

DEPARTEMENT INFORMATIQUE - IUT 2 GRENOBLE



Année Universitaire 2019-2020

MEMOIRE DE STAGE

**CREATION D'UNE APPLICATION D'ARCHIVAGE
Gespro**



Présenté par

Naod Bekele

Jury

IUT: Delphine Chollat-Namy

IUT: Jérôme Goulian

Société: Benjamin Perry

TABLE DES MATIERES


I. Introduction	3
II. Contexte	3
II.1. L'entreprise GESPRO	3
II.2. Présentation globale du stage	4
II.3. Environnement de travail	5
II.4. Annonce du plan	5
III. Spécification et attentes	5
III.1. Cahier de charges	5
<i>Fonctionnalités primaires</i>	
<i>Fonctionnalités supplémentaires</i>	
III.2. Analyse de l'existant	8
III.3. Choix technologies	9
IV. Conception	10
IV.1. Déroulement des tâches processus	10
<i>Diagramme BPMN</i>	
IV.2. Interface IHM	12
<i>Maquettes fonctionnelles</i>	
IV.3. Modélisation de la base de données	14
<i>Diagramme UML</i>	
IV.4. Études des risques et éventuels plans de mitigation	16
V. Réalisation	17
V.1. Calendrier de travail	17
<i>Diagramme Gantt</i>	
V.2. Développement backend	18
<i>Synchronisation avec serveur LDAP</i>	
<i>Utilisation des queues Laravel</i>	
<i>Intégration des scripts</i>	
<i>Implémentation du SDK OpenStack</i>	
<i>Téléchargement</i>	
V.3. Développement frontend	25
<i>Templates Blade</i>	
<i>Pages dynamiques avec JavaScript</i>	
V.4. Contraintes et problèmes rencontrés	26
<i>Solutions</i>	
VI. Test	27

Conclusion	28
Glossaire	29
Annexes	30

Déclaration de respect des droits d’auteurs

Par la présente, je déclare être le seul auteur de ce rapport et assure qu’aucune autre ressource que celles indiquées n’ont été utilisées pour la réalisation de ce travail. Tout emprunt (citation ou référence) littéral ou non à des documents publiés ou inédits est référencé comme tel.

Je suis informé qu’en cas de flagrant délit de fraude, les sanctions prévues dans le règlement des études en cas de fraude aux examens par application du décret 92-657 du 13 juillet 1992 peuvent s’appliquer. Elles seront décidées par la commission disciplinaire de l’UGA.

A, Grenoble, Le 09/08/20 

I. Introduction

Ce document présente un rapport de stage suite au projet de fin d’études que j’ai réalisé dans l’entreprise Gespro.

Ce projet a été réalisé du 01/06/2020 au 15/08/2020 et concerne le sujet suivant :

“Développement d’une application de gestion de BACKUPS à destination des équipes techniques et du pôle commercial”.

II. Contexte

II.1. L’entreprise Gespro

Gespro est une entreprise technologique spécialisée dans le domaine de construction. L’entreprise a été créée en 1999 et s’affirme depuis près de 15 ans comme l’un des partenaires technologiques du secteur de construction. Elle propose une plateforme de travail collaboratif en ligne, permettant la gestion intégrale d’un projet de construction (phases administratives, techniques et financières). Elle réunit des entreprises de toutes tailles sur cette plateforme afin que tous les utilisateurs aient accès aux mêmes outils et fonctionnalités.

Elle offre un outil numérique de gestion de projet, de la phase d’études à la phase de réception. Cet outil collaboratif est complètement dématérialisé et accessible depuis tous les navigateurs Web sur mobile ou ordinateur.

L'entreprise a développé deux versions de cette plateforme, avec la deuxième version ayant des fonctionnalités supplémentaires.

L'outil Gespro contient quatre modules :

-CENTRINFO : Gestion documentaire permettant la consultation, le dépôt et le téléchargement des fichiers.

-DIFDOC : Dans la phase de construction, les documents passent par une série de validation. Ce module gère les circuits de validation et d'approbation des documents.

-PROFI : Profi assure une plateforme de gestion financière automatique. Il garantit un suivi en temps réel de l'avancement des demandes d'acomptes et de la maîtrise des délais

-BIMPRO : Système de visualisation et partage des fichiers 3D. Elle met à disposition une fenêtre interactive 3D et garde un historique de tous les changements appliqués.

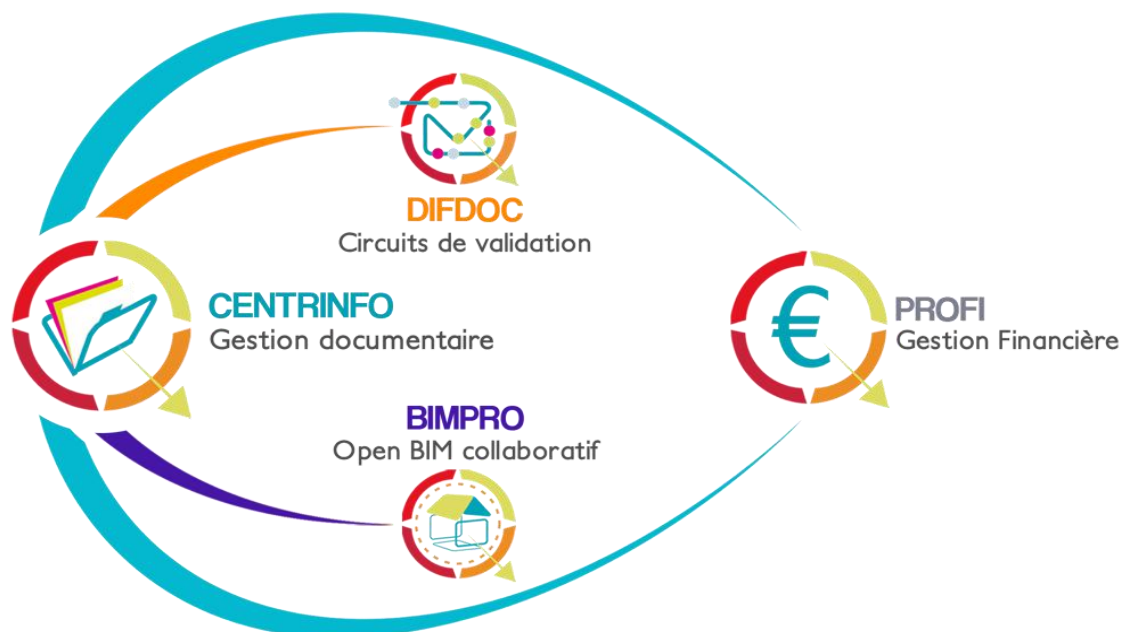


Figure 1 : Modules Gespro

II.2. Présentation globale du stage

Chaque projet est composé de plusieurs documents. Ces documents sont de types différents allant d'un simple PDF à des fichiers 3D. Un projet occupe alors beaucoup d'espace mémoire; en moyenne 9 gigaoctets par projet. Gespro contient sur son serveur quelques centaines de ces projets.

Le problème se pose dans le fait qu'avec le temps qui passe, le nombre de projets terminés augmente aussi. Ces projets occupent alors de plus en plus d'espace mémoire. L'entreprise souhaite alors exporter ces projets chez un hébergeur et ne plus avoir ce problème.

Le projet du stage était donc d'analyser la plateforme actuelle et de trouver puis intégrer une solution d'archivage externe, accessible au long terme.

