressPage donne possibilité de gérer les adresses de l’utilisateur grâce aux HOC (Compo- sants d’ordre supérieur) qui interagissent avec l’interface GraphQL :

* + - * withAddAddress : ajout d’une nouvelle adresse en exécutant la mutation suivante

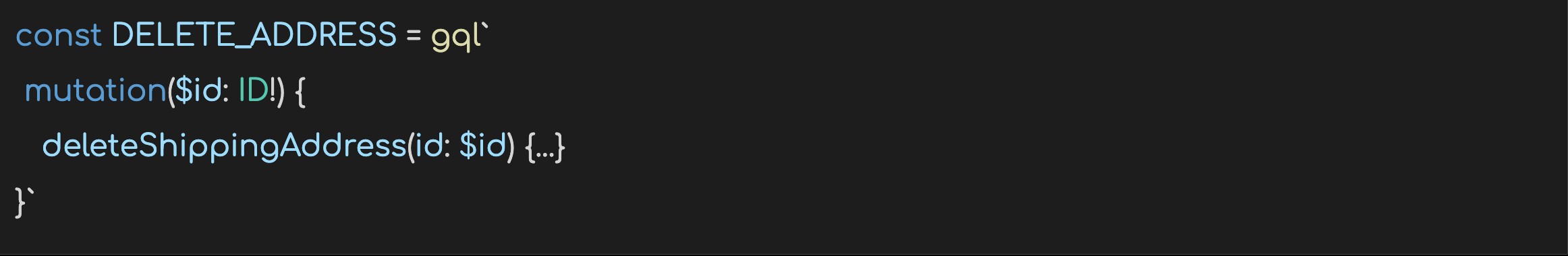


* + - * withUpdateAddress : contient la mutation qui permet la modification d’une adresse





* + - * withDeleteCartItem : contient la mutation qui permet la suppression d’une adresse



#### Autres pages et composants

**Composant Detail Product** sous src/modules/product/container/Detail Visualise les infor- mations importantes d’un seul produit

