Rapport de stage

# BNP Paribas

Alexandre GIBOZ
29 novembre 2021 | 21 janvier 2022

# Table des matières :

# Introduction

Dans le cadre de ma seconde année au sein du **BTS S**ervices **I**nformatiques aux **O**rganisations, option **S**olutions **L**ogicielles et **A**pplications **M**étier, j'ai dû réaliser, sur une durée de huit (08) semaines prenant place du 29 novembre 2021 au 21 janvier 2022, un stage en milieu professionnel.

Ce rapport est un résumé de mon expérience en entreprise au cours de cette période, et mettra en avant l'ensemble des activités et tâches effectuées, ainsi que les connaissances acquises sur la durée.

## Remerciements

Je souhaite débuter ce rapport en remerciant toute l'équipe Digital Delivery Acceleration pour m'avoir accueilli tout au long de ces huit (08) semaines de stage.

J'aimerais également adresser mes remerciements à Christophe Molin, qui m'a accompagné sur toute la durée du stage, et qui, en m'accordant une aide précieuse, m'a permis d'apprendre encore davantage.

Je remercie Selim Tavukcuoglu qui m'a beaucoup apporté sur la partie dédiée à Laravel et à la mise en place du COS.

Pour conclure cette partie, je tenais à remercier Virginie Porquez pour m'avoir aidé à mener à bien les tickets Jira, et à qui j'ai pu, en retour, apporter mon aide en ce qui concerne Docker et le processus de migration.

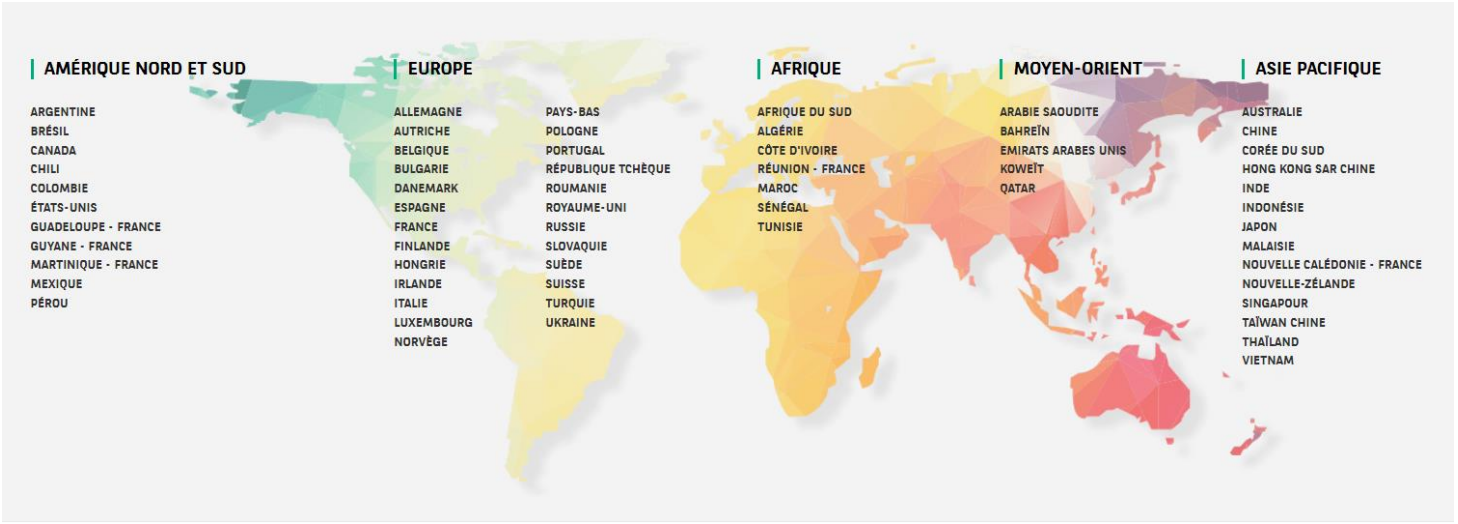
# Présentation de l'organisation

## Historique

La société BNP Paribas émerge en mai 2000 à la suite de l'OPE lancée par BNP sur Paribas. De part son activité et sa rentabilité, elle est actuellement la première banque d'Europe, et une banque influente et omniprésente à travers le monde. Elle est à l'heure actuelle dirigée par Jean-Laurent Bonnafé et Jean Lemierre.

## Structure juridique, taille, capital, CA…

BNP Paribas est une société anonyme s'illustrant principalement en tant que Banque et assurance. La firme a réalisé un bénéfice net de 7,1 milliards d'euros au 31 décembre 2020, et prend part à l'indice CAC 40.
Elle possède un effectif de plus de 193 000 employés au travers du monde, ce qui atteste de son influence et omniprésence.

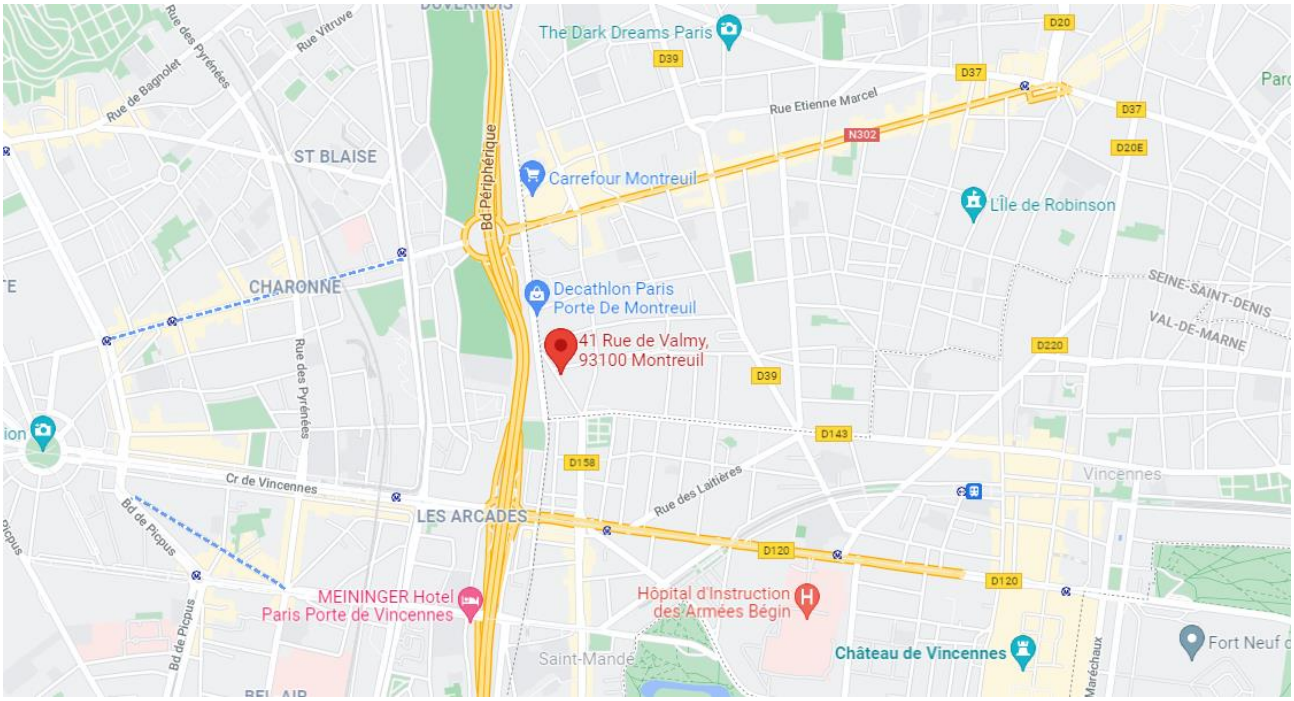



## Situation géographique

Le siège social de la firme est situé au 16 Boulevard des Italiens, dans le neuvième (09e) arrondissement de Paris. BNP Paribas possède une importante branche dédiée à l'informatique. Cette branche, nommée « BDDF IT » est groupée dans le secteur de Valmy, située dans la ville de Montreuil, et se décompose en 8 bâtiments.

J'ai pour ma part travaillé à Valmy 2 :

**Adresse :** 41 Rue de Valmy, 93100 Montreuil




## Système informatique de l'organisation

Chaque employé BNP Paribas de Valmy reçoit un ordinateur et un badge. Ce badge permet de déverrouiller l'ordinateur et d'accéder au Wi-Fi BNP Paribas « First ». L'accès à internet s'effectue via « Cisco AnyConnect », un logiciel qui accroit la sécurité et les restrictions de connexions en passant par du matériel Cisco.

Afin d'accéder à Internet, le VPN intégré doit être activé, et ce via l'identifiant BNP Paribas attribué à chaque employé ou stagiaire.

Le reste des informations concernant le système informatique ne peuvent être divulgués à du personnel ne travaillant pas directement sur le réseau BNP Paribas, je n'y ai donc pas eu accès.

## Présentation de l'équipe DIDA

**Di**gital **D**elivery **A**cceleration (**DIDA**) est une équipe composée d'une dizaine de personnes ayant pour rôle de produire et mettre sur pied un projet dans un délai bien plus court que ceux normalement alloués à d'autres projets, pouvant s'étendre sur plusieurs années.
De plus, il est bien souvent confié à l'équipe des projets dont le développement s'avère trop lent et laborieux, afin qu'ils relancent le projet tombé en désuétude.

Les projets pris en charge par l'équipe sont bien souvent des applications web, ce qui correspond aux attentes de ma formation et à mes compétences actuelles. L'équipe, par ailleurs, prend en charge plusieurs projets à la fois.

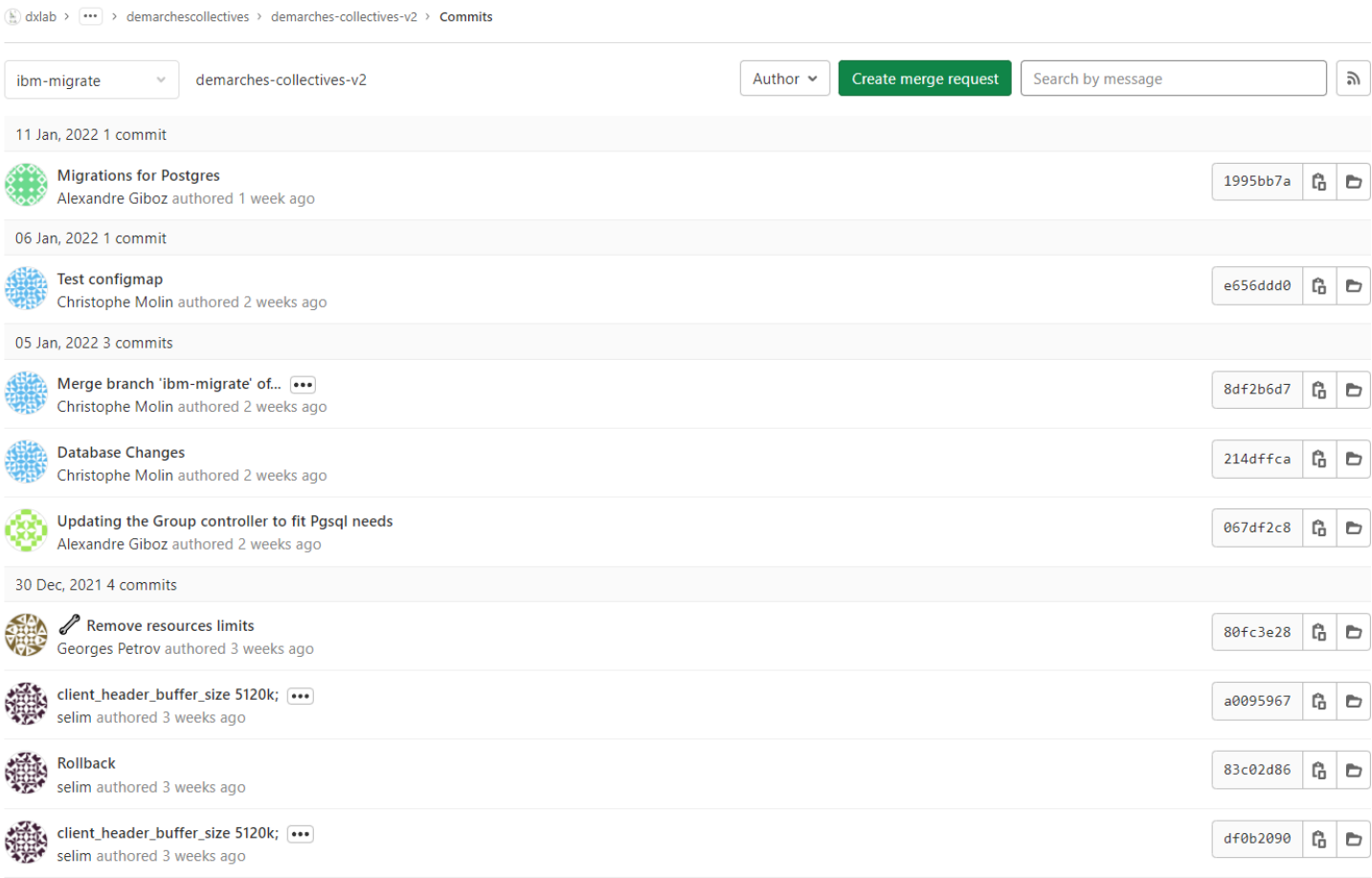
Une part importante des membres de l'équipe sont des « externes » commissionnés par BNP Paribas sur une durée déterminée afin de mener à bien un projet précis qui leur est attribué.