L'archivage de ces données doit se faire via une application web, accessible que depuis l'entreprise. Cette application doit appliquer une partie frontend et doit utiliser un système d'authentification *LDAP*.

II.3. Environnement de travail

Le projet s'est déroulé au sein de l'entreprise en présentiel du mois de juin au mois de juillet .

J'avais accès à un ordinateur avec un système d'exploitation Windows sur lequel est installé une *VM* Oracle donnant accès à un *OS* Linux Debian, pour simuler l'environnement de production. Une installation d'une *IDE* Visual Studio Code pour le développement et de Termius, une *CLI* qui facilite l'accès à la VM depuis une terminale.

Avec le temps j'ai rajouté des logiciels que je détaille dans la *III.3 Choix technologies*.

J'avais un compte mail entreprise, que j'ai utilisé pour la communication externe pour des sujets en liaison avec mon stage.

II.4. Annonce du plan

Je vais en première partie présenter le cahier de charges et les fonctionnalités attendues. Ensuite je vais décrire l'approche que j'ai utilisé pour l'analyse de l'existant et le choix de la technologie qui a procédé. Dans cette même partie, j'inclurai un bref détail sur les éventuels risques détectés et les plans de mitigations implémentés.

Dans une deuxième partie je présenterai toute la partie conception. Je détaillerai la base de données et les pages prévues pour le frontend et une description sur l'enchaînement des tâches processus pour la procédure d'archivage et pour la récupération des archives. Chaque description est approfondie avec des maquettes et diagrammes détaillés.

La troisième partie est celle de la réalisation du projet. Elle inclut la planification de travail choisie pour l'intégralité du projet. Elle est suivie par deux grandes sous-parties représentant la phase de développement détaillant les solutions intégrées et les obstacles rencontrés.

Enfin, je conclus avec une description approfondie sur le projet, sur ce que j'ai ressenti et les connaissances que j'ai gagnées à travers cette première expérience professionnelle.

III. Spécification et attentes

III.1. Cahier de charges

III.1.1 Fonctionnalités primaires

Données à récupérer :

Les données à récupérer sont les documents de chaque module d'un projet (Centrinfo, Difdoc, Profi et Bimpro). Un script a déjà été écrit avec le langage GO, il récupère ces documents et en forme un ZIP.

Ce script travaille en parallèle avec un deuxième script PHP, localisé sur le serveur principal V1. Il est en charge de la récupération de la base de données d'un projet à archiver.

Il faut récupérer ce fichier .sql et l'ajouter au ZIP principal pour former un dossier complet, prêt à être archiver.

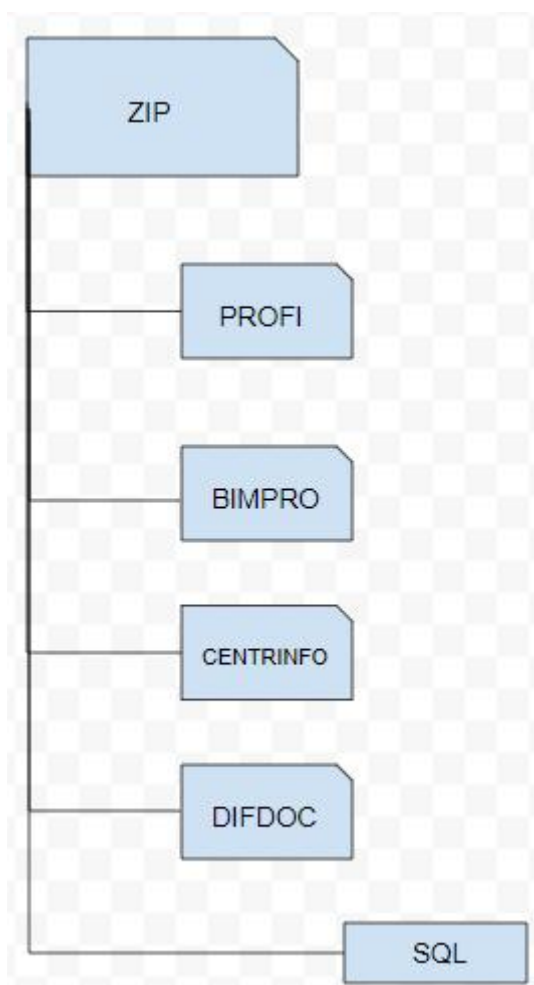


Figure 2 : Arborescence d'un ZIP à archiver pour un projet donné

Authentification :

Pour pouvoir archiver, un employé doit saisir ces identifiants, un compte mail et un mot de passe, il sera redirigé vers la page principale si l'authentification est réussie.

Les applications Web utilisent majoritairement une authentification locale signifiant que lorsqu'un utilisateur s'authentifie, le serveur récupère de la base de données les identifiants correspondants et tente de

les comparer. Dans le cas de ce projet, l'authentification n'utilise pas sa base de données locale mais fait appel à un serveur dédié à l'authentification. Ce serveur appelé un serveur *LDAP* est conçue spécialement pour les entreprises, il permet la gestion des informations des employés au sein des entreprises. Ce serveur est hautement configurable et offre une facilité d'usage par rapport aux bases de données SQL.

Il a fallu donc configurer un serveur *LDAP* et trouver une solution pour intégrer ce type d'authentification au sein de l'application.

Liste des projets :

Tous les projets de Gespro V1, quelque soit leur état, seront affichés sous la forme d'un tableau. Les projets seront récupérés depuis la base de données de Gespro V1. La récupération des projets devra être la seule interaction avec la base de données de Gespro V1.

Il faudra afficher les informations relatives à un projet :

- Nom du projet
- Login du projet
- Modules actifs
- Poids du projet
- Temps depuis l'archivage (si archivé)
- Etat du projet

Etat du projet :

Un projet a un état donné : «Actif», «Bloqué», «A archiver» et «Archivé». Les états «Actif», «Bloqué» et «A archiver» sont des états donnés uniquement depuis Gespro V1. L'état «Archivé» sera donné depuis cette application d'archive.

Il faudra distinguer facilement l'état d'un projet, avec une couleur spécifique. Il faudra distinguer également les projets dont l'archive a été faite il y a plus de 5 ans.

Filtre :

Il faudra pouvoir filtrer la liste des projets par nom, login, état et date d'archivage :

- Nom et login seront filtrables à l'aide d'une zone de saisie de texte

- Etat du projet sera filtrable à l'aide d'une liste déroulante

- La date de début et de fin d'un projet sera filtrable avec une zone de saisie de texte (les dates sont optionnelles, ce qui permet une recherche que sur la date de fin par exemple)

Tri :

Un tri par nom, login, état, poids et temps d'archive doit être implémenté.

Pagination :

L'affichage de 354 projets sur une même page Web n'offre pas une ergonomie visuelle. Il faut donc implémenter la pagination, avec le nombre de projets par page paramétrable.

Actions pour projet :

Il existe principalement deux actions pour les projets : l'archivage et le téléchargement. Lorsqu'un projet passe en état «A archiver», l'utilisateur aura le droit d'archiver le projet. L'archivage complété, l'utilisateur pourra ultérieurement télécharger le projet.

Logs des actions :

Il faudra qu'il soit possible de récupérer tous les événements effectués, contenant les informations suivantes : type d'action (archivage ou téléchargement), personne ayant effectué l'action, le projet en question avec sa taille et la date d'action.