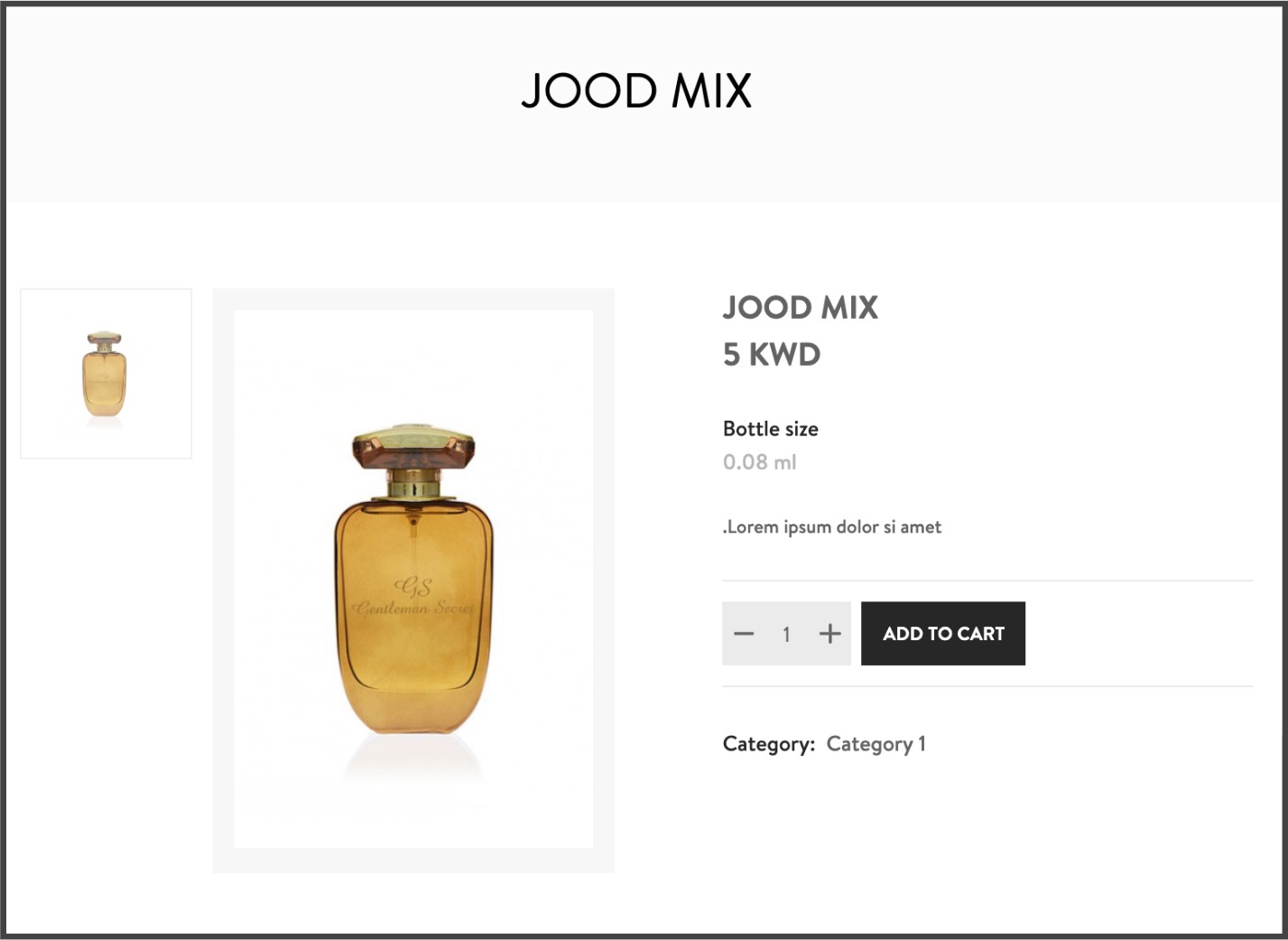


Figure 6.11: détails d’un produit

**Composant Edit Profile** sous src/modules/profile/container/Edit Donne la main à l’utilisa- teur a modifié ses données personnelles

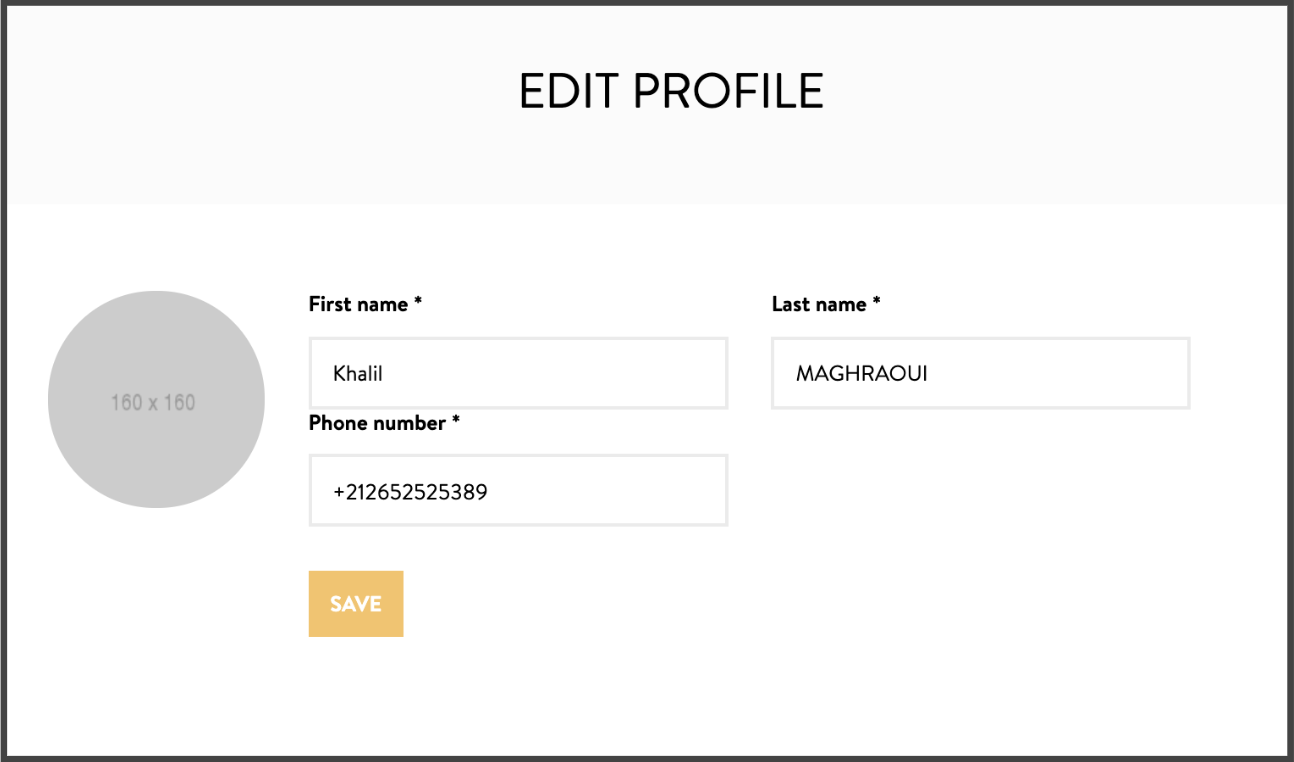


Figure 6.12: Modification des données d’un utilisateur

**Composant Change Password** sous src/modules/profile/container/ChangePassword Donne la main à l’utilisateur à réinitialiser son mot de passe