# Environnement de travail

## GitLab

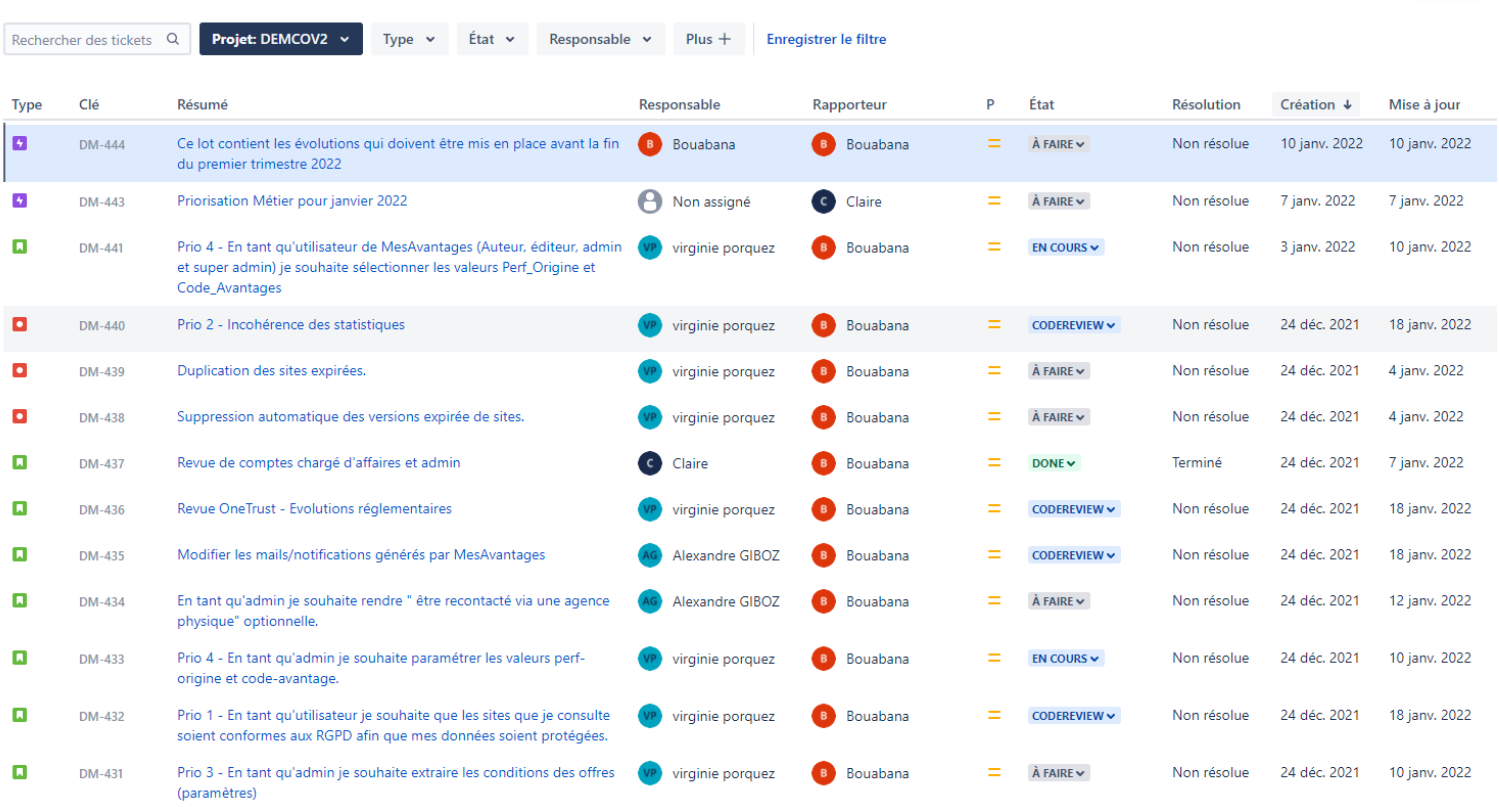
L'avancement du projet progresse sur plusieurs fronts et aspects au même instant. Ainsi, chaque équipe prenant en charge une mission se voit attribuer une « branche » dédiée, une copie de l'espace principal (maître), mais séparée de celle-ci. Cela permet de diviser chaque domaine d'activité et de les fusionner avec la branche maître une fois l'ensemble de l'avancement validé par un supérieur.

Une branche « ibm-migrate » me fût confiée, regroupant l'ensemble des modifications apportées au projet afin que ce dernier puisse fonctionner convenablement avec l'intégralité des nouvelles fonctionnalités et modifications apportées au projet.
En plus de permettre de tenir compte de mon avancement, cela permet de gérer l'état du projet dans sa globalité, et de visualiser ou revenir à une version antérieure si cela s'avère nécessaire.



## Jira

Jira est un logiciel permettant une gestion dite « agile » des projets et un suivi des problèmes rencontrés au cours de leurs développements. Le logiciel permet d'établir des ordres de priorités et de faciliter drastiquement le travail en équipe.
Jira permet donc aux testeurs et développeurs du groupe de signaler un problème à régler tout en pouvant spécifier la priorité de cette requête. Un supérieur hiérarchique peut ainsi confier cette mission à l'un des membres de l'équipe.




| Type | Clé | Résumé | Responsable | Rapporteur | P | État | Résolution | Création | Mise à jour |
|---|---|---|---|---|---|---|---|---|---|
| | DM-444 | Ce lot contient les évolutions qui doivent être mis en place avant la fin du premier trimestre 2022 | Bouabana | Bouabana | = | À FAIRE | Non résolue | 10 janv. 2022 | 10 janv. 2022 |
| | DM-443 | Priorisation Métier pour janvier 2022 | Non assigné | Claire | = | À FAIRE | Non résolue | 7 janv. 2022 | 7 janv. 2022 |
| | DM-441 | Prio 4 - En tant qu'utilisateur de MesAvantages (Auteur, éditeur, admin et super admin) je souhaite sélectionner les valeurs Perf_Origine et Code_Avantages | virginie porquez | Bouabana | = | EN COURS | Non résolue | 3 janv. 2022 | 10 janv. 2022 |
| | DM-440 | Prio 2 - Incohérence des statistiques | virginie porquez | Bouabana | = | CODEREVIEW | Non résolue | 24 déc. 2021 | 18 janv. 2022 |
| | DM-439 | Duplication des sites expirées. | virginie porquez | Bouabana | = | À FAIRE | Non résolue | 24 déc. 2021 | 4 janv. 2022 |
| | DM-438 | Suppression automatique des versions expirée de sites. | virginie porquez | Bouabana | = | À FAIRE | Non résolue | 24 déc. 2021 | 4 janv. 2022 |
| | DM-437 | Revue de comptes chargé d'affaires et admin | Claire | Bouabana | = | DONE | Terminé | 24 déc. 2021 | 7 janv. 2022 |
| | DM-436 | Revue OneTrust - Evolutions réglementaires | virginie porquez | Bouabana | = | CODEREVIEW | Non résolue | 24 déc. 2021 | 18 janv. 2022 |
| | DM-435 | Modifier les mails/notifications générés par MesAvantages | Alexandre GIBOZ | Bouabana | = | CODEREVIEW | Non résolue | 24 déc. 2021 | 18 janv. 2022 |
| | DM-434 | En tant qu'admin je souhaite rendre " être recontacté via une agence physique" optionnelle. | Alexandre GIBOZ | Bouabana | = | À FAIRE | Non résolue | 24 déc. 2021 | 12 janv. 2022 |
| | DM-433 | Prio 4 - En tant qu'admin je souhaite paramétrer les valeurs perf-origine et code-avantage. | virginie porquez | Bouabana | = | EN COURS | Non résolue | 24 déc. 2021 | 10 janv. 2022 |
| | DM-432 | Prio 1 - En tant qu'utilisateur je souhaite que les sites que je consulte soient conformes aux RGPD afin que mes données soient protégées. | virginie porquez | Bouabana | = | CODEREVIEW | Non résolue | 24 déc. 2021 | 18 janv. 2022 |
| | DM-431 | Prio 3 - En tant qu'admin je souhaite extraire les conditions des offres (paramètres) | virginie porquez | Bouabana | = | À FAIRE | Non résolue | 24 déc. 2021 | 10 janv. 2022 |

# Présentation du projet

## MesAvantages BNP Paribas

La plateforme MesAvantages est une application web permettant de diffuser des offres commerciales dédiées à une cible spécifique dans le cadre de démarches collectives ou de partenariats. Le partenaire se verra attribuer un mini-site, récapitulant les offres liées à BNP Paribas qui lui sont associées, ainsi que les évènements organisés entre le groupe en question et la banque.

Ces sites dédiés sont accessibles 24H/7J par des prospects directement sur le site intranet ou internet des entreprises, écoles ou associations partenaires. Ils contiennent également des offres bancaires préférentielles et personnalisées.

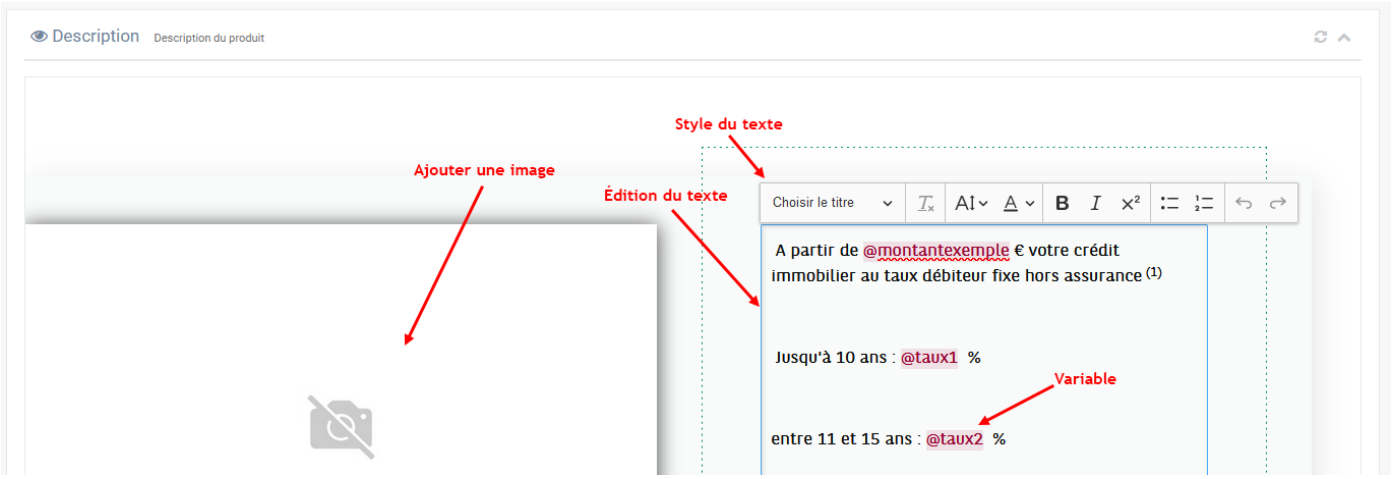
Le cœur de l'application s'approche d'une interface administrateur. Les utilisateurs ont une autorité établie sur quatre niveaux **: Auteur**, **Éditeur de site**, **Administrateur** et **Superadministrateur**
Chaque rôle possède des droits spécifiques, afin d'établir une hiérarchie entre les rôles.
C'est donc depuis cette interface que les opérations sont organisées, et que chaque utilisateur, en fonction de ses droits et privilèges, effectue ses missions.

Les **sites dédiés** (nommé aussi « **minisites** ») sont des sites personnalisés accessibles depuis une URL privatisée envoyée au partenaire. Le site possède une durée limitée, et un renouvellement du site de la part du partenaire doit être effectuée. Si le site n'est pas renouvelé, il apparaîtra comme expiré, et ne sera plus consultable via l'URL.
La création d'un site dédié demande un certain nombre d'éléments conçus au préalable par un administrateur ou un auteur, qui seront directement liés à un « **modèle** ». Cependant, la mise en place des éléments ci-dessous sont requis afin d'établir le modèle.

Une **thématique** est un simple nom associé à une image. Celle-ci regroupera des produits afin de constituer l'arborescence du modèle et donc du site dédié.

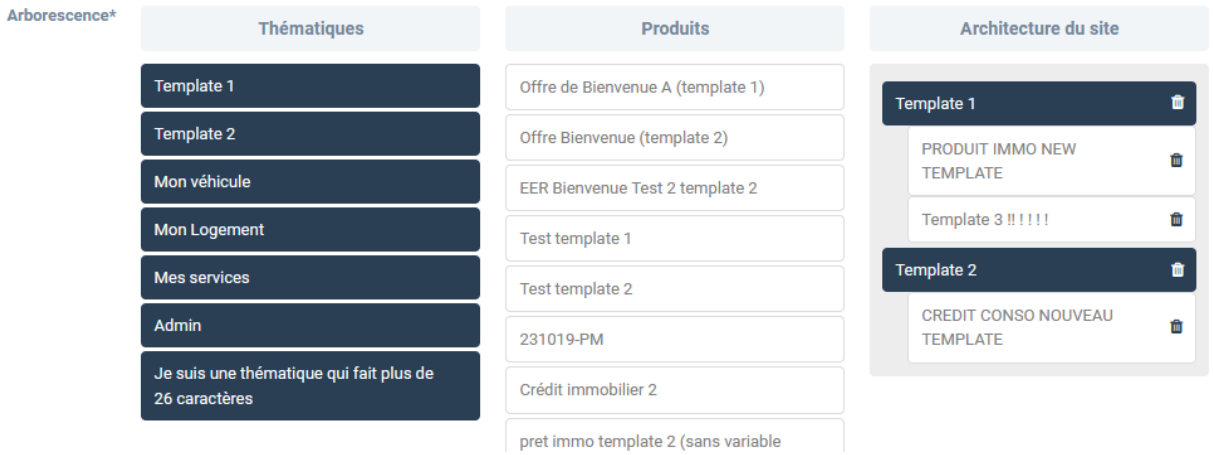
Le **Template** d'un produit définit l'affichage (le design) de la page, ainsi que les différentes sections de celle-ci. Le contenu des sections (textes, images et styles) est à remplir directement sur la page d'édition. Les « **variables** » sont à définir à la fin de la page d'édition, avec éventuellement une valeur par défaut, pour ensuite les intégrer dans les différents corps de texte. Ils apparaitront comme les seuls éléments d'édition (champs à remplir) lors de la création d'un site dédié.