III.1.2 Fonctionnalités supplémentaires

Suppression des archives :

La suppression des archives doit se faire en plusieurs parties et selon les droits utilisateurs. Elle doit être autorisée par un employé ayant les droits administratifs avant qu'elle est mise en action.

Statistiques :

Si toutes les fonctionnalités primaires sont intégrées avant les délais prévus, concevoir une page de statistiques contenant ces informations :

- Volume total des transferts
- Nombre total de projets archivés
- Volume total des transferts
- Nombre total de projets archivés dans le temps
- Taille totale des archives

III.2. Analyse de l'existant

L'étude du marché pour les services d'hébergement était la première partie de l'analyse de l'existant. Il existe plusieurs fournisseurs de stockage tel que Amazon Web Service avec Amazon S3 dédiée au stockage long terme, ou Google Cloud Storage le plus récent dans le marché.

Cependant, Gespro utilise OVH comme fournisseurs de services Web. OVH est une entreprise française offrant des serveurs dédiés pour la computation Web, des noms de domaines pour les sites, des domaines mails et un réseau de Cloud privé entre autres. Leur solution de stockage glaciers présente plusieurs avantages. Pour commencer, il offre un stockage avec des prix intéressants : 0.01€ par gigaoctet transféré et 0.002€ par gigaoctet stocké par mois. De plus, parmi plusieurs dizaines de leurs serveurs de stockage, un se trouve en Europe de l'Ouest, qui réduit les temps de latence de quelques secondes. Il s'agit également d'un point central du réseau OVH.

Un autre avantage pour les serveurs OVH est la durée de stockage. On a la possibilité de stocker des archives pour plusieurs années sans soucis de perte des documents..

Enfin et surtout le bénéfice le plus important de OVH est la taille des projets à archiver. OVH n'a pas de limite pour les documents. Il est possible d'archiver un document faisant plusieurs téraoctets en une action.

En plus de tous les avantages présentés, Gespro est déjà familiarisé avec OVH. Le choix le plus raisonnable pour la recherche d'un hébergeur serait donc cette entreprise.

III.3. Choix technologiques

Gespro étant un fournisseur de services en ligne, utilise Laravel : un framework PHP très puissant constamment évolutif et simplifiant les tâches complexes.,

Développement

- Laravel : Fournit un outil de ligne de commande intégré appelé Artisan. Cet outil aide à créer une architecture code squelette. La gestion de la base de données en devient plus facile. L'un des atouts le plus important venant avec Laravel est la sécurité de haut niveau. Il permet une protection des URL et un abri global sur la base de données, qui est très susceptible aux injections SQL (forme de cybe- attaque).

Les avantages Laravel sont très nombreux pour tous les lister, c'est un framework incontournable dès lorsqu'on développe un grand projet Web.

- MySQL: MySQL est globalement reconnue comme étant la base de données la plus sécurisée. Elle offre aussi un avantage dans sa haute paramétrabilité. Avec un moteur de recherche très rapide et une intégration facile avec les dépendances PHP, c'est devenu le choix du système de gestion de la base de données.

- PHP OpenStack SDK: OVH est basé sur la technologie OpenStack pour sa gestion de services. OpenStack est un ensemble de logiciels open source qui permet de gérer les infrastructures du *cloud computing*, incluant la sauvegarde glacière. Il existe un SDK Openstack en PHP permettant de gérer le réseau du cloud computing. C'est celui qu'on va utiliser pour le développement.

Cet outil est responsable pour toutes les interactions avec la plateforme OVH. Il est utilisé pour effectuer les fonctionnalités primaires d'archivage et de téléchargement entre autres.

- OpenLDAP -Serveur: OpenLDAP est un outil fondé du protocole LDAP. Il permet la gestion des informations utilisateurs. OpenLDAP contient une configuration serveur, c'est ce serveur qui est interrogé par notre application pour l'authentification.

Les entrées LDAP partagent des ressemblances aux entrées SQL, ils sont formés des paires clé-valeur.

```
dn: cn=Naod Bekele, dc=example, dc=org
cn: Naod Bekele
givenName: Naod
sn: Getachew
telephoneNumber: +33 6123456
telephoneNumber: +33 6123456
mail: nbekele@gespro.fr
manager: cn=Naod Bekele, dc=exemple, dc=com
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: person
objectClass: top
```

Figure 3 :Exemple d'entrée LDAP

- **LdapAccountManager**: OpenLDAP est normalement géré depuis une fenêtre Linux, donc en frappant des lignes de commandes. Cela peut devenir fastidieux et long, cet outil permet alors d'avoir une fenêtre graphique pour gérer le serveur LDAP.

The image shows a graphical user interface for managing LDAP users. It features four input fields on the left: 'Prénom' (First Name) with the value 'Naod', 'Nom de famille' (Last Name) with the value 'Bekele', 'Initiales' (Initials) with the value 'N.B', and 'Description' with the value 'Stagiaire'. Each field has a small blue question mark icon to its right. To the right of these fields is a user profile icon (a person with a blue head and a red tie) and a button labeled 'Ajouter une photo'. Two red arrows point from the 'Prénom' and 'Nom de famille' fields towards the user profile icon.

Figure 4 : Gestion serveur LDAP avec LdapManager

- **LdapRecord**: Un paquetage PHP qui permet de travailler avec les serveurs LDAP. Gère les connexions et authentifications mais aussi la recherche des entrées LDAP. Offre une meilleure sécurité et lisibilité du code.

Initialement j'ai rédigé le code d'authentification LDAP qu'en PHP brute, avec ce paquetage j'ai pu le réécrire plus rapidement et avec moins de complexité de code. Par défaut, Laravel a déjà un système d'authentification, ce paquetage s'intègre harmonieusement dessus.

- **JavaScript ES2015**: L'interaction avec notre application sera faite à travers des pages Web. La page Web doit être dynamique, par exemple pour montrer une animation de progression lors de l'archivage. JavaScript permet de réaliser cela, l'entreprise utilise par convention la version ES2015 de Javascript.

- **GitLab**: Outil de version contrôle *Git*, utilisé pour le dépôt de notre code PHP.

IV. Conception

IV.1. Déroulement des tâches processus

Archivage

Le déroulement de la procédure d'archivage se fait en plusieurs parties. L'utilisateur commence par choisir un projet, le projet doit être en état "à archiver" sinon l'archivage est impossible. L'état des projets est déterminé par l'administrateur qui s'occupe de Gespro V1.

Si le projet est en état "à archiver", on exécute le premier script de récupération des données SQL. Ce script prend en paramètre l'id du projet et récupère toutes les données SQL associées à ce projet. Ces données sont récupérées de la base de données de Gespro V1, auquel j'ai pas accès directement. Encore une fois, ce script ne récupère pas les documents d'un projet mais que ses entrées SQL.

Après qu'on a récupéré les données SQL, on exécute le deuxième script qui lui est en charge de récupérer tous les documents liés à un projet (bimpro,profi..). Ce script va placer tous les documents dans un ZIP. On procède par placer le fichier SQL dans ce nouveau ZIP .

Le dossier prêt, on l'envoie sur le serveur Cloud OVH et on notifie l'utilisateur de l'envoi. On met à jour toutes les entrées de la base de données, tel que le changement d'état de "à archiver" vers "archiver" et on supprime le ZIP de notre serveur.