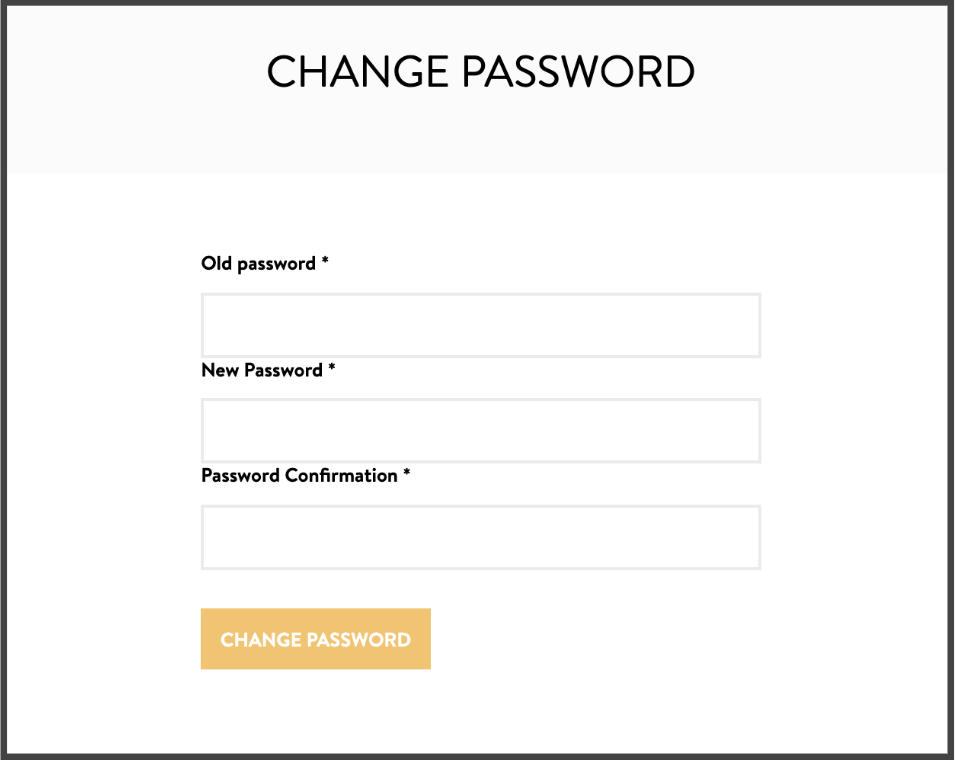


Figure 6.13: Interface pour changer un mot de passe

### Conclusion

Dans ce chapitre, on a pu présenter le fonctionnement de la côté front end, et aussi le fruit de notre travail sur la plateforme de tout en mettant en place les différents outils techniques et logicielles qui nous a permis de concevoir les différents modules de projet.

Le présent rapport représente la synthèse du projet de fin d’études que j’ai effectue au sein de l’organisme Orange Bytes, et qui consiste à le développement d’une plateforme e commerce , concevoir et à implémenter une infrastructure Cloud adaptée pour l’architecture d’une platforme e-commerce.

Notre mission a consisté à contribuer dans partie BackEnd et DevOps dans le développement, la conception et l’implémentation des environments Cloud pour les différents services. Notre mission a consisté aussi sur la maintenance corrective et préventive de la totalité de l’infrastructure Cloud de projet, la correction des différents anomalies .

Nous avons aussi travaillé pour améliorer la sécurité et la performance de la plateforme via la création d’une infrastructure Cloud scalable, hautement disponible et résiliente, augmenter la productivité et accélérer les processus de déploiement et de mise en production par la mise en place d’un systéme d’intégration continue (CI) et de déploiement continu (CD) lié avec l’infrastructure de projet, à cela s’ajoute la mise en œuvre des évolutions souhaitées par le client.

Nous avons également mis en pratique un systéme de Monitoring pour surveiller la santé des environments Cloud de A à Z afin de livrer un projet qui répond aux normes de qualité et qui satisfait le client et l’utilisateur final.

Pour mettre en œuvre notre projet, nous avons été amenés, dans un premier lieu, à établir une étude conceptuelle afin de dégager les différents modules de la plateforme, ainsi qu’une étude des outils et technologies susceptibles de convenir à sa réalisation et la réalisation de sa infrastructure Cloud.