Les variables apparaissent ci-dessous en rouge et en surbrillance :

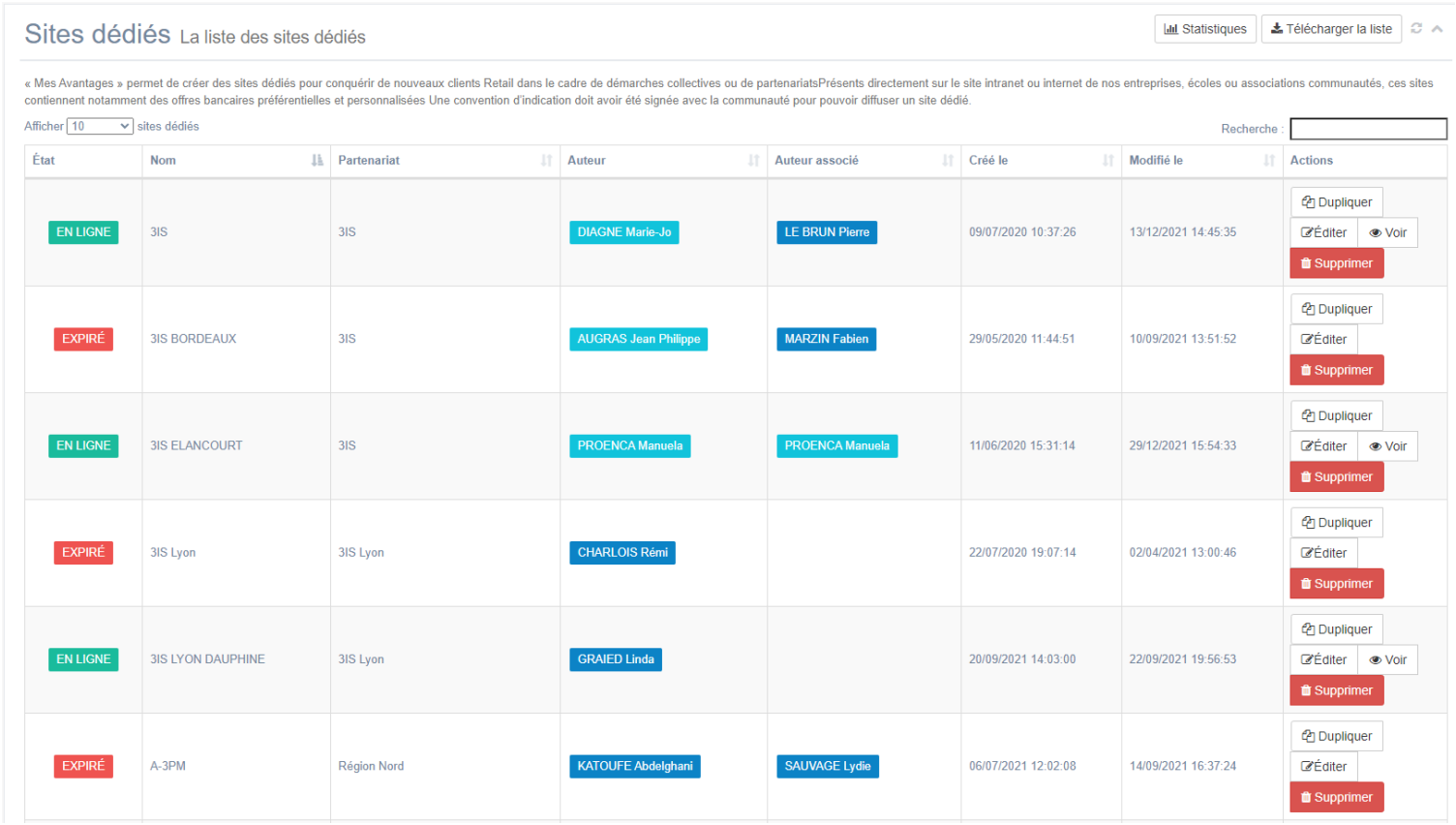



Un **modèle** (nommé aussi « **minisite abstrait** ») est la représentation générique d'un ou plusieurs sites dédiés. C'est celui-ci qui définit tous les éléments qui s'afficheront (images et textes), mais également les offres proposées (combinaisons de produits et de thématiques). Son contenu est composé principalement de textes à trous qui seront ensuite complétés et personnalisés lors de la conception des sites dédiés.

Après avoir défini un **thème**, qui représente l'affichage (le design) du site et de la page d'accueil, les sections de ce dernier sont à remplir de la même manière que pour un produit. De plus, le modèle définit l'arborescence du site : les thématiques et les produits, qui vont donc constituer, une fois édités, les offres du site dédié.

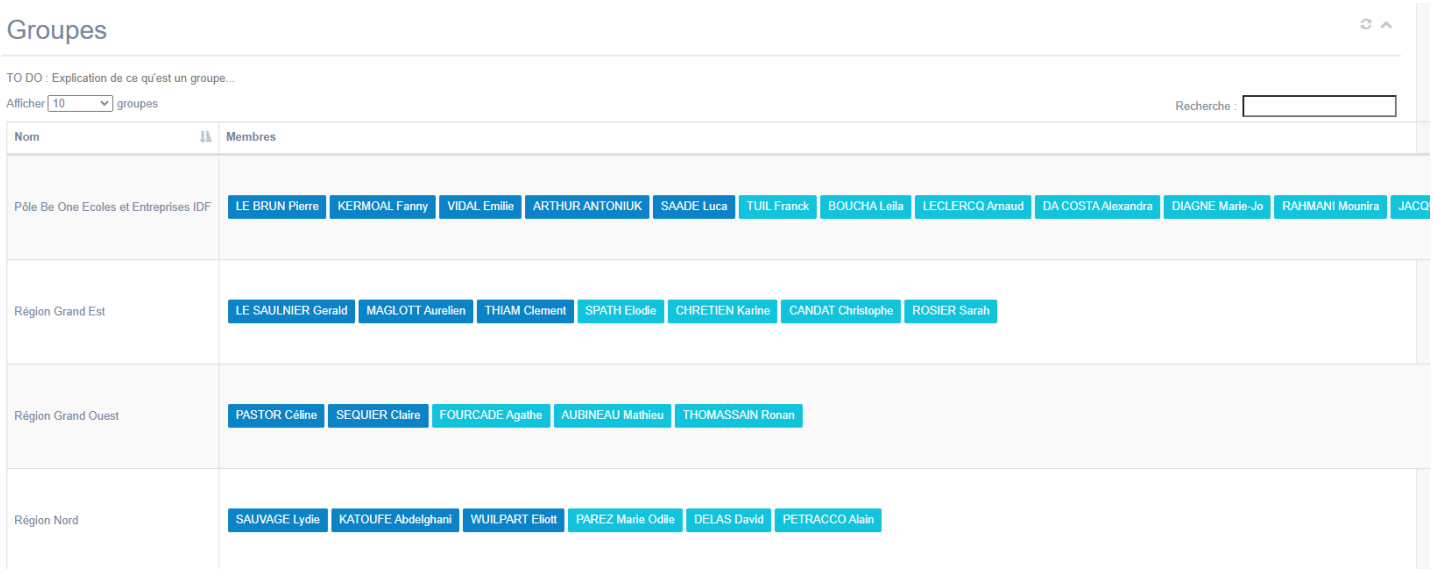


L'état des sites, ainsi que leur apparence et leur rendu final sont accessibles depuis la page principale de l'interface administrateur.

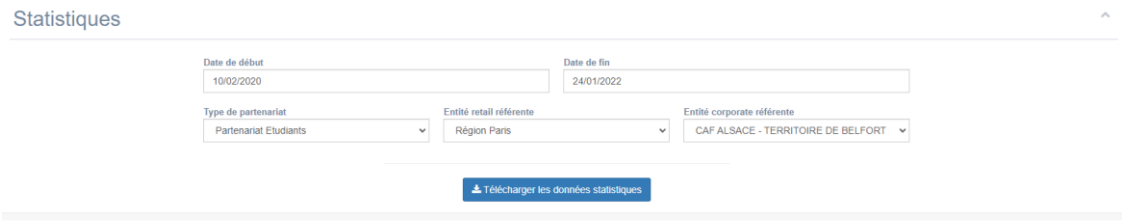



Il est possible de contrôler la date de validité d'un site à partir du bouton « Editer ». Une date de début et de fin de validité sera à indiquer. Une fois la fin de validité franchie, le site ne sera plus consultable et sera répertorié comme « Expiré ».
Un renouvellement du site avec le partenaire devra alors être effectué.

Chaque site est mis en place et géré par un auteur et, bien souvent, un auteur ou éditeur associé. Tous deux appartiennent à un « **groupe** » : un ensemble d'individu liés à un pôle géographique BNP Paribas précis.




Une importante partie statistique est également disponible. Celle-ci permet d'obtenir sur un document Excel un important nombre d'informations sur un ou plusieurs sites.




Voici un exemple des statistiques pouvant être obtenues au travers de cette fonctionnalité :

| | | |
|---|---|---|
| **Nom du site dédié** | **kone client** | |
| **Type de partenariat** | Autre partenariat | |
| **Entité référente Retail** | BDDF | |
| **Entité référente Corporate** | | |
| Date d'export | 2020-12-22 14:51:42 | |
| Date de début | -- | |
| Date de fin | -- | |
| | | |
| **Visiteurs uniques** | **8** | |
| Visiteurs uniques produit: "Offre de Bienvenue A (template 1)" | 4 | https://recettev2.demco.dxlab.fr/partenaire/35959.kone-client/6 |
| Visiteurs uniques produit: "pret immo template 2 (sans variable exemple legal)" | 2 | https://recettev2.demco.dxlab.fr/partenaire/35959.kone-client/7 |
| Visiteurs uniques produit: "Crédit Auto (template 1)" | 1 | https://recettev2.demco.dxlab.fr/partenaire/35959.kone-client/8 |
| **Nombre de vues** | **56** | |
| Nb pages vues en moyenne par visiteur unique | 7 | |
| Nombre de vues HP | 45 | https://recettev2.demco.dxlab.fr/partenaire/35959.kone-client |
| **Nombre de vues produit** | **11** | |
| Nombre de vues produit: "Offre de Bienvenue A (template 1)" | 6 | https://recettev2.demco.dxlab.fr/partenaire/35959.kone-client/6 |
| Nombre de vues produit: "pret immo template 2 (sans variable exemple legal)" | 3 | https://recettev2.demco.dxlab.fr/partenaire/35959.kone-client/7 |
| Nombre de vues produit: "Crédit Auto (template 1)" | 2 | https://recettev2.demco.dxlab.fr/partenaire/35959.kone-client/8 |
| **Actions** | | |
| Nb de personnes ayant cliqué sur "Je suis client" | 1 | |
| Nb de personnes ayant cliqué sur "Je ne suis pas client" | 11 | |
| Nb de personne ayant cliqué sur "Devenir Client" (new) | 6 | |
| Nb de personne ayant cliqué sur "Être recontacté" (new) | 0 | |
| Nb de personnes ayant cliqué sur "Je suis particulier" (new) | 11 | |
| dont ayant validé l'envoi du formulaire de contact (new) | 0 | |
| Nb de personnes ayant cliqué sur "Je suis un professionnel" (new) | 3 | |
| dont ayant validé l'envoi du formulaire de contact (new) | 0 | |
| Nb de personne ayant cliqué sur "Agence de proximité" (new) | 11 | |
| Nb de personne ayant cliqué sur "Agence de ligne" (new) | 4 | |
| Nb d'envoi du formulaire de contact | 36 | |

## Méthodes de conception du projet

MesAvantages est basé sur le Framework « Laravel », qui permet une conception d'applications web relativement rapide et flexible en passant par le langage orienté objet « PHP ». Le projet aborde une architecture dite **M**odèle **V**ue **C**ontrôleur (**MVC**), permettant de compartimenter l'application en trois parties jouant un rôle précis.

Le **S**ystème de **G**estion de **B**ases de **D**onnées **R**elationnelles (**SGBDR**) présentement utilisé pour MesAvantages est « MySQL ».

## Architecture du code

Comme évoqué précédemment, le code aura pour motif d'architecture le MVC, décomposant le code en trois parties principales :

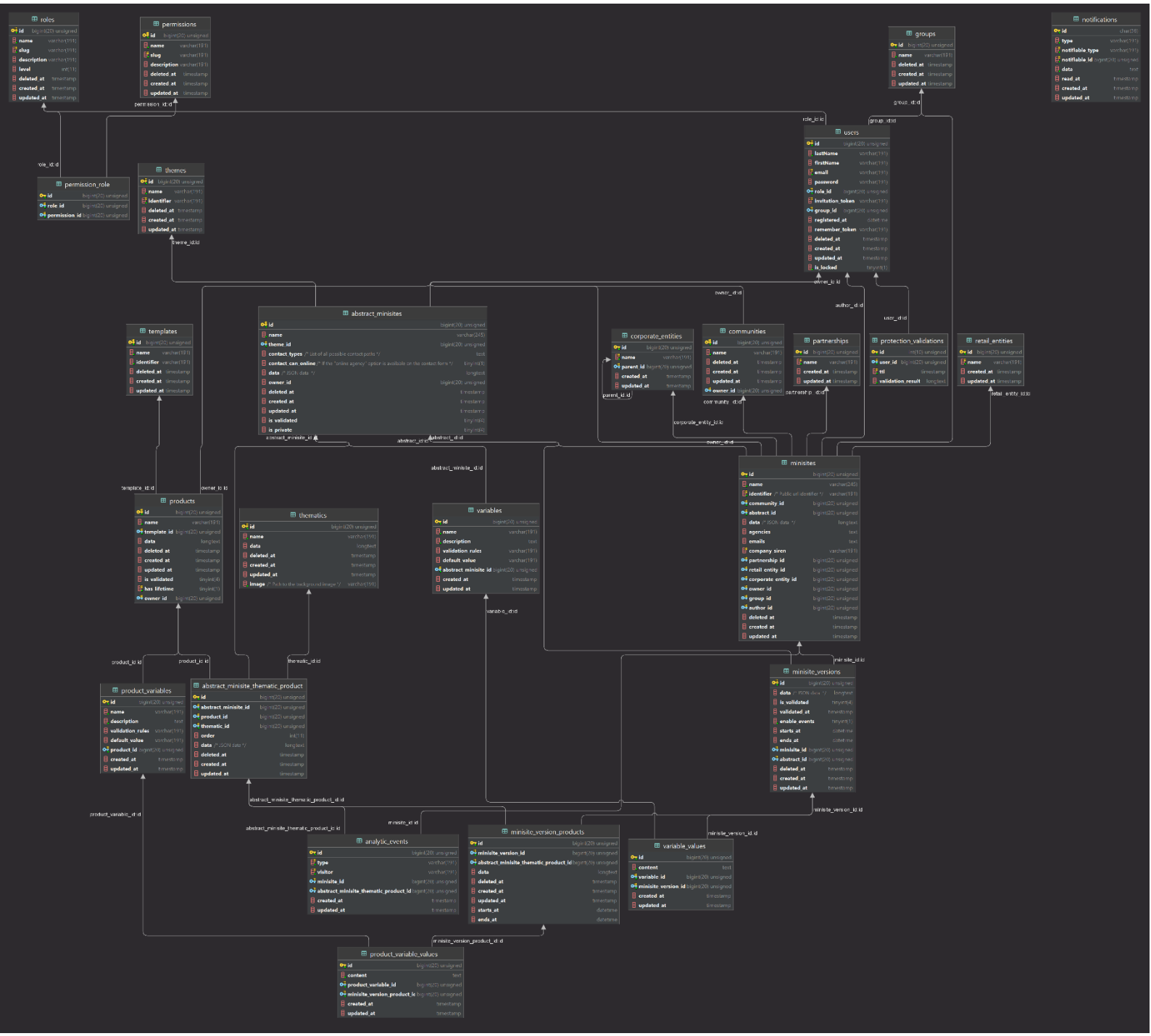
- **Modèle** : Contient l'ensemble des classes métiers relatives à la base de données, et permet ainsi de récupérer les valeurs souhaitées provenant de cette dernière sous la forme d'objets.
- **Vue** : Affichage et partie visible par l'utilisateur. Elle permet d'afficher et de structurer les valeurs récupérées. L'utilisateur peut ainsi interagir avec les données récupérées.
- **Contrôleur** : Squelette du projet. Permet de traiter les actions de l'utilisateurs, de gérer la redirection, et d'effectuer des actions en fonction des données en provenance du modèle.