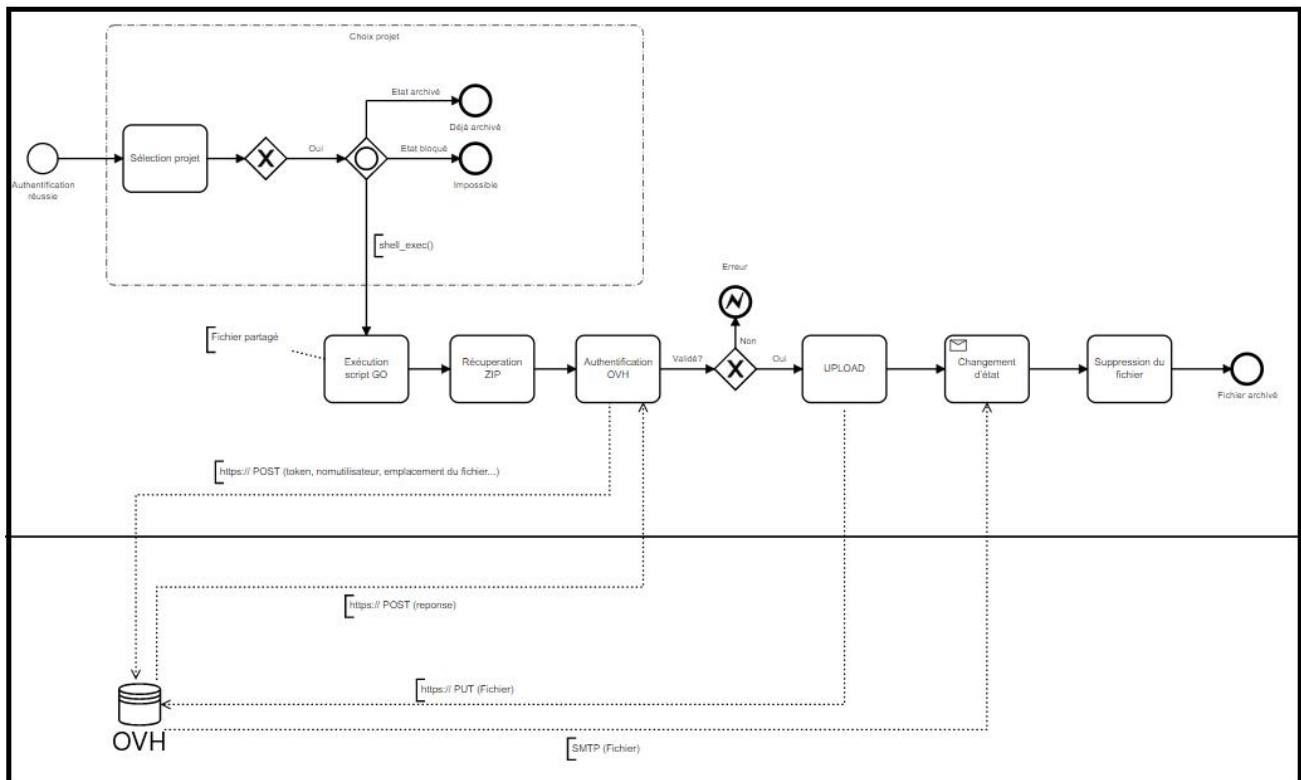


Figure 4 :Diagramme BPMN du processus d'archivage

Téléchargement

Tous les projets stockés sur le cloud OVH peuvent être téléchargés. L'utilisateur choisit le projet et fait une demande de téléchargement. Le téléchargement n'est pas disponible à tout moment, OVH demande un délai de traitement pour récupérer le projet. Une date à laquelle le téléchargement sera disponible est envoyée à l'utilisateur. Une fois cette date atteinte, l'utilisateur peut récupérer le projet dans les 24 heures qui suivent.

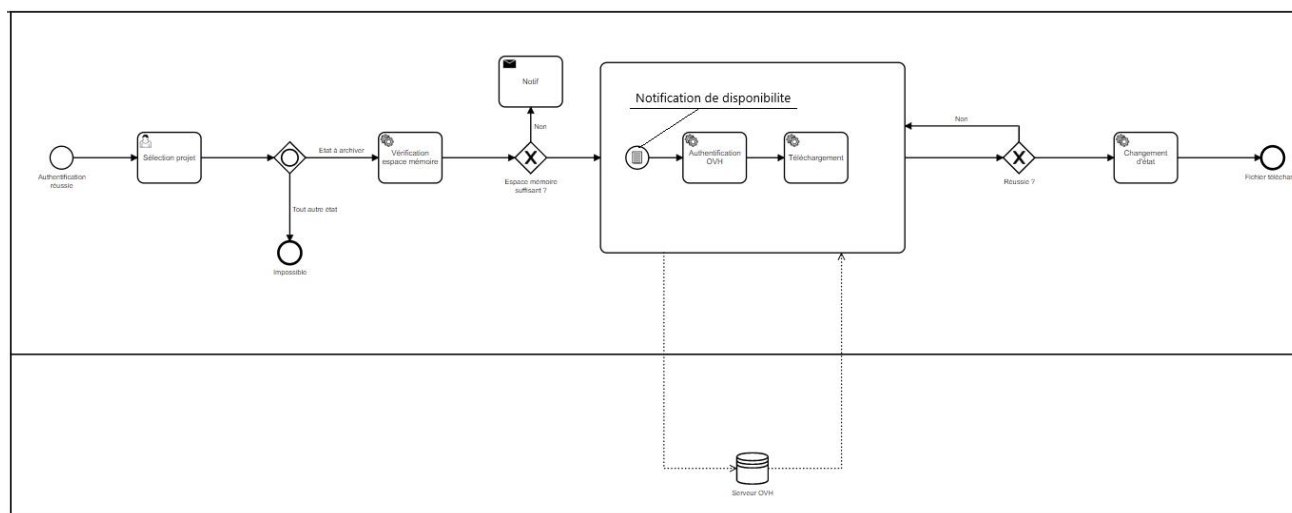


Figure 5 :Diagramme BPMN du processus de téléchargement

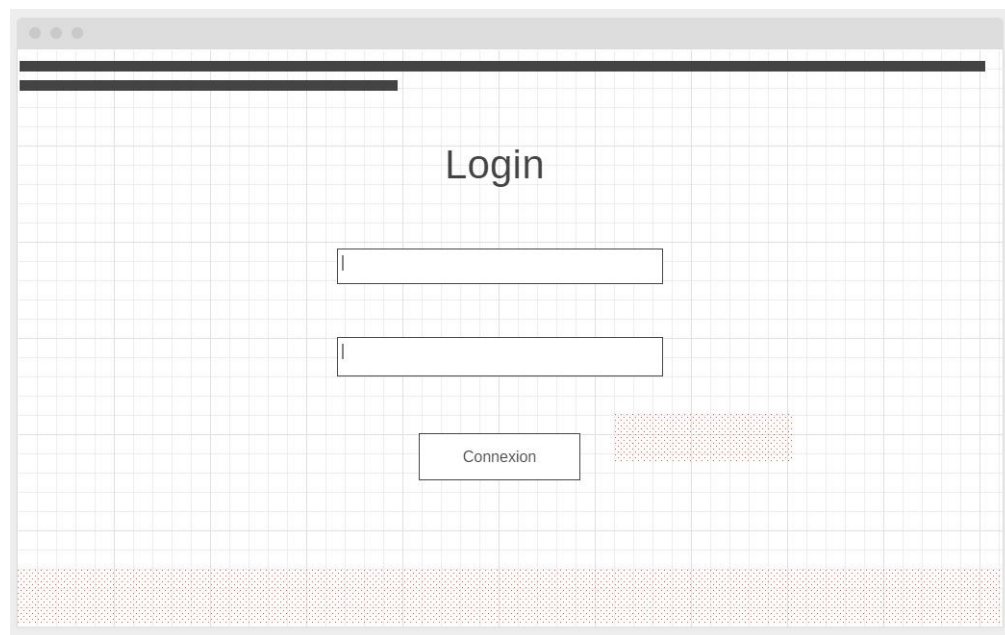
IV.2. Interface IHM

Notre application fonctionne à travers une page Web Intranet (accessible que depuis l'entreprise). Elle doit lister tous les projets de construction. Comme décrit dans le cahier des charges, les projets doivent être filtrables et triables. Il faut donc avoir des boutons permettant de faire cela, la norme est que ces boutons soient positionnés sur la barre de navigation. Pour le filtre, on ajoute une barre de recherche.