Dans un second lieu, nous avons abordé l’analyse et la conception du projet. Un certain nombre de diagrammes ont été élaborés afin de mieux diviser le projet, ce qui a facilité sa mise en œuvre. Finalement, l’implémentation des différents service et ses environments Cloud dont Le résultat final répond aux besoins déjà cités dans ce rapport.

*AMQP* Advanced Message Queuing Protocol

*API* Application Programming Interface

*AWS* Amazon Web Service

*CD* Continuous Development

*CI* Continuous Integration

*CLI* Commande Line Interface

*CMS* Content Management System

*CPU* Central Processing Unit

*CRUD* Create, Read, Update and Delete

*CSRF* Cross-Site Request Forgery

*CSS* Cascading Style Sheets

*CSV* Comma-Separated Values

*DDD* Domain Driven Design

*DevOps* DEVelopment and OPerationS

*DNS* Domain Name System

*DRY* Don’t Reapeat Yousrself

*ESB* Enterprise Service Bus

*HTML* HyperText Markup Language

*HTTP* HypterText Transfer Protocol

*IaaS* Infrastructure as a Service

*IANA* Internet Assigned Authority

*IP* Internet Protocol

*IT* Information Technology

*ITS* Intelligent Tutorial System

*JPL* Jet Propulsion Laboratory

*JSON* JavaScript Object Notation

*LMS* Learning Management System

*MV C* Model View Controller

*MV T* Model View Template

*NASA* National Aeronautics and Space Administration

*NoSQL* Not only SQL

*ORM* Object Relational Mapper

*PaaS* Platform as a Service

*PDF* Portable Document Format

*REST* REpresentational State Transfer

*RPC* Remote Procedure Call

*RSS* Rich Site Summary

*SaaS* Software as a Service

*SGBD* Système de Gestion de Base de Données

*SI* Système d’Information

*SOA* Service Oriented Architecture

*SOAP* Simple Object Access Protocol

*SQL* Structured Query Language

*SSO* Single Sign-On

*TDD* Test Driven Developpement

*UI* User Interface

*UML* Unified Modeling Language

*URI* Uniform Resource Identifier

*URL* Uniform Ressource Locator

*UUID* Universally Unique IDentifier

*UX* User eXperience

*V M* Virtual Machine

*XML* eXtensible Markup Language

*XSS* CROSS-Site Scripting

|  |  |
| --- | --- |
| **Le 05/02/2019**  **Le 07/02/2019** | [http://http://www.obytes.com/](http://www.obytes.com/)  <https://fr.wikipedia.org/wiki/Python_(langage)> |
| **Le 14/02/2019** | <https://docs.docker.com/> |
| **Le 21/02/2019** | <https://docs.djangoproject.com/en/2.2/> |
| **Le 22/02/2019** | <https://fr.wikipedia.org/wiki/SQLite> |
| **Le 22/02/2019** | <http://www.django-rest-framework.org/tutorial/quickstart/> |
| **Le 01/03/2019** | <https://www.howtographql.com/> |
| **Le 02/03/2019** | https://blog.octo.com/graphql-et-pourquoi-faire/ |
| **Le 03/03/2019** | https://stackabuse.com/building-a-graphql-api-with-django/ |
| **Le 05/04/2019** | [https://www.youtube.com/watch?v=BDBvHOaaKHo&list=PLv2a\_5pNAko0Mijc6mnv0](https://www.youtube.com/watch?v=BDBvHOaaKHo&list=PLv2a_5pNAko0Mijc6mnv04xeOut443Wnk) |
| **Le 05/04/2019** | <https://docs.aws.amazon.com/> |
| **Le 22/04/2019** | <https://learn.hashicorp.com/terraform/> |
| **Le 01/05/2019** | <https://www.terraform.io/docs/providers/aws> |
| **Le 08/05/2019** | <https://www.terraform.io/docs/providers/github/index.html> |
| **Le 15/05/2019** | <https://www.terraform.io/docs/providers/cloudflare> |
| **Le 02/06/2019** | [https://medium.com/devopslinks/deep-dive-into-aws-vpc-and-vpc-peering-](https://medium.com/devopslinks/deep-dive-into-aws-vpc-and-vpc-peering-3ea919bd367a) |
| **Le 15/06/2019** | [https://medium.com/@amolkokje/aws-setup-bastion-host-ssh-tunnel-f5ec5c](https://medium.com/%40amolkokje/aws-setup-bastion-host-ssh-tunnel-f5ec5cf10524) |
| **Le 05/07/2019** | <https://www.lucidchart.com/> |

**Webographie**