Laravel inclut également de nombreux autres principes plus avancés, comme la mise en place de « middlewares » permettant d'autoriser ou non l'accès à une page à l'individu envoyant la requête en fonction de ses privilèges, mais encore des fichiers de configurations, permettant d'ajuster les fonctionnalités proposées par Laravel à nos besoins.

En plus de cette base, Laravel fournit de nombreux fichiers relatifs à l'ajout de Librairies ou de Modules externes.

## Architecture de la base de données

La base de données de l'application regroupe un total d'une trentaine de tables. Un **M**odèle **L**ogique de **D**onnées (**MLD**) est disponible ci-dessous, indiquant le nom des tables, ainsi que les colonnes, clefs primaires et étrangères pour chacune. (L'image étant de bonne qualité, zoomer aide à voir convenablement les informations qui y sont indiquées).



# Activités

## Migration des données

### Contexte

L'équipe DIDA a prévu depuis un certain temps de migrer l'ensemble de leurs projets en production actuellement sur une instance MySQL vers une instance PostgreSQL.
Mon rôle sera de migrer la base de données de production rattachée au projet MesAvantages, et de mettre en place, dans un second temps, une documentation expliquant le processus de migration, afin que ce même processus puisse être reproduit ultérieurement, pour ce projet ou pour un autre.

Chaque projet subsistait auparavant sur une instance à part. Chaque projet était donc isolé, ce qui favorise la sécurité et la gestion des données par rapport à une instance groupant plusieurs bases de données en un même endroit, mais revient relativement cher au fur et à mesure que les projets s'accumulent.
C'est principalement pour cette raison que l'ensemble des projets doivent être, une fois migrés vers PostgreSQL, groupés au sein d'une seule et même instance.
De multiples bases de données sensibles étant situées au même endroit, une isolation et un renforcement de la sécurité devra être effectué, principalement via la création de profils uniques possédant les droits adéquats pour chaque projet, mais également par divers autres processus.

MySQL et PostgreSQL sont tous deux des SGBDR abordant de nombreuses similarités, mais comportent toutefois certaines différences qui viennent complexifier toute migration. Ces dissemblances sont majoritairement liées au principe d' « objets », qui concerne, entre autres, les « Séquences » ou les « Schémas ». Bien que les deux SGBDR passent par le langage « **S**elect **Q**uery **L**anguage » (**SQL**), le langage natif est adapté et modelé dans chaque cas. Les requêtes ou actions auront donc une syntaxe et une logique différente.

La migration sera effectuée via l'outil « PgLoader », qui permet des migrations de bases de données de MySQL vers PostgreSQL. PgLoader a pour qualité principale, contrairement à d'autres solutions envisagées, comme « AWS Schema Conversion Tool », de fonctionner au travers d'un script ou de nombreux paramètres peuvent y être précisés. PgLoader offre bien plus de liberté et est de ce fait bien moins restrictif qu'un outil comme celui proposé par Amazon.

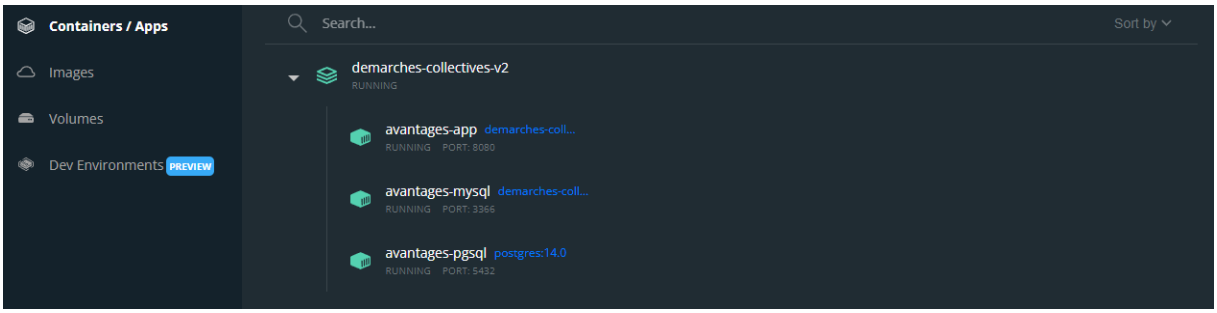
Afin de mener le processus de migration dans les meilleures conditions possibles, et de transvaser les données de MySQL à un système Postgres, il convient de mettre en place un environnement adéquat et configuré de manière optimale. C'est ainsi que « Docker » intervient.

## Présentation de « Docker »

Docker est un logiciel permettant d'empaqueter des applications, et de les isoler de la machine principale. Le principe est d'attribuer à chaque partie de l'application une « image » qui lui sera dédiée (une image MySQL 5.7 ou Redis par exemple). Chaque morceau de l'application sera par la suite placé dans un « conteneur », une machine virtuelle très légère et entièrement isolée de l'environnement de l'hôte, contenant l'image ainsi que d'autres informations annexes apportées. Docker permet toutefois à ces mêmes conteneurs de communiquer entre eux, en leur attribuant le même « Réseau ».

Le principal avantage à mettre en place un environnement dédié à l'application est de travailler dans un contexte qui sera relativement identique à l'environnement de production, et d'obtenir au plus vite une configuration adaptée au projet. C'est un gain de temps.

Les conteneurs permettent également une plus grande flexibilité et rapidité de mise en place du fait de leur volatilité. Les conteneurs peuvent être gérés directement par invite de commande ou par le logiciel dédié, et peuvent donc être redémarrés ou stoppés à tout moment, et ce en l'espace d'un instant. Ainsi, si une erreur survient sur l'un des conteneurs, il peut être remis en service aisément et dans un délai réduit.



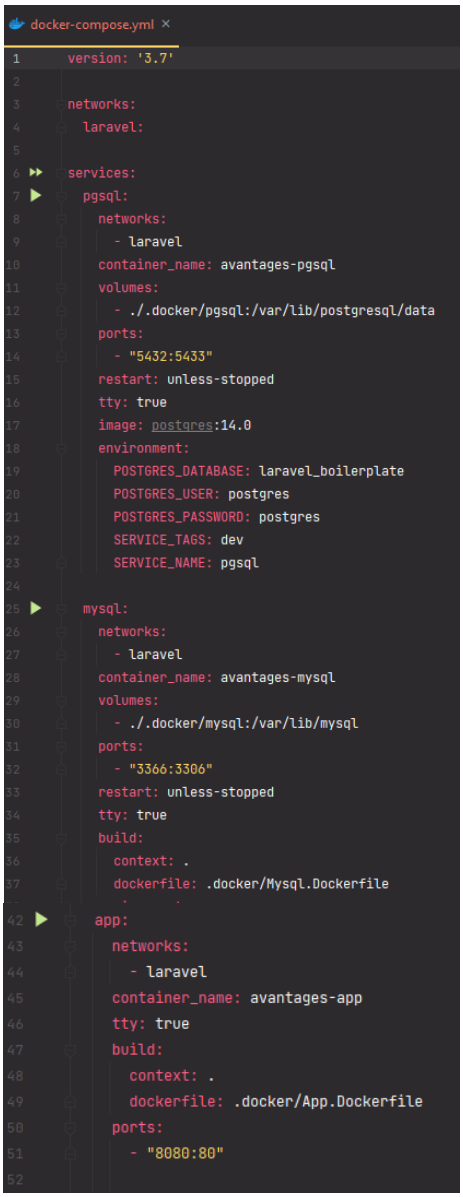

Un suivi complet de la mise en service et des interactions avec chaque conteneur est simple d'accès, ce qui permet d'optimiser la sécurité et la configuration de son application.

## Mise en place d'un environnement local via Docker

Un total de 3 images seront nécessaires afin de mener à bien la migration, chacune dans un conteneur dédié :

Un conteneur MySQL comprenant la base de données à migrer, un conteneur PostgreSQL recevant la nouvelle base de données, et un conteneur PHP comprenant l'application Laravel.

Étant donné que plusieurs conteneurs sont attendus, la meilleure solution est de passer par un fichier « docker-compose ». Contrairement à un fichier « .Dockerfile » classique, un docker-compose permet la mise en place de multiples conteneurs de manière plus organisée et en un seul document. Le fichier compose prendra la forme d'un fichier « yml ».



```yaml
version: '3.7'

networks:
  laravel:

services:
  pgsql:
    networks:
      - laravel
    container_name: avantages-pgsql
    volumes:
      - ./.docker/pgsql:/var/lib/postgresql/data
    ports:
      - "5432:5433"
    restart: unless-stopped
    tty: true
    image: postgres:14.0
    environment:
      POSTGRES_DATABASE: laravel_boilerplate
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: postgres
      SERVICE_TAGS: dev
      SERVICE_NAME: pgsql

  mysql:
    networks:
      - laravel
    container_name: avantages-mysql
    volumes:
      - ./.docker/mysql:/var/lib/mysql
    ports:
      - "3366:3306"
    restart: unless-stopped
    tty: true
    build:
      context: .
      dockerfile: .docker/Mysql.Dockerfile

  app:
    networks:
      - laravel
    container_name: avantages-app
    tty: true
    build:
      context: .
      dockerfile: .docker/App.Dockerfile
    ports:
      - "8080:80"
```

On commence par définir un réseau (Network) sur lequel les trois conteneurs pointeront, afin que ces derniers puissent communiquer les uns avec les autres.

La partie « services » permet de définir chaque conteneur. On attribue à chacun un nom de service explicite, à savoir « pgsql », « mysql » et « app ».

La mise en place des conteneurs MySQL et PostgreSQL est relativement similaire dans la méthode. On définit pour chacun le port correspondant, à la fois pour la machine virtuelle et l'hôte. Le paramètre « tty » nous permet d'accéder et de communiquer avec le conteneur à partir d'une invite de commande.

On définit l'image PostgreSQL, suivi de sa version. Ici, on souhaite préciser une version récente et stable du SGBD, la version 14.0 a donc été retenue.

On remarque cependant une différence importante dans la conception et la mise en place des deux autres conteneurs, le conteneur PostgreSQL possède une image directement définie dans le fichier docker-compose, là où celles de MySQL et PHP ne semblent pas apparaitre.

Docker offre en réalité deux méthodes de mise sur pied d'un conteneur :

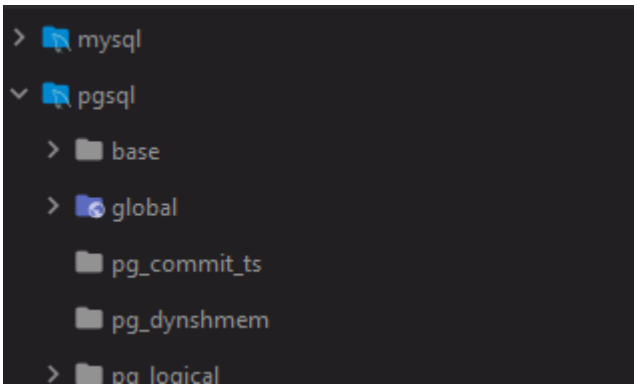
- La première consiste à définir une image directement dans le fichier docker-compose pour les conteneurs ne nécessitant aucun ajout ou configuration spécifique, ce qui est le cas du conteneur PostgreSQL.
- La seconde méthode requiert une attention plus particulière, et nécessite qu'un « build » dédié au conteneur soit définit.

Ce « build » va permettre de paramétrer le conteneur depuis un fichier à part, la partie « build » du fichier de composition servant à faire le lien entre ce nouveau fichier et le conteneur.

La méthode de build s'oppose donc, finalement, à celle qui consiste à l'indication d'une image directement dans le fichier de composition.

L'application Laravel nécessitant l'installation de NodeJS et Composer, et l'installation de l'outil de migration prenant place au sein du conteneur MySQL, ceux-ci nécessiteront un fichier de mise en place dédié, c'est pourquoi un build respectif leur est rattaché.

Les conteneurs sont, par défaut, prévus pour être volatiles. Il est donc primordial de rendre le contenu des SGBD persistant, leur contenu et configuration allant disparaitre à chaque relancement du conteneur. C'est pourquoi des « volumes » sont spécifiés dans le fichier de composition. On y définit le chemin vers le dossier du projet qui va contenir l'ensemble de ces informations.




On ajoute donc deux fichiers au projet : un dédié au conteneur MySQL, et l'autre au conteneur PostgreSQL.
Un volume pour l'application Laravel est dispensable, cette dernière étant déjà finalisée, et représentant un espace conséquent.

Les volumes comprendront, entre autres : les utilisateurs, les tables, les fichiers de configurations, ou encore le fichier responsable du filtrage des adresses IP.

## Configuration des conteneurs

Les conteneurs étant à présent en place, il convient de les configurer proprement.
Les fichiers de build sont des « Dockerfiles » standards. Chaque ligne représente, de manière générale, une commande qui sera exécutée lors de la construction du conteneur. C'est une suite d'instructions qui nous permet de configurer automatiquement le conteneur dès son lancement avec l'ensemble des dépendances et logiciels nécessaires à son bon fonctionnement.

```
FROM php:7.4.9-apache

USER root

WORKDIR /var/www/html/

# NodeJs
RUN rm -rf /var/lib/apt/lists/ && curl -sL https://deb.nodesource.com/setup_12.x | bash -
RUN apt-get install nodejs -y

# Clear cache
# RUN apt-get clean && rm -rf /var/lib/apt/lists/*

# Dependencies
RUN apt-get update && apt-get install -y \
    libjpeg62-turbo-dev \
    libfreetype6-dev \
    libzip-dev \
    libpq-dev \
    zip \
    unzip \
    git \
    libonig-dev \
    nano \
    postgresql \
    postgresql-contrib \
    curl
# Apache URL rewrite
RUN a2enmod rewrite
```

Conteneur Application Laravel :

Le fichier débute par la valeur « FROM », permettant de définir l'image allant s'appliquer au conteneur. L'image sélectionnée est une version 7.4.9 de PHP comprenant également le serveur web Apache.
L'application Laravel nécessite l'installation de NodeJS afin de gérer au mieux certaines fonctionnalités du site web, ainsi que d'autres dépendances, comme Git, ou encore cURL.

```
# Copy Laravel project
COPY . /var/www/html

# Composer
RUN curl -sS https://getcomposer.org/installer | php -- --install-dir=/usr/local/bin --filename=composer
```

On copie par la suite la totalité du projet Laravel dans le conteneur PHP. On installe par la même occasion « Composer », qui est un gestionnaire de dépendances nécessaire au bon fonctionnement de Laravel et des librairies utilisées au sein de MesAvantages.