La page doit respecter les ergonomies Web :

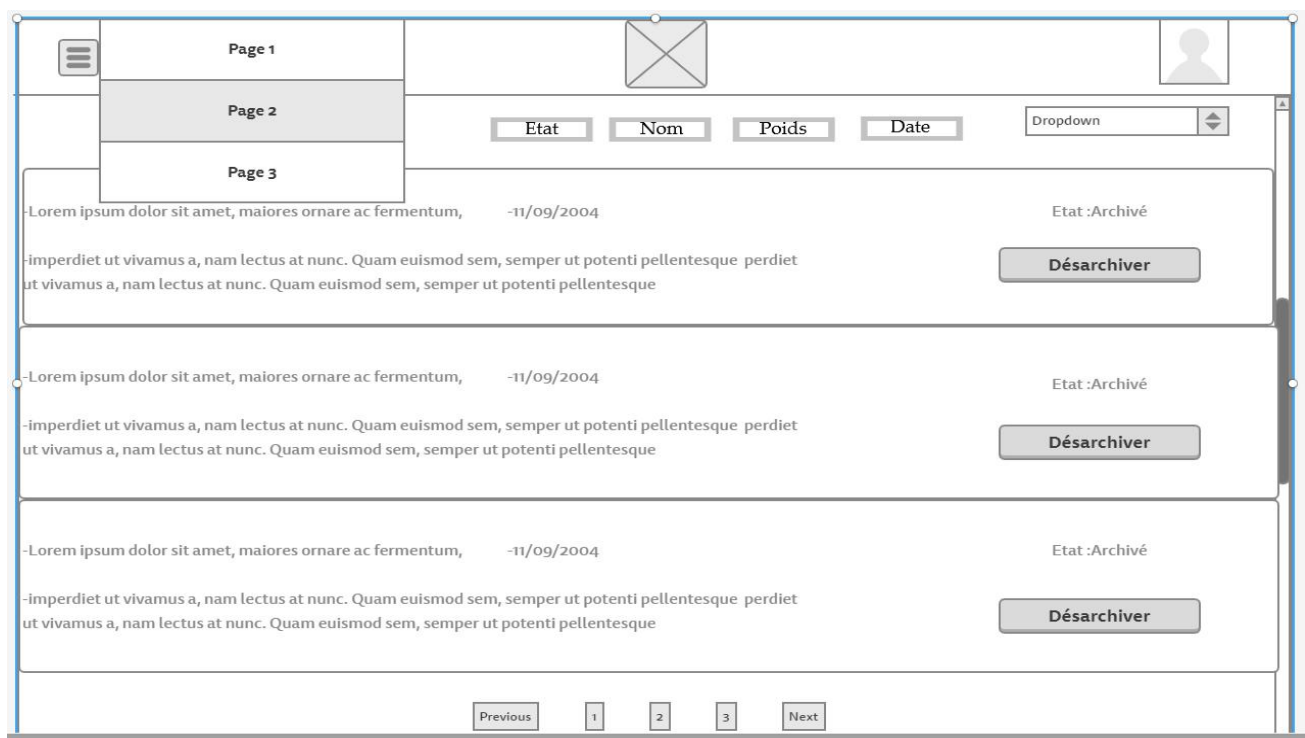
- Une hiérarchie visuelle des éléments
- Des menus accessibles et intuitifs
- Un affichage rapide
- Des textes lisibles avec un bon choix de couleurs

On a trois pages pour le site. La première page est la page d'authentification LDAP, la deuxième page est la page principale, elle est dédiée à l'affichage des projets. Enfin la troisième page est la page des statistiques.



A wireframe of a login page. It features a central heading "Login" above two input fields for username and password. Below the password field is a "Connexion" button. The entire page is set against a light gray grid background. At the top, there are three small circles representing window controls. At the bottom, there is a horizontal bar with a repeating pattern of small red dots.

Figure 6 :Maquette de la page d'authentification



A wireframe of a main page layout. It includes a sidebar on the left with a menu icon and three items labeled "Page 1", "Page 2", and "Page 3". The main content area has a header with a close button (X icon) and a user profile icon. Below the header is a table with columns: "Etat", "Nom", "Poids", "Date", and a "Dropdown" menu. The table contains three rows of data, each with a "Désarchiver" button. The footer has a "Previous" button, a pagination list with "1", "2", and "3", and a "Next" button.

Etat	Nom	Poids	Date	Dropdown
-11/09/2004	-11/09/2004	-11/09/2004	-11/09/2004	-11/09/2004
-11/09/2004	-11/09/2004	-11/09/2004	-11/09/2004	-11/09/2004
-11/09/2004	-11/09/2004	-11/09/2004	-11/09/2004	-11/09/2004

Figure 7 :Maquette de la page principale

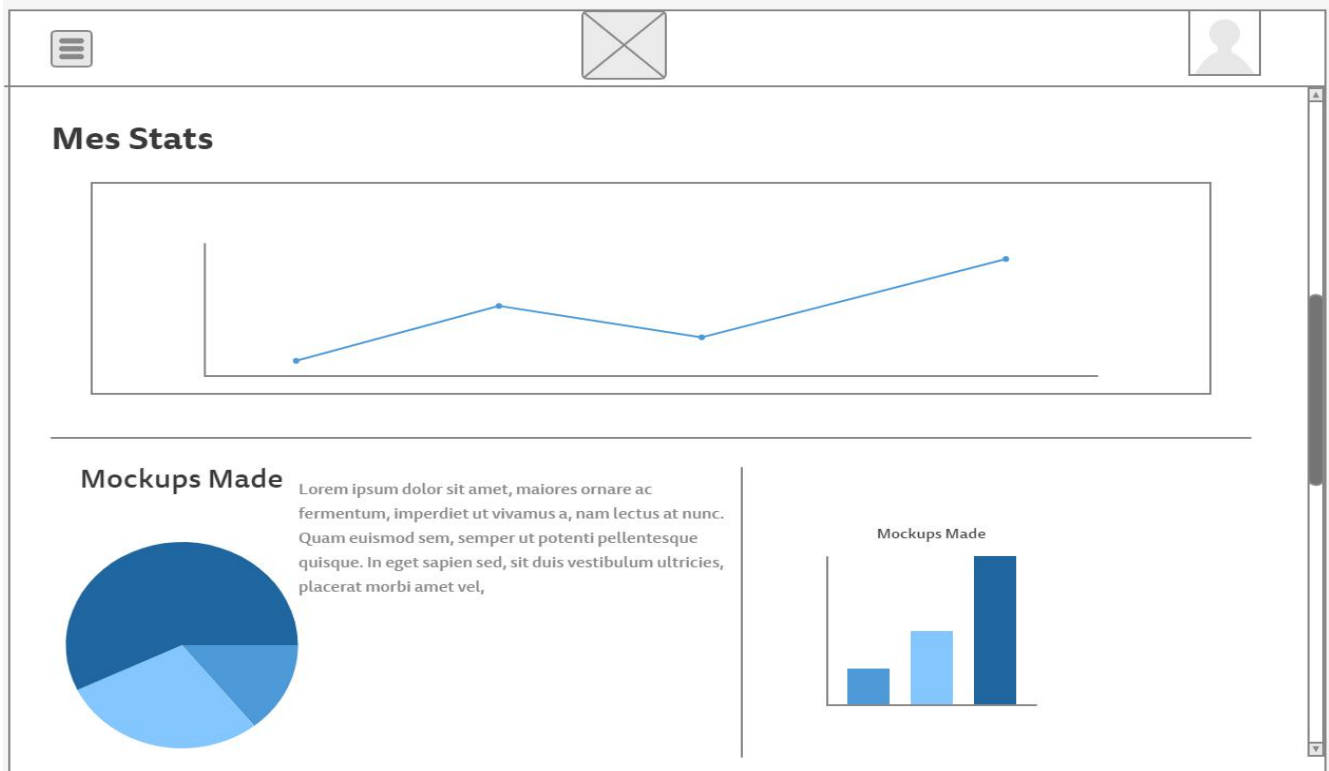


Figure 7 : Maquette de la page des statistiques

IV.3. Modélisation de la base de données

Notre base de données SQL comporte quatre tables : la table des archives, la table des utilisateurs, la table des historiques et la table des étapes d'archivage.

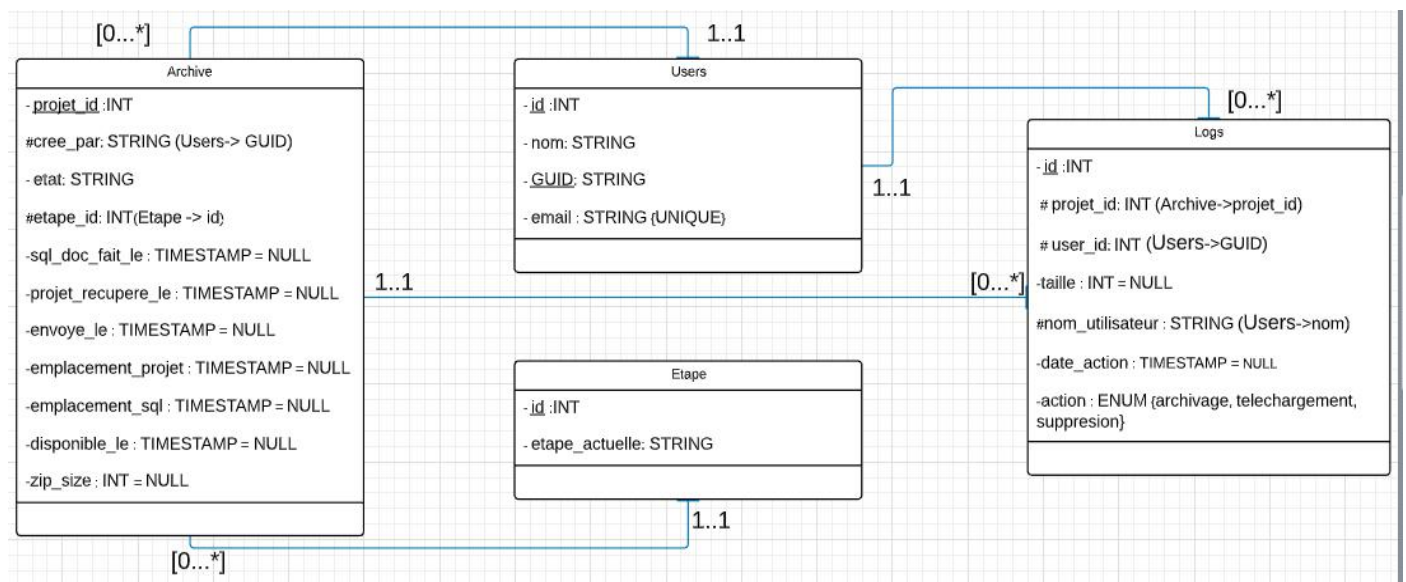


Figure 8 : Diagramme UML des tables de notre projet

Etape:

- id: Clé primaire
- etape_actuelle : Il y a quatre étapes pour le processus d'archivage.

1	Vérification système (espace mémoire disponible...)
2	Récupération du projet (bimpro, profi, centrinfo...)
3	Récupération du fichier SQL
4	Envoi vers le cloud OVH

Users: Un utilisateur peut archiver 0 ou plusieurs projets

- id: Clé primaire
- nom: Prénom et nom des utilisateurs récupérés du serveur LDAP
- GUID : Identifiant unique LDAP pour chaque utilisateur
- email : Adresse mail utilisé pour l'authentification LDAP

Archive : Un projet est uniquement archivé par un utilisateur

- projet_id : Clé primaire; chaque projet a un numéro identifiant unique.
- cree_par : Clé étrangère sur la table "Users"; GUID de la personne demandant l'archivage du projet.
- etat : Représente l'état actuel de l'archivage. Dans le cas d'une exception levée, elle garde le message d'erreur et le nom du fichier PHP déclenchant l'erreur.
- etape_id : Clé étrangère sur la table "Etape", prend en valeur une des quatre étapes décrit dessus.
- sql_doc_fait_le : Date de récupération du fichier SQL.
- projet_recupere_le : Date de récupération du projet de Gespro V1
- envoie_le : Date d'envoi du projet vers le cloud OVH.
- emplacement_projet : Répertoire utilisé pour placer le projet récupéré de Gespro V1.
- emplacement_sql: Répertoire utilisé pour placer le fichier SQL.
- disponible_le: Date de disponibilité pour télécharger le projet archivé.
- zip_size: Taille en octet du projet archivé.

Logs (Historiques) :

- id : Clé primaire
- projet_id : Clé étrangère sur la table “Archive”, numéro identifiant du projet .
- user_id : Clé étrangère sur la table “Users”; personne ayant fait l’action.
- taille : Taille du projet archivé ou téléchargé.
- nom_utilisateur: Clé étrangère sur la table “Users”; nom de la personne ayant fait l’action.
- date_action: Date de demande de téléchargement/archivage/suppression.
- action: Il existe trois actions possibles pour un projet : le téléchargement, l’archivage et la suppression.

,

IV.4. Etudes des risques et mise en œuvre des solutions

Comme tout projet de développement, notre projet est susceptible de rencontrer des obstacles techniques. Pour éviter cela, il faut faire une analyse de risque et créer un plan de mitigation. Dans le cas de ce projet, l’analyse des risques est très importante parce que le processus d’archivage est un processus prenant plusieurs heures. Pendant cette durée, toute apparition d’erreur non prévue peut provoquer des pertes de données irréversible.