Conteneur MySQL :

```
FROM mysql:5.7

RUN apt-get update && apt-get install -y \
     make \
     nano \
     curl

# Getting PgLoader latest release to this day and extracting tarball
RUN curl -fsSLO https://github.com/dimitri/pgloader/archive/v3.6.2.tar.gz
RUN tar xvf v3.6.2.tar.gz

# Make up pgloader and moving it to the correct directory
RUN cd pgloader-3.6.2 && make pgloader
RUN mv pgloader-3.6.2/build/bin/pgloader /usr/local/bin/
```

La première contrainte pour cette migration est l'utilisation d'une version antérieure à la version 5.8 de MySQL, l'outil de migration rencontrant des difficultés à gérer les migrations à partir de versions ultérieures à celle-ci. La version 5.7 a donc été retenue.

L'ensemble des commandes suivant le paramètre « FROM » ont pour but de récupérer pgLoader, et de le placer dans le bon répertoire.

## Migration des données de « Staging » localement via PgLoader

L'environnement étant à présent configuré convenablement, il requiert dorénavant un script qui va rendre cette migration possible.

Script pour pgLoader :

```
LOAD DATABASE
     FROM mysql://root:changeme@mysql:3306/laravel_boilerplate
     INTO postgresql://postgres:postgres@pgsql:5432/mesavantages

SET PostgreSQL PARAMETERS
      maintenance_work_mem to '128MB',
      work_mem to '12MB'

SET MySQL PARAMETERS
      net_read_timeout  = '120',
      net_write_timeout = '120'

-- Avoid incompatible type error (numeric and bigint)
CAST type bigint with extra auto_increment to bigserial drop typemod,
     type bigint when (= 20 precision) to bigint drop typemod,

     -- Excluding timestamp modifications (timezones-related problems)
     type datetime to timestamp drop typemod,
     type timestamp to timestamp drop typemod

-- Elements managed
-- Add "quote identifiers" to keep the case sensitivity of the columns
WITH include drop, truncate, create tables, create indexes, drop indexes, drop schema

-- Renaming used schema to public (default: laravel_boilerplate)
ALTER SCHEMA 'laravel_boilerplate' RENAME TO 'mesavantages_prod'

-- Creating the accurate schema before actions
BEFORE LOAD DO
   $$ create schema if not exists public; $$;
```

Le fichier de migration se décompose en plusieurs parties, jouant chacune un rôle précis :

- **LOAD DATABASE** : Cette partie sert à définir les « data sources », à savoir les éléments servant à se connecter aux deux bases de données. On y précisera le type de SGBDR, l'utilisateur accompagné de son mot de passe, l'hôte et le port associé au service. Pour finir, le nom de la base de données ciblée sera à indiquer.
  On apporte tout d'abord une précision au paramètre « FROM », qui représente la base de données MySQL d'où proviennent les données, avant de définir « INTO », qui représente la base de données allant recevoir le résultat de cette migration.
- **SET :** Deux paramètres « SET » se suivent, chacun permettant une meilleure gestion de la mémoire pour le SGBD qui lui est attribué.
- **CAST :** Le paramètre « CAST » permet d'apporter de spécifier une conversion de type souhaitée. PgLoader convertis un grand nombre de types par lui-même, mais certaines erreurs ou difficultés peuvent être rencontrées dans des situations plus particulières, ainsi, certains types de données (notamment ceux liés à l'horodatage) doivent être converties manuellement.

Dans ce cas précis, on indique au script notre volonté d'obtenir un horodatage excluant les fuseaux horaires :

Avec UTC : 2022-01-21 12:05:00+01
Sans UTC : 2022-01-21 12:05:00

Il fut constaté ultérieurement que la mention d'un horodatage comprenant un fuseau horaires pose un réel problème et empêche la majorité des fonctionnalités de l'application de subsister.

- **WITH :** Permet de spécifier l'ensemble des commandes SQL que PgLoader est autorisé à prendre en compte et effectuer lors de la migration. Cette partie se révèle très utile si on ne souhaite effectuer qu'une migration partielle de la base de données, par exemple : migrer uniquement la structure, sans les données qui vont avec.

- **ALTER SCHEMA :** Cette partie est une requête SQL générique. Étant donné que MySQL ne possède pas ce principe de « schémas », PgLoader va par défaut créer un schéma portant le nom de la base de données MySQL, et y placer la résultante de la migration. Cette partie a donc pour rôle de renommer le schéma, afin de lui donner un nom convenable une fois sur l'instance PostgreSQL de production.

La flexibilité et l'agilité que permet PgLoader est relativement bien illustrée dans ce cas précis. De plus, PgLoader parvient à résoudre de nombreuses problématiques de lui-même, le principe de « Séquences » illustrant le mieux cette capacité.

MySQL utilise le concept d'« Auto-incrémentation », qui consiste en l'augmentation d'une valeur d'un nombre précis (par défaut de 1) sur un champ, et ce chaque fois qu'une nouvelle donnée est ajoutée à la table. Cette méthode est en général très communément utilisée afin d'attribuer un identifiant unique à une valeur (en période de développement).

Cependant, PostgreSQL possède une approche relativement différente de MySQL, et contraint à passer par ces séquences mentionnées ci-dessus.
Une séquence prend la forme d'une table, et permet au champ auto-incrémenté d'avoir une valeur de départ, ainsi qu'une valeur d'incrémentation.

Mise en place d'une séquence :

```
1  create sequence users_id_seq;
2
3  alter sequence users_id_seq owner to postgres;
4  alter sequence users_id_seq owned by users.id;
5
6  alter sequence users_id_seq increment 11;
7  alter sequence users_id_seq minvalue 1;
```

De la même manière que les tables ou tout autre objet, il est possible de donner un propriétaire à la séquence.
La séquence, par défaut, n'est attribuée à aucun champ d'aucune table si cette dernière est mise en place après la création de la table ciblée. Il est donc nécessaire de l'appliquer à un champ précis, ce qui est fait ici à la ligne 4 (champ « id » de la table « users »).
Dans le cas présent, on attribue une séquence aux identifiants de la table dédiée aux comptes utilisateurs. On lui attribue un propriétaire, une valeur minimale attribuable par la séquence, ainsi qu'une incrémentation de 11 pour chaque nouvelle donnée instanciée.

L'ensemble de la structure de la base de données ainsi que son contenu ont été migrée, cependant, il convient d'adapter la syntaxe des requêtes présentes dans l'application Laravel à PostgreSQL.

## Mise à jour du projet Laravel

Le principal problème pouvant faire surface lors d'une migration d'un SGBD vers un autre est la forme que prennent les requêtes. En effet, MySQL et PostgreSQL exploitent tous deux une version généralisée de SQL, mais l'adaptent à leur propre cas et implémentent leurs propres fonctionnalités. Le résultat s'avère, finalement, relativement différent sur de nombreux points.
Laravel présente l'avantage non-négligeable de permettre au développeur de changer de SGBD aisément et rapidement, ce en passant par un fichier d'environnement dédié :

Le fichier d'environnement (.env) est un fichier directement rattaché aux fichiers de configuration du Framework. Il permet la gestion des data sources d'une base de données, mais également de son type.

```
DB_CONNECTION=pgsql
# Database host IP od DNS name (e.g. 127.0.0.1, localhost, mysql (for docker-compose))
DB_HOST=127.0.0.1
DB_PORT=5432
DB_DATABASE=mesavantages_laravel
DB_USERNAME=admin-mesavantages
DB_PASSWORD=eEKD3IJ291?kls992Jkslmd90DJkjdkJDInKl09
```

Ce document permet également la gestion de nombreux autres paramètres, à savoir l'implémentation d'un test de Turing (Captcha), ou encore la mise en place d'un protocole d'envoi de messages (SMTP).

Il est donc possible, grâce à ce document, de passer au Framework le type du SGBD utilisé. Laravel s'occupera d'adapter une importante partie des requêtes SQL, notamment pour ce qui est des Seeders et Migrations Laravel. On indique ici, pour le paramètre « DB_CONNECTION », que le SGBD utilisé sera PostgreSQL (pgsql).
Cependant, le fichier d'environnement de base n'inclut pas de paramètres permettant de choisir un « Schéma Postgres » à utiliser, ce qui pose un réel problème car l'application va, par défaut, utiliser le Schéma public, qui est vide et inutilisé.
Il est donc nécessaire d'intervenir sur les fichiers de configuration, et d'ajouter une fonctionnalité nous permettant de préciser au Framework sur quel schéma se baser.

Les fichiers de configuration Laravel prennent la forme de tableaux PHP, avec une valeur associée à chaque clef du tableau. Les clefs sont ici les paramètres de l'application et des différents SGBD compatibles :

```
'sqlite' => [
    'driver' => 'sqlite',
    'database' => env('DB_DATABASE', database_path('database.sqlite')),
    'prefix' => '',
],

'mysql' => [
    'driver' => 'mysql',
    'host' => env('DB_HOST', '127.0.0.1'),
    'port' => env('DB_PORT', '3306'),
    'database' => env('DB_DATABASE', 'forge'),
    'username' => env('DB_USERNAME', 'forge'),
    'password' => env('DB_PASSWORD', ''),
    'unix_socket' => env('DB_SOCKET', ''),
    'charset' => 'utf8mb4',
    'collation' => 'utf8mb4_unicode_ci',
    'prefix' => '',
    'strict' => false,
    'engine' => null,
],

'pgsql' => [
    'driver' => 'pgsql',
    'host' => env('DB_HOST', '127.0.0.1'),
    'port' => env('DB_PORT', '5432'),
    'database' => env('DB_DATABASE', 'forge'),
    'username' => env('DB_USERNAME', 'forge'),
    'password' => env('DB_PASSWORD', ''),
    'charset' => 'utf8',
    'prefix' => '',
    'schema' => 'public',
    'sslmode' => 'prefer',
],
```

L'ensemble des SGBD compatibles nativement avec Laravel sont listés dans ce fichier. Ils sont au nombre de trois (03), mais dans notre cas, uniquement la partie dédiée à PostgreSQL nous préoccupe.
À noter que les valeurs auxquelles les tableaux sont associés sont celles qui peuvent être précisées dans l'option « DB_CONNECTION » du fichier d'environnement, c'est pourquoi ce dernier se voit attribué la valeur « pgsql », et non « postgres », « psql » ou autres.

Les paramètres accessibles dans le fichier env font tous appel à une fonction « env » nécessitant deux paramètres : Le nom de paramètre dédié, puis une valeur qui lui sera attribuée par défaut.
Si ce deuxième paramètre est représenté par des guillemets vides, alors aucun paramètre par défaut n'est précisé. Le paramètre par défaut s'applique si et seulement si le paramètre est non-renseigné dans le fichier d'environnement.

Ici, le paramètre « schema » se base bel et bien sur le schéma public, mais sans possibilité pour le développeur de le modifier depuis l'environnement.

```
'pgsql' => [
    'driver' => 'pgsql',
    'host' => env('DB_HOST', '127.0.0.1'),
    'port' => env('DB_PORT', '5432'),
    'database' => env('DB_DATABASE', 'forge'),
    'username' => env('DB_USERNAME', 'forge'),
    'password' => env('DB_PASSWORD', ''),
    'charset' => 'utf8',
    'prefix' => '',
    'schema' => env('DB_SCHEMA', 'public'),
    'sslmode' => 'prefer',
],
```

Dorénavant, il sera possible de définir le schéma depuis l'espace adéquat, grâce au paramètre « DB_SCHEMA ». Le schéma public sera tout de même utilisé par défaut, ce dernier étant généralement utilisé durant les phases de développement.

Il est à présent nécessaire de modifier certaines requêtes pouvant poser problèmes au sein des contrôleurs. Cela est principalement dû au fait que le développeur n'écrive jamais une réelle commande SQL, mais passe par un ensemble de fonctions proposées par Laravel. Le Framework convertira ensuite le résultat en requête SQL.

Exemple de requête Laravel :

```
protected function dataTableQuery()
{
    return Group::query()
        ->select(
            'groups.id',
            'groups.name',
            'groups.created_at',
            'groups.deleted_at',
            'groups.updated_at'
        )
        ->leftJoin('users', 'users.group_id', '=', 'groups.id')
        ->selectRaw('COUNT(users.id) AS '.'users_counter'.'')
        //->whereNull('groups.deleted_at')
        ->groupBy('groups.id');
}
```

Cette requête donnera, pour finir, le résultat suivant :

```
SELECT COUNT(*) AS AGGREGATE FROM (SELECT groups.id, groups.name, groups.created_at,
                                   groups.deleted_at, groups.created_at, COUNT(users.id) AS user_counter
                                   FROM groups LEFT JOIN "users" 1<->0..n: on users.group_id = groups.id
                                   GROUP BY groups.id) COUNT_ROW_TABLE;
```

Ces opérations restent un ensemble de détails, mais une révision complète de la totalité des contrôleurs doit impérativement être effectuée afin que les fonctionnalités de l'application web fonctionnent comme souhaité, ce qui peut s'avérer chronophage au vu de la taille et du nombre de fonctionnalités que propose l'application MesAvantages.

## Mise en place et organisation de l'instance PostgreSQL définitive

La migration étant à présent possible, il convient de l'effectuer vers une instance PostgreSQL de production.

Afin de réduire les coûts liés aux hébergements chez IBM, le groupe DIDA avait prévu de rassembler l'ensemble des projets actuellement en production sur une seule et même instance Postgres, chaque projet se voyant attribuer une base de données dédiée.
Cette solution a pour avantage majeur de réduire drastiquement les dépenses du groupe, cependant, la mise en œuvre de cette solution implique une sécurité bien moindre, car les bases de données de chaque projet (dont la majorité comportent des données personnelles) vont être réunies en un seul et même endroit.
Afin de garantir une sécurité suffisante au sein de l'instance Postgres, des mesures d'isolation doivent être effectuées.

L'ensemble des projets étant réunis au même endroit, il convient de délimiter les projets, de les isoler le plus possible.

Le principe de schéma est une première étape dans l'isolation des données. Les bases de données sont séparées en plusieurs espaces, chaque utilisateur pouvant se voir attribuer des droits différents sur chacun. C'est pourquoi un espace public peut être accessible à tous, mais qu'un autre schéma, contenant des données plus sensibles, n'autoriseras qu'un droit de lecture, et cela à une poignée d'utilisateurs.

Il est donc nécessaire de placer le contenu de chaque projet dans un schéma dédié. Les permissions seront gérées en temps voulu. Cela est déjà effectué lors de la migration via PgLoader, comme mentionné plus haut.

PostgreSQL introduit également le concept de « Tablespaces ». Les tablespaces sont des « objets serveurs », à savoir des objets présents sur l'ensemble de l'instance et non sur une seule base de données.
Les tablespaces ont pour rôle d'optimiser les performances en plaçant des tables ou index précis sur un disque plus ou moins rapide. Dans le cas de MesAvantages, certaines indexes étant très sollicités, un tablespace dédié fût mis en place, afin que les performances soient optimales. Cependant, il est préférable de garder les tables servant d'archivage ou ayant peu de lecture et de modifications en dehors des tablespaces pointant vers un disque rapide.

La connexion, l'accès et la modification du contenu d'une instance (comme les schémas ou les tablespaces mentionnés auparavant) est réalisée au travers d'un compte utilisateur, aussi appelé « rôle » dans le cas de Postgres.
Chaque utilisateur possède un ensemble de permissions, généralement définies par un « super utilisateur » ayant tous les droits sur l'instance. C'est pourquoi il convient de limiter les permissions des comptes se connectant à l'instance, et de passer le moins souvent possible par ce rôle ayant plein pouvoirs.

## Création d'utilisateurs et gestion des permissions et filtrage IP (Sécurité)

Afin de préserver au mieux le principe d'isolation des bases de données, la mise en place de rôles dédiés à chaque projet est nécessaire.
La méthode envisagée est la suivante : Un rôle sera créé et attribué à un seul et unique schéma d'un projet. Cet utilisateur aura la possibilité de lire, d'insérer, de supprimer et de modifier les données présentes dans le schéma qui lui sera associé.

On débute par la création de l'utilisateur dédié au projet MesAvantages :

```
1   -- MESAVANTAGES
2   CREATE USER mesavantages_user WITH ENCRYPTED PASSWORD 'c4dm3j42c23r0psiystishhxxi8ifmkrj39k4is02n2ryoe' ;
3   ALTER USER mesavantages_user VALID UNTIL 'Feb 10 12:00:00 2022 +1';
```

L'utilisateur recevra également un mot de passe complexe adoptant une forme similaire à celui présent sur la capture d'écran ci-dessus (servant d'illustration). L'initialisation de l'utilisateur et de son mot de passe mentionne un chiffrement du mot de passe.

Une importante partie des données liées aux utilisateurs Postgres sont stockées dans la vue « pg_shadow », qui n'est consultable que par les super utilisateurs. Ce paramètre permet un chiffrement du mot de passe, qui ne pourra ainsi apparaitre en clair lors de l'utilisation de cette vue. Ce paramètre est utile dans le cas où l'instance Postgres aborderait une version relativement ancienne, ou le paramètre « UNENCRYPTED PASSWORD » pouvait encore être appliqué.

La seconde ligne fait état d'une durée de validation du mot de passe de l'utilisateur. Le mot de passe est par défaut valide indéfiniment, or un mot de passe fixe et statique représente un risque dans la sécurité de l'instance. Il est recommandé d'appliquer un délai de conservation aux mots de passe, afin que celui-ci soit régulièrement renouvelé. Le délai est ici d'un mois avant l'expiration.

```
4   GRANT USAGE ON SCHEMA mesavantages_laravel TO mesavantages_user;
5   ALTER USER mesavantages_user SET SEARCH_PATH = public,mesavantages_laravel;
6   GRANT SELECT, UPDATE, INSERT, DELETE, REFERENCES ON ALL TABLES IN SCHEMA mesavantages_laravel TO mesavantages_user;
```

L'utilisation et l'accès à un schéma pour un utilisateur se définit grâce au « Search path », qui répertorie l'ensemble des schémas ciblés. Afin que l'utilisateur ai directement accès au schéma dédié et qu'il n'ait pas à modifier quoi que ce soit manuellement, on le définit dans son search path personnel, accompagné du schéma public qui est utilisable par tous.

Les lignes 5 et 6 servent à attribuer les permissions mentionnées auparavant, mais uniquement au schéma de MesAvantages. Les droits de création et de modification de la structure et cependant exclus. Le paramètre « REFERENCES » permet une utilisation des clefs étrangères, ce qui est primordial à l'application.

Il est important de noter que les permissions non-mentionnées sont par défaut refusées à l'utilisateur. Une bonne pratique consiste également à révoquer l'ensemble des droits sur l'ensemble de l'instance au rôle une fois celui-ci créé, afin de partir d'un modèle vierge.

Les « migrations Laravel » sont couramment utilisées en phase de développement. Ce principe de migration utilise des fichiers PHP, afin d'établir l'ensemble de la structure de la base de données. De la même manière que pour les requêtes SQL, la mise en place de la structure s'effectue via des fonctions prédéfinies par Laravel.

Exemple de mise en place pour la table des utilisateurs :

```php
Schema::create($this->tableName, function (Blueprint $table) {
    $table->engine = 'InnoDB';
    $table->bigIncrements( column: 'id');
    $table->string( column: 'name');
    $table->string( column: 'email');
    $table->string( column: 'password');
    $table->bigInteger( column: 'role_id')->unsigned();
    $table->bigInteger( column: 'group_id')->nullable()->unsigned();
    $table->timestamp( column: 'email_verified_at')->nullable();
    $table->string( column: 'remember_token')->nullable();

    $table->index(["group_id"], name: 'fk_users_groups1_idx');

    $table->index(["role_id"], name: 'fk_users_roles1_idx');

    $table->unique(["id"], name: 'id_UNIQUE');

    $table->unique(["email"], name: 'email_UNIQUE');
    $table->softDeletes();
    $table->nullableTimestamps();
});


Schema::table($this->tableName, function (Blueprint $table) {

    $table->foreign( columns: 'role_id', name: 'fk_users_roles1_idx')
            ->references( columns: 'id')->on( table: 'roles')
            ->onDelete( action: 'restrict')
            ->onUpdate( action: 'cascade');

    $table->foreign( columns: 'group_id', name: 'fk_users_groups1_idx')
            ->references( columns: 'id')->on( table: 'groups')
            ->onDelete( action: 'set null')
            ->onUpdate( action: 'cascade');
});
```

Chaque ligne permet la création d'un champ précis, on y précise d'abord le type (« String » pour une chaîne de caractère, ou encore « bigInteger » pour un nombre), puis le nom de la colonne. Des paramètres additionnels peuvent être précisés si cela est nécessaire. Par exemple, le paramètre « nullable » permet de préciser que les valeurs pour ce champ peuvent être nulles.

La seconde partie du code permet la mise en place des clefs étrangères, servant à faire un lien entre deux identifiants de deux tables différentes. On y précise le nom de la clef, ainsi que les colonne et la table visée par cette contrainte.

L'avantage de cette méthode est que la base de données peut être vidée, détruite puis reconstruite en un bref instant, via une simple commande. Cependant, ce processus implique d'avoir les droits permettant la destruction, la création et l'altération de la **structure** du contenu de ce schéma.

Afin de permettre à l'utilisateur de manipuler la structure des tables, la première ligne doit être revue :

```
GRANT USAGE, CREATE ON SCHEMA mesavantages_laravel TO mesavantages_user;
```

La mention « CREATE » y est ajoutée. Cette mention doit être impérativement appliquée au schéma, et non à la base de données tout entière, afin de ne pas permettre à l'utilisateur de créer ses propres schémas au sein de la base de données.

```
GRANT USAGE, SELECT ON ALL SEQUENCES IN SCHEMA mesavantages_laravel TO mesavantages_user;
```

La dernière partie consiste à donner un droit complet sur les séquences à l'utilisateur.

Cet utilisateur possède actuellement des droits adéquats à l'utilisation qui en sera faite. Cette étape de paramétrage des permissions reste importante, afin de ne pas instancier un utilisateur ayant des droits trop importants et inutiles à sa mission (la possibilité de créer d'autres utilisateurs, ou de mettre en place et supprimer des schémas par exemple).

L'utilisateur peut à présent être utilisé par l'application MesAvantages.

L'application allant être déployée sur un seul « Cluster », il convient de limiter le nombre d'adresses IP pouvant se connecter à l'instance Postgres. Cette mesure augmente davantage la sécurité de l'instance et des données qu'elle contient, le cluster étant la seule entité pouvant accéder au contenu.

Le filtrage est régulé par le fichier « pg_hba.conf », un fichier de configuration ayant pour rôle de contrôler et établir certaines restrictions concernant l'authentification.

```
# Mesavantages Cluster
host    mesavantages    mesavantages_user   10.8.14.0/24       md5
host    all             admin               10.8.14.0/24       md5
```

Il est possible de passer à ce fichier les connexions que nous souhaitons rendre possible ou non. Chaque ligne du fichier est dédiée à un paramètre de connexion particulier. Une ligne se décompose en 5 paramètres séparés, à savoir :
Le type de connexion, la base de données ciblée, l'utilisateur ciblé, l'adresse IP, et enfin la méthode de connexion.

Le paramètre « host » indique qu'on souhaite effectuer une connexion en passant par le protocole TCP/IP, contrairement au paramètre « local ».
Les deux paramètres suivants permettent de définir l'utilisateur et les bases de données visées. L'utilisateur mesavantages_user, depuis cette adresse, ne pourra avoir accès qu'a la base de données « mesavantages ». Cependant, l'utilisateur admin lui aura accès à l'ensemble des bases de données (« all »).
Pour finir, le paramètre « md5 » correspond à la méthode d'authentification classique nécessitant un mot de passe. D'autres méthodes sont spécifiables, comme « trust », qui permet à une adresse de se connecter sans mot de passe, ou encore « reject », qui bloque toute connexion depuis l'adresse, ce qui est utilisé afin d'exclure certaines adresses et d'établir des « blacklists ». Une « whitelist » est le terme qui décrit le mieux notre processus de filtrage, on interdit toute connexion sauf celle depuis le cluster, contrairement à une blacklist qui autorise tout sauf certains cas précis.

Cette méthode est relativement pratique pour rajouter au fur et à mesure de nouvelles adresses, et de gérer dans chaque cas les droits de connexions qui lui sont alloués.

Dans notre cas, le fait de passer par une authentification md5, et de restreindre l'utilisateur mesavantages à sa table permet de garantir une isolation optimale lors de toute connexion à l'instance. Une connexion à l'instance est donc réalisable uniquement depuis le cluster dont l'adresse IP est précisée.

## Documentation du processus

Une documentation doit être rédigée. Elle devra expliquer en détail l'ensemble du processus de migration afin que cette opération puisse être reproduite dans des temps futurs sans perte de temps.

La documentation sera placée dans un espace de stockage dédié aux documentations BNP Paribas IT, et comportera un tableau attestant de l'évolution de cette dernière au fil du temps, la documentation pouvant évoluer au fur et à mesure de l'avancement du projet.

| Objectif du document |
|---|
| DOCUMENTATION : MIGRATION DE MYSQL VERS POSTGRESQL VIA PGLOADER POUR L'APPLICATION « MES AVANTAGES » |

| Document réalisé | |
|---|---|
| Par | Alexandre GIBOZ |
| Le | 06/01/2022 |

| Statut du document | | | Début | Fin |
|---|---|---|---|---|
| Statut | V1 | | 05/01/2022 | 06/01/2022 |

| Date | Prénom NOM | Description |
|---|---|---|
| 05/01/2022 | Alexandre GIBOZ | Création du document |
| 06/01/2022 | Alexandre GIBOZ | Fin de la rédaction de la V1 |
| 12/01/2022 | Virginie PORQUEZ | Consignes pour migration via environnement local |

Validation du document

| Prénom / Nom | Poste | Validé le |
|---|---|---|
| Maximilien RUFIN | Project Owner | 13/01/2022 |
| | | |

# Mise en place du Stockage (COS)

## Contexte

Les mini-sites, leurs produits, ainsi que les thématiques, nécessitent une ou plusieurs images afin de les illustrer au mieux et d'améliorer l'expérience utilisateur. Il est donc nécessaire de relier le projet à un espace de stockage définit, et de permettre aux Administrateurs de gérer l'ensemble des images présentes dans la galerie de manière simple et compréhensible. Il fût décidé qu'un espace cloud serait plus adapté pour la sauvegarde des images qu'un enregistrement local directement dans les fichiers du projet.

La solution retenue est de passer par un **C**loud **O**bject **S**torage (**COS**) de chez IBM. Cette solution a pour avantage de faciliter l'accès aux images, et de limiter les restrictions liées à l'espace de stockage, un stockage distant possédant un espace bien moins restrictif. Le COS IBM est très identique au « **S**imple **S**torage **S**ervice » (**S3**) que propose Amazon.

## Intégration du COS au projet Laravel

Les architectes supervisant la conception de l'application requirent l'utilisation d'une librairie créée par BNP Paribas, permettant de relier un stockage IBM à un projet Laravel.
La mise en place de cette librairie passe par un fichier « Composer », qui permet d'avoir une mainmise sur l'ensemble des dépendances prenant part au projet.

Une fois la librairie intégrée au projet, il suffit d'intégrer et de remplir les valeurs adéquates au sein du fichier d'environnement (.env). Les valeurs apportées sont appelées des « credentials », et se décomposent en cinq (05) valeurs distinctes :

- **La clef d'accès :** La clef publique est attribuée à un utilisateur. Cette technique de connexion permet donc l'accès à l'espace de stockage en héritant des droits d'un utilisateur.

- **La clef privée :** De pair avec la clef d'accès mentionnée ci-dessus. La méthode de connexion peut s'apparenter à celle d'un « jeton ».

- **Le bucket :** Le bucket est un élément obligatoire dans la mise en place d'un S3/COS, et servira de réceptacle aux données qu'on passera au serveur.
  Le nom du bucket doit être unique et uniquement en minuscules, exactement comme pour un DNS.
  Le bucket permet également l'accès aux données stockées via un lien, c'est pourquoi le bucket doit être conforme aux normes appliquées aux DNS.

- **La région :** Une zone géographique ou le bucket va résider doit être choisie. L'application MesAvantages ciblant principalement une population francophone et Européenne, le bucket sera conservé en Allemagne (eu-de) afin de limiter au maximum le délai des requêtes.