Risque #1 : Panne du serveur LDAP, authentification impossible

Solution : Créer et garder une copie des utilisateurs dans la base de données SQL locale

Risque #2 : Coupure électricité ou Internet pendant l’archivage

Solution : Garder le projet ZIP en copie locale dans un dossier temporaire. Dans le cas où il y a aucune coupure, supprimer ce dossier temporaire à la fin de l’archivage. Dans le cas d’une coupure, sauter le processus de récupération de projet et directement chercher ce dossier temporaire pour continuer l’archivage.

Risque #3 : Espace insuffisant sur le serveur lors de la récupération du projet

Solution : Avant l’archivage, récupérer la taille du projet et faire un calcul pour déterminer s’il y aura de l’espace restant pour la création du ZIP. S’il n’y a pas d’espace suffisant, notifier l’utilisateur avec un message d’erreur.

V. Réalisation

V.1. Calendrier de travail

Le projet était divisé en quatre phases :

- Conception : La phase la plus importante du projet. On définit les tables de la base de données, on crée les maquettes du site, on fait une analyse de l'existant et on détermine le déroulement des tâches processus d'archivage.

Cette phase nous oriente sur le développement du projet et permet d'éviter les erreurs dans le futur. Plus on passe du temps sur la phase de conception, plus notre code est structuré et on atteint les objectifs dans les délais. Le temps prévu pour cette phase était **deux semaines**, allant du 01/06/20 jusqu'au 15/06/20.

- Développement : Cette phase est dédiée au codage du projet. Elle inclue le temps utilisé pour les installations des logiciels et leurs configurations. J'ai divisé cette phase en deux : une première sous-phase représentant le développement d'authentification LDAP, et une deuxième sous-phase qui est dédié à l'implémentation de l'outil OpenStack et des scripts.

En total, le temps prévu pour cette phase était **1 mois et 15 jours**, allant du 15/06/20 jusqu'au 20/07/20.

- Test : La phase du test permet de garantir que toutes les fonctionnalités sont bonnes et présentent aucune erreur. Laravel intègre un module permettant de faire des test unitaires et fonctionnels.

Le temps prévu pour cette phase était **10 jours**, allant du 20/07/20 jusqu'au 30/07/20.

Chaque semaine j'avais une séance de revue de code durant 1 heure où deux développeurs me donnait des points d'amélioration sur le code. Cette séance était très importante vis-à-vis le fait que je n'avais pas beaucoup d'expérience avec Laravel.

- Amélioration : Les 15 jours du mois d'Août étaient réservés pour améliorer le code en ajoutant des fonctionnalités supplémentaire qui ne sont pas dans le cahier des charges. Dans mon cas c'était le développement d'un système de notification qui notifie l'utilisateur lorsque un projet est archivé avec succès. L'archivage peut prendre entre quelques minutes à plusieurs heures. L'utilisateur ne va pas forcément attendre la fin de l'archive. Il sera donc notifié à travers la messagerie Discord de la fin d'un archivage d'un projet.

J'ai gardé un diagramme Gantt que je mettais à jour à chaque fonctionnalité complétée :

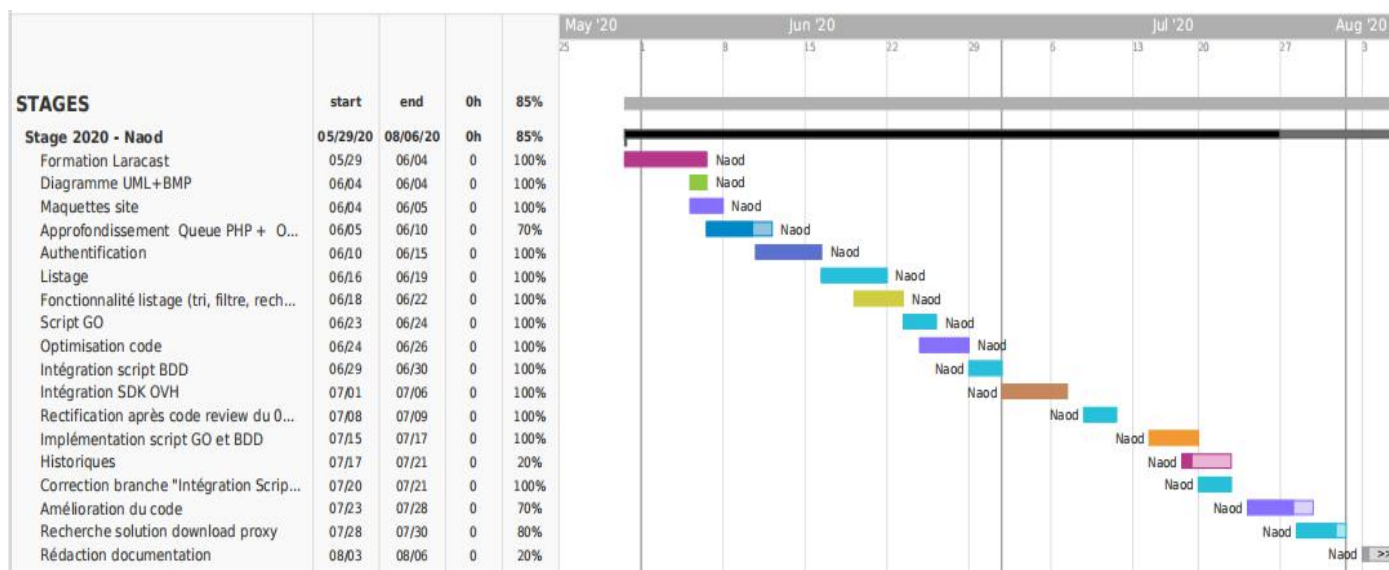


Figure 9 : Diagramme Gantt des tâches effectuées

Comme vous pouvez le constater il y a eu un temps dédié à la rédaction de la documentation. Le site final ne sera pas hébergé sur mon ordinateur mais sur le serveur Gespro. Il faut donc refaire toutes les étapes d'installation, tel que la configuration du serveur LDAP, la création de la base de données sur le serveur... Il est très important de rédiger la documentation car l'utilisateur saura comment installer et utiliser l'application.

V.2. Développement backend

Synchronisation avec le serveur LDAP

Lorsqu'on construit une application avec Laravel on a de nombreuses tâches à accomplir, comme par exemple créer des classes, créer les contrôleurs, créer les tables ...

C'est là qu'intervient Artisan, le compagnon indispensable. Il fonctionne en ligne de commande, donc à partir de la console.

Pour pouvoir intégrer l'authentification LDAP il faut au début installer la librairie LdapRecord avec Artisan.

```

BASH
$ _ php artisan vendor:publish --provider="LdapRecord\Laravel\LdapServiceProvider"

```

Après avoir installé la librairie, il faut configurer un fichier de configuration appelé le fichier env. Il s'agit d'un simple fichier texte utilisé pour définir certaines variables que nous souhaitons transmettre à notre application. Le fichier env est utilisé pour garder les mots de passe et les informations confidentielles. Il est conseillé de ne pas directement mettre en clair les informations privées dans le code et de plutôt utiliser ce fichier.

On précise toutes nos variables LDAP dans le fichier env :

```
LDAP_LOGGING=true
LDAP_CONNECTION=default
LDAP_HOST=127.0.0.1
LDAP_USERNAME="cn=user,dc=local,dc=com"
LDAP_PASSWORD=secret
LDAP_PORT=389
LDAP_BASE_DN="dc=local,dc=com"
LDAP_TIMEOUT=5
LDAP_SSL=false
LDAP_TLS=false
```

Il faut définir une page de connexion qui sera utilisé par la librairie, par défaut Laravel utilise le chemin URL www.notre_site.com/login. Tous les chemins URL sont définis dans un fichier (voir la fin du doc).