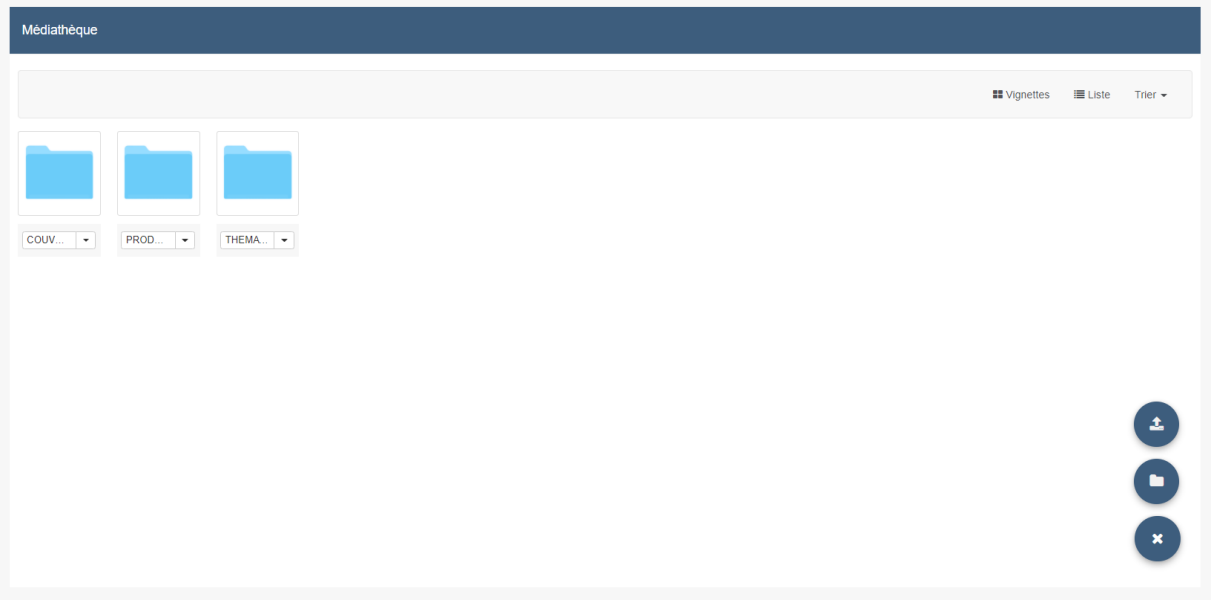
- **L'endpoint :** L'endpoint est un concept qu'on retrouve également pour les « API ». Cela permet d'établir un canal de communication entre le service utilisé et le projet

```
# IBM S3
IBM_COS_ACCESS_KEY_ID="8de2382e7y69481d8fg72ce23a342f10"
IBM_COS_SECRET_ACCESS_KEY="zcbx921bf4b0902d1a9b4e817280a7028848cfe38admf471"
IBM_COS_DEFAULT_REGION="eu-de"
IBM_COS_BUCKET="bnp-mesavantages-staging"
IBM_COS_ENDPOINT="https://s3.eu-de.cloud-object-storage.appdomain.cloud"
```

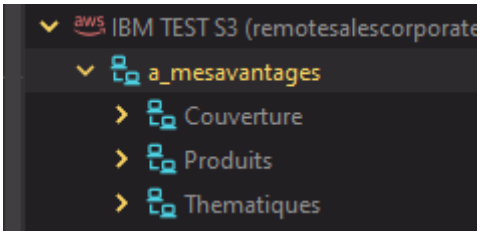
Voici un exemple de valeurs que peut prendre un COS IBM. Les valeurs attribuées ci-dessus sont erronées et ne servent qu'à illustrer la méthode.

Une fois le COS relié au projet, il convient de modifier les fichiers de configurations afin que l'espace de stockage utilisé ne soit plus l'espace local, mais l'espace à distance.

Les fichiers sont désormais accessibles depuis l'espace Médiathèque de l'application, et les utilisateurs peuvent y interagir simplement :




Les fichiers correspondent bien au contenu du COS :



# Déploiement du projet

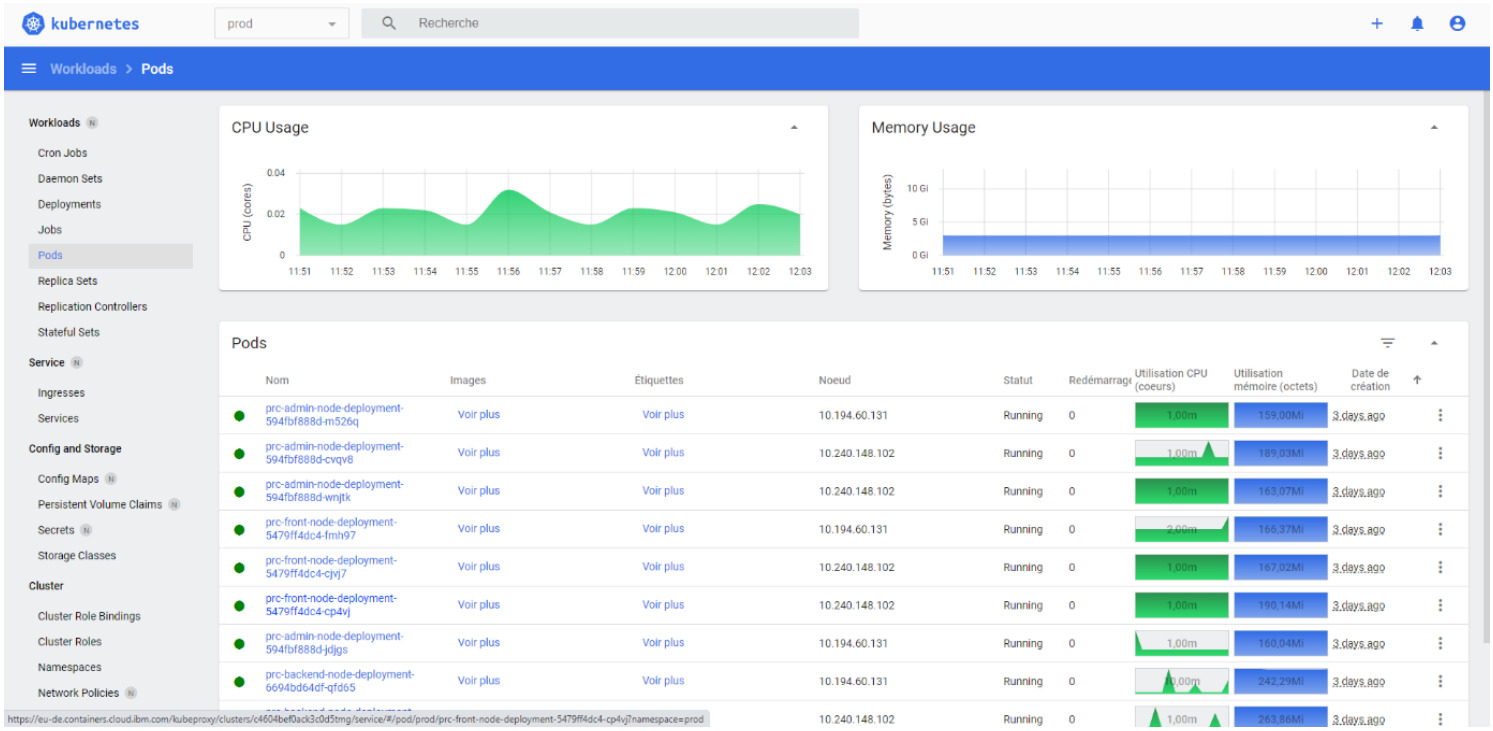
## Contexte

L'ensemble des projets du groupe DIDA, qu'ils soient en cours de développement ou en production, devront être déployés via Kubernetes, qui permet une automatisation de tâches et une mise à niveau rapide via de nombreux outils accessibles, modernes et documentés.
Cette activité consiste donc à adapter l'application MesAvantages à Kubernetes, et d'y implanter la configuration adéquate.

## Introduction à « Kubernetes »

Kubernetes est un orchestrateur de conteneurs, permettant le déploiement d'applications à grande échelle en en permettant la mise en place sur des grappes de serveurs (clusters).

Le principal avantage de Kubernetes est de pouvoir changer l'amplitude de déploiement d'un projet, et ce même si ce dernier est composé de nombreux services.
Contrairement à une infrastructure plus standard et utilisant des Machines Virtuelles simples, Kubernetes peut prendre à sa charge une partie importante des tâches relatives au maintien des conteneurs. Kubernetes permet ainsi d'automatiser le processus de remise en service d'un conteneur défaillant, mais également de gérer l'approvisionnement des ressources pour chacun des conteneurs.



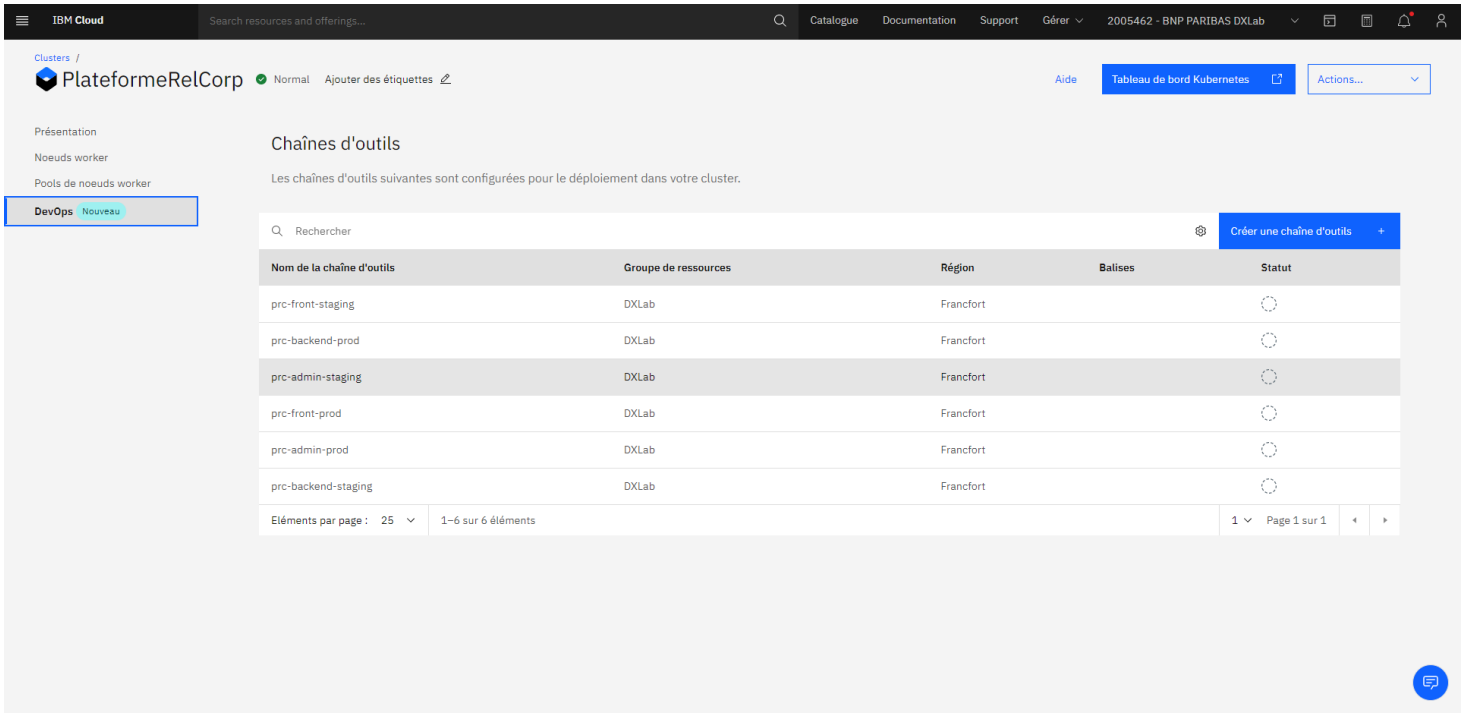
Chaque conteneur Docker est stocké sur un « Pod », et est démultiplié autant de fois que nécessaire afin que l'application puisse gérer la forte demande. Un suivi de l'activité de chaque pod, ainsi que de sa consommation de ressources nous est disponible, afin que l'on puisse réguler le déploiement de l'application le mieux possible.
Ces pods, contenant une instance de l'application, sont d'une part conçus pour être relativement éphémères, mais également pour travailler dans un groupe de Pods accomplissant la même tâche, des répliques (replicas) d'eux-mêmes. C'est ce qu'on constate sur l'image ci-dessus, ou les pods sont répartis sur plusieurs « nœud », à savoir des machines ouvrières contenant les Pods. Les Pods ont également une date de création relativement récente, ce qui atteste d'un cycle de vie bref.

La capacité qu'a Kubernetes à utiliser les conteneurs Docker et à les déployer à grande échelle tout en permettant une gestion complète des ressources allouées à chacun fait de cet orchestrateur la solution idéale aux besoins de MesAvantages.

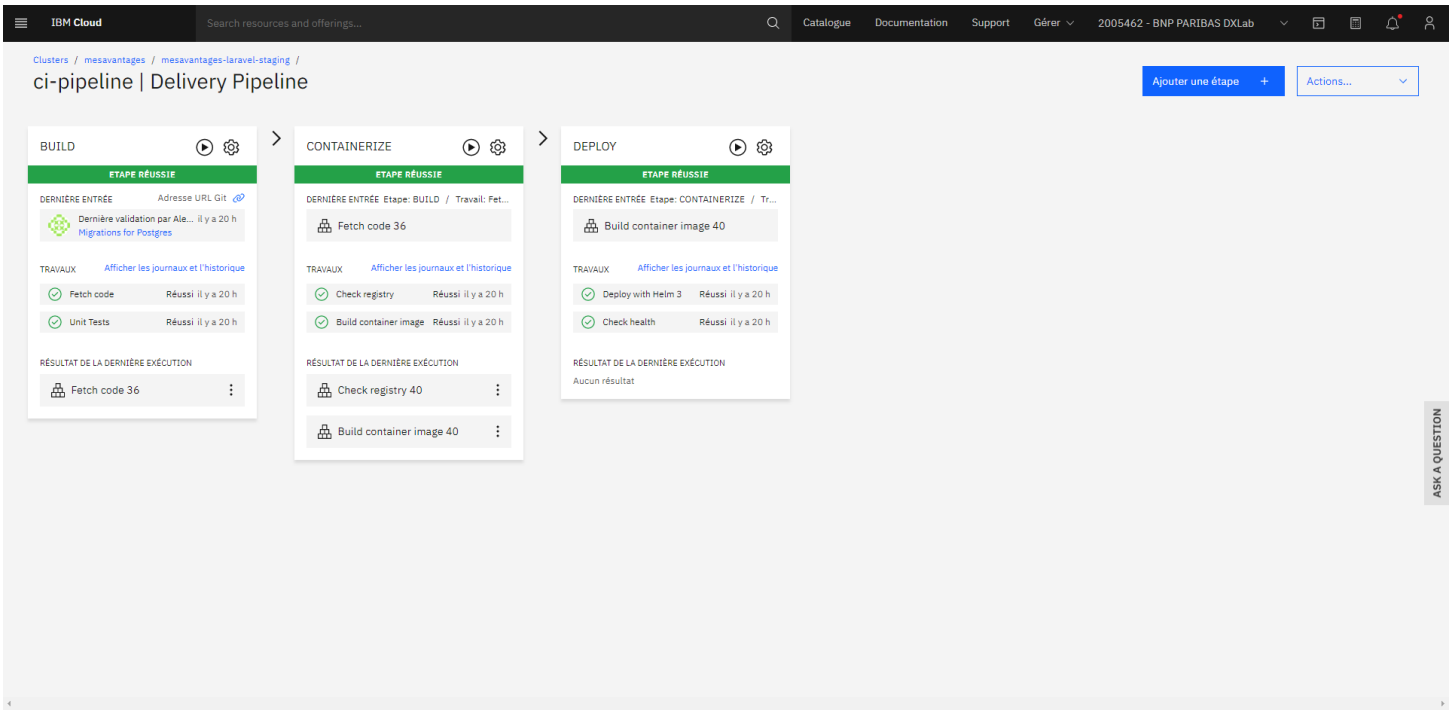
## Configuration et Mise en place du Cluster Kubernetes

Le cluster Kubernetes est directement lié à la branche ibm-migrate du répertoire GitLab MesAvantages. Si une nouvelle modification est détectée, un « build » sera effectué, et l'application déployée sera celle avec le nouveau contenu.

Cela permet d'avoir un aperçu de l'application le plus fidèle possible au résultat final, et de constater si de nouveaux problèmes peuvent apparaître avant d'établir le cluster définitif.
En effet, BNP Paribas imposant de nombreuses normes, notamment liées à la cybersécurité de ses applications, MesAvantages devra passer par une « Toolchain » lors de son déploiement, contenant principalement une étape de « Runtime » faisant passer le projet par de nombreuses API BNP Paribas, ainsi qu'une étape de débogage et de vérification du programme qui est passé au cluster depuis GitLab.



L'ensemble du processus de mise en place du projet est détaillé dans la « Delivery Pipeline », et se décompose en 3 étapes :




La première étape consiste à récupérer l'application depuis la branche ibm-migrate sur GitLab dès qu'un changement est détecté.
La seconde étape consiste à créer l'ensemble des conteneurs requis afin de garantir le bon fonctionnement de l'application. La configuration des conteneurs pour MesAvantages est relativement similaire à celle vue au préalable dans la partie dédiée à Docker.
La troisième et dernière étape consiste à déployer l'application à présent conteneurisée. Cette étape comprend donc la Toolchain mentionnée auparavant, ainsi que l'ensemble des fichiers liés à Kubernetes auquel j'ai eu l'occasion de contribuer.

La mise en place du cluster et de ses composants s'effectue au travers de fichiers « yaml », abordant une syntaxe et une méthode très semblable à celle utilisée pour les fichiers docker-compose.
Un nombre important de ces fichiers sont créés afin de garantir un bon fonctionnement de l'application, et une stabilité du cluster.

Un premier fichier contiendra l'ensemble des paramètres dont aura besoin le cluster afin de se connecter à l'instance, ou encore de se synchroniser avec les paramètres Laravel.

```
envVars:
  APP_NAME: mesavantages
  APP_ENV: production
  # APP_KEY: SECRET
  APP_DEBUG: false
  FORCE_HTTPS: true
  APP_URL: https://mesavantages.bnpparibas/

  DB_CONNECTION: pgsql
  DB_HOST: a0688cb2-a806-4a71-bd67-bkabkakgbeok.zedazfdzerfc.private.db.appdomain.cloud
  DB_PORT: 32667
  DB_DATABASE: mesavantages
  # DB_USERNAME: SECRET
  # DB_PASSWORD: SECRET

  BROADCAST_DRIVER: log
  CACHE_DRIVER: array
  QUEUE_CONNECTION: sync
  SESSION_DRIVER: cookie
  SESSION_LIFETIME: 120
```

Le fichier est directement lié au fichier d'environnement, ce qui permet de récupérer les mêmes clefs, et d'y indiquer les valeurs souhaitées. On y ajoute cependant un protocole SMTP, afin de gérer des envois de mails :

```
MAIL_DRIVER: smtp
MAIL_HOST: smtp.sendgrid.net
MAIL_PORT: 587
MAIL_USERNAME: apikey
# TODO: SECRET
# MAIL_PASSWORD: SECRET
# MAIL_ENCRYPTION: tls
# MAIL_FROM_NAME: "Jimini'Budget"
# MAIL_FROM_ADDRESS: noreply@dxlab.fr
```

On précise le port utilisé, ainsi que le mail exploité afin de procéder à l'envoi de mails. On passera par le protocole **T**ransport **L**ayer **S**ecurity (**TLS**) afin de garantir une communication sécurisée.

De nombreux autres fichiers, adoptant une forme similaire, permettent la mise en place des pods et nœuds afin de mettre le cluster Kubernetes sur pied.

## Problèmes rencontrés

Il fût décidé par les architectes, au cours de ce transfert vers Kubernetes, que l'application n'utiliserait plus une configuration Apache mais Nginx, un serveur web gagnant en popularité.

J'ai donc effectué la nouvelle configuration, qui se compose comme ceci :

nginx-config.yaml

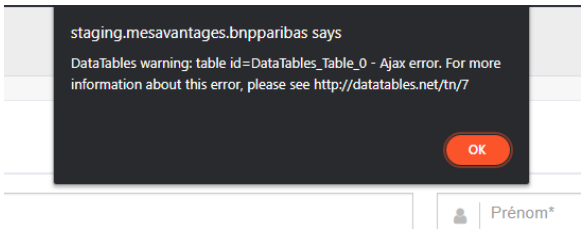
```yaml
kind: ConfigMap
apiVersion: v1
metadata:
  name: nginx-config-map
data:
  nginx.conf: |
    user www-data;
    worker_processes auto;
    pid /run/nginx.pid;

    events {}

    http {
      sendfile on;
      tcp_nopush on;
      tcp_nodelay on;
      keepalive_timeout 65;
      types_hash_max_size 2048;
      client_max_body_size 50M;
      client_header_buffer_size 5120k;
      large_client_header_buffers 16 5120k;
      include /etc/nginx/mime.types;
      default_type application/octet-stream;
      gzip on;
      gzip_vary on;
      gzip_proxied any;
      gzip_comp_level 6;
      gzip_buffers 16 8k;
      gzip_min_length 256;
      gzip_http_version 1.1;
```

Un fichier de configuration Nginx est grossièrement une suite d'instructions ou l'on attribue à chaque paramètre une valeur qu'Nginx devra respecter.
Cette opération peut se révéler relativement laborieuse, étant donné le vaste nombre de paramètres à prendre en compte. L'image n'est qu'un court extrait de cette configuration.

Une fois la configuration Apache entièrement remplacée par celle-ci, une seule erreur survient. Cette erreur est dû aux requêtes des tableaux Ajax, se révélant trop longue. Cela prit un long moment avant de trouver que ceci en était la cause, étant donné les messages d'erreurs relativement classiques obtenus



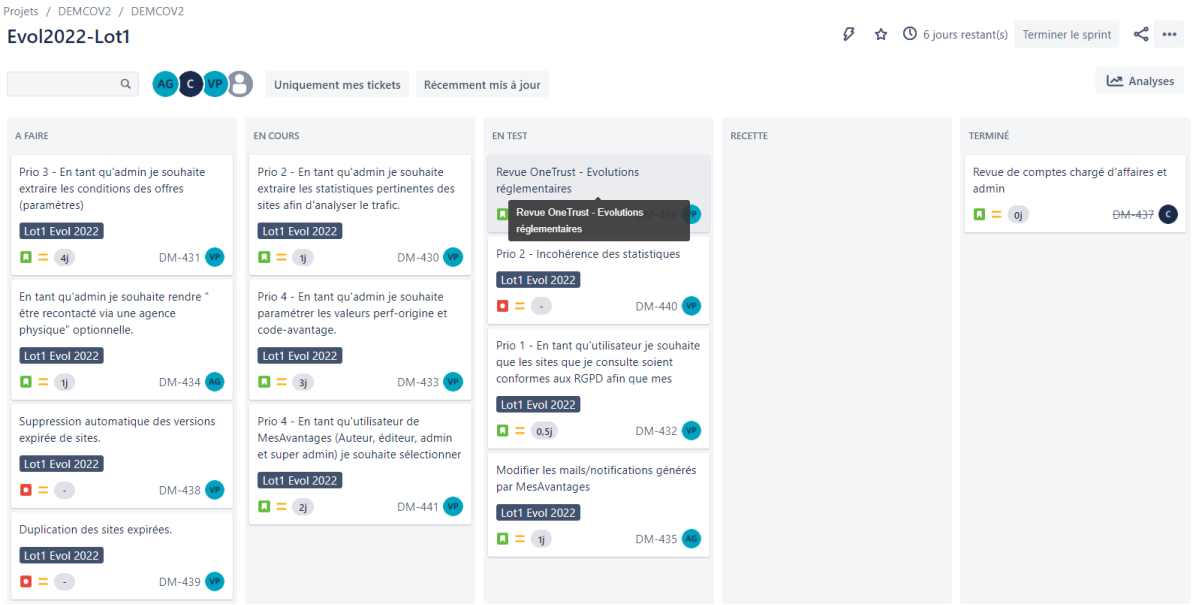

Afin de palier à cette erreur, il convient d'augmenter la taille maximale des requêtes. On apporte la modification suivante au fichier de configuration Nginx pour les deux valeurs (en orange) :

```
types_hash_max_size 2048;
client_max_body_size 150M;
client_header_buffer_size 10240k;
large_client_header_buffers 16 5120k;
```

## Tickets Jira

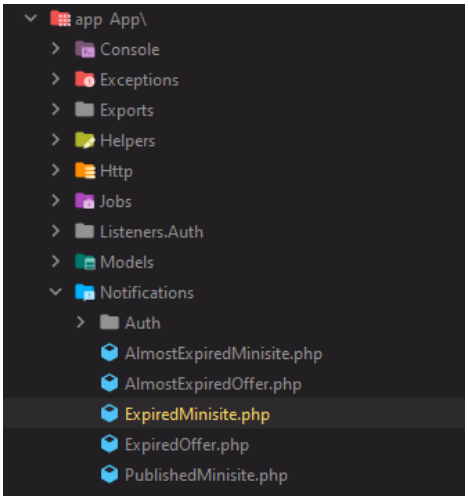
Les tickets Jira sont des évènements devant subir une modification. Ce service est souvent utilisé afin de mettre en lumière certains problèmes mineurs pouvant survenir au cours du développement d'une application, mais également pour y incorporer de nouvelles fonctionnalités.
Chaque ticket est attribué à une personne, qui aura pour mission de clore le ticket.

L'espace dédié aux tickets est compartimenté en étapes d'évolution, cela afin que le supérieur hiérarchique et les autres membres de l'équipe puissent suivre l'avancement global des tâches.




### Mise à jour des notifications et mails envoyés par MesAvantages

Comme évoqué précédemment lors de la phase de mise en place du cluster Kubernetes, un service d'envoi de courriel a été intégré au projet.
L'objectif de ce ticket est le suivant : L'auteur et l'auteur associé d'un minisite doivent recevoir des mails les prévenant de la fin de validité d'un de leur site, à des intervalles précis, et ce afin qu'ils puissent entrer en contact avec le partenaire afin de renouveler ou non le site. L'objectif est donc de mettre ce système en place.




Laravel intègre un espace dédié à la prise en charge des mails au projet : les « notifications »

Le fichier « AlmostExpiredMinisite.php » prendra en charge l'ensemble des mails relatifs aux sites allant expirer. Le système d'envoi de mails concernant les site déjà expirés est déjà en place.

La fonction « toMail() » va instancier un nouvel objet appelé « MailMessage ». On passe ensuite à cet objet l'ensemble du contenu souhaité :

```
 */
public function toMail($notifiable) {
    $mailMessage = new MailMessage();
    $notification = (object) $this->toArray($notifiable);

    $mailMessage->subject( subject: '[' . config( key: 'app.name') . '] ' . $notification->subject);
    $mailMessage->greeting( greeting: 'Bonjour ' . $notifiable->fullName() . ',');
    $mailMessage->line( line: 'Vous recevez cet email car l\'offre "' . $this->offer->name . '" sur le site dédié "' . $this->offer->minisite->name  . '" est échue le '
        . $this->offer->ends_at->format( format: 'd/m/Y') . '.');
    $mailMessage->line( line: 'Vous pouvez l'éditer en cliquant sur le bouton ci-dessous :');
    $mailMessage->action( text: 'Éditer le site dédié', $notification->link);
    $mailMessage->line( line: 'Si vous disposez d'Internet Explorer, merci de copier coller ce lien dans Firefox, disponible sur tous les PC BNP PARIBAS : '
        . $notification->link);

    return $mailMessage;
}
```

L'ensemble des informations nécessaires, à savoir le nom de l'auteur, le nom du minisite et les dates sont à portée de la classe, et sont donc utilisées ici.
La fonction retourne l'objet une fois celui-ci (le contenu du mail) remplit.

Il faut, à présent, gérer la fréquence d'envoi de ces mails. On veut qu'un mail soit envoyé aux auteurs quinze (15), huit (08) et une (01) journée avant l'expiration du site.

On passe les dates auxquelles on souhaite envoyer un mail :

```
$this->almostExpiredMinisites( delay: '1 day');
$this->almostExpiredMinisites( delay: '8 days');
$this->almostExpiredMinisites( delay: '15 days');
```

Il ne reste à présent qu'a mettre en route ce système. Laravel permet de définir des tâches qui vont être automatiquement effectuées à des heures précises, et ce dans l'espace dédié à la « Console ».

```
protected function schedule(Schedule $schedule) {
    $schedule->command( command: 'minisites:check')
             ->dailyAt( time: '00:00');
//             ->dailyAt('00:00')->emailOutputTo(explode(',', config('app.debug_emails')));

}
```

La fonction « schedule », prenant comme paramètre une variable prédéfinie par Laravel lui permettant de comprendre le but de la fonction, permet de planifier une tâche à une heure précise. La commande « minisite:check » permet de collecter l'ensemble des dates d'expirations des minisites actifs, et de les comparer aux dates d'avertissement par mail qu'on a rempli auparavant.

Cette tâche sera effectuée tous les jours à minuit, comme indiqué à la ligne 3.

# Conclusion

Au cours de ce stage j'ai eu la possibilité de mettre en pratique, et donc de développer de nombreuses compétences acquises au cours de ces deux années de BTS dans un milieu et un contexte professionnel, notamment en ce qui concerne la programmation orientée objet au travers de Laravel.
Cependant, ce stage m'a également permis d'acquérir et de commencer à développer de nouvelles compétences, principalement Docker, Kubernetes et Nginx, ce qui me sera d'une grande utilité dans le futur.

Le fait d'user de Jira et Git permet de garder une trace de l'évolution du projet, mais surtout de son propre travail, ce qui m'a aidé à m'organiser dans l'ensemble de mes tâches et de gérer mon emploi du temps de la meilleure des façons. Cette méthode d'organisation et de collaboration me permit également d'avoir un aperçu des méthodes utilisées en entreprise.