Lorsqu'un utilisateur se connecte, le contrôleur PHP récupère ses identifiants et l'envoie au serveur LDAP. Si les entrées ne sont pas présentes dans le serveur, une réponse d'erreur est envoyée et l'utilisateur doit se ré-authentifier. Dans le cas contraire, il sera redirigé vers la page d'accueil.

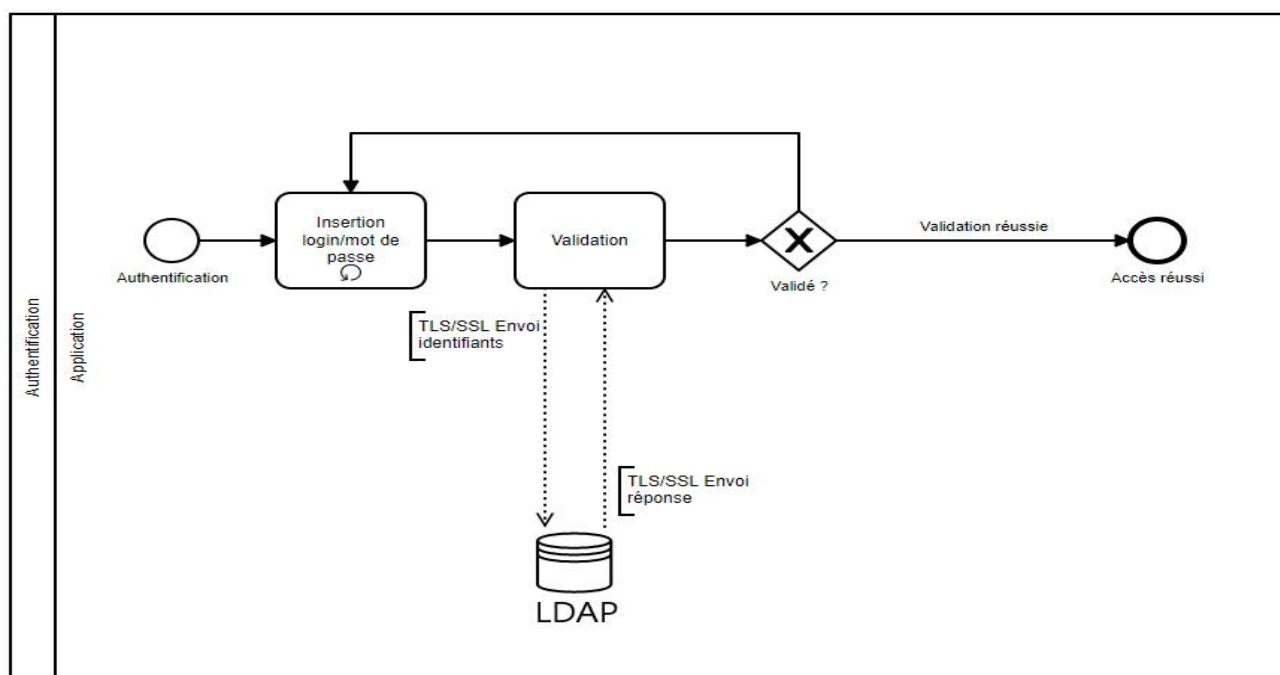


Figure 10 : BPMN du processus d'authentification

Utilisation des queues Laravel

Le processus d'archivage est un traitement long, la nature "synchrone" de PHP fait que cette opération bloque toutes autres actions pendant son traitement. On souhaite que d'autres actions soient exécutées en parallèle. La mise en place d'un système de file d'attente, que Laravel appelle "Queues" va permettre de déléguer une partie des traitements à un processus séparé et ainsi d'améliorer les performances de l'application.

Pour créer un processus queue on utilise encore une fois Artisan avec la commande :

```
php artisan make:job Archivage
```

A l'issue de cette commande un code squelette est automatiquement générée qu'on devra compléter.

On crée quatre processus queues correspondant à chaque phase de notre table "étape" : la vérification d'espace disponible, la récupération du projet, la récupération du fichier SQL et l'envoi vers le cloud OVH.

Vérification d'espace disponible :

Le premier processus queue vérifie deux conditions : l'existence du fichier SQL et la disponibilité d'une marge de 10% pour l'espace.

Comme décrit précédemment, le fichier SQL est récupéré du site Gespro V1 en faisant un appel HTTP. Lorsque le site Gespro V1 reçoit notre requête il récupère toutes les données SQL du projet et nous l'envoie sous la forme d'un fichier texte. Cependant, il peut survenir une erreur durant le traitement; on ne sait pas s'il y a eu une erreur puisque le traitement se fait au niveau du site Gespro V1. Pour éviter alors une attente infinie, le site Gespro V1 nous envoie une réponse sur l'état du fichier SQL. Elle représente la première vérification.

```
2      //Enter your code here, enjoy!
3      url_betagespro = \Config::get('script.uri_gespro'); // cle recuperee du fichier env
4
5      $url_script_bdd = $url_betagespro.'/index.php?action=116&idProjet='.$this->projet_id.'&type=datas';
6
7      $requete = Http::get($url_script_bdd); // envoi de la requete avec l'id du projet souhaité
8
9      $json_response = json_decode($requete,true); // réponse sous format JSON retourné
10
11
12  /**
13   * Dans le cas ou la reponse ne correspond pas a "success" , on lance une exception
14   */
15  if($json_response['status'] != "success") {
16      throw new \Exception( 'Le projet désiré n\'est pas trouvé dans la prod V1');|
17  }
```

La deuxième vérification correspond à l'espace mémoire, elle est précisée dans le cahier des charges. Il faut qu'il y ait au moins 10% d'espace libre pendant l'archivage. Lorsque l'on récupère le

projet, on recopie ses contenus dans un ZIP et une fois envoyé la copie sera supprimée . Cependant, la copie occupe d'espace mémoire il faut donc vérifier l'espace disponible avant l'archivage.

Dans un exemple, on a un projet que l'on souhaite archiver qui pèse 705 Go et un espace disponible de 800 Go : la marge de 10% correspond à 80 Go. Cela signifie que l'espace restant après l'archivage ne doit pas être inférieure à 80 Go. Dans cet exemple, l'espace qui sera disponible après l'archivage est 95Go (800-705), il est supérieur de 80 Go; on peut donc faire l'archivage.

```

1  /**
2   * Fonction qui vérifie l'espace restant sur le serveur.
3   * @param $projet_id
4   * @return bool
5   */
6  private function espaceDispo($projet_id)
7  {
8      // execute une commande shell
9      $shell = Terminal::with([
10         'emplacement' => \Config::get('script.emplacement_archivage'),
11     ])->run('df -BG -h --output=avail {{ $emplacement }} | sed \'1d\' | sed s/./\//');
12
13     $espace_dispo = $shell->output(); //on recupere l'espace restant total
14
15     $projet = Projet::find($projet_id); // on pioche de la bdd le projet souhaité
16
17     $taille_projet = (int) octetsToGo($projet->tailleProjet); //taille du projet en Go
18
19     //Si un marge de 10% est disponible on retourne "true", sinon retourne on "false"
20     return [(( (int) $espace_dispo - ((int) $espace_dispo * 0.1)) >= $taille_projet), $espace_dispo ];
21 }

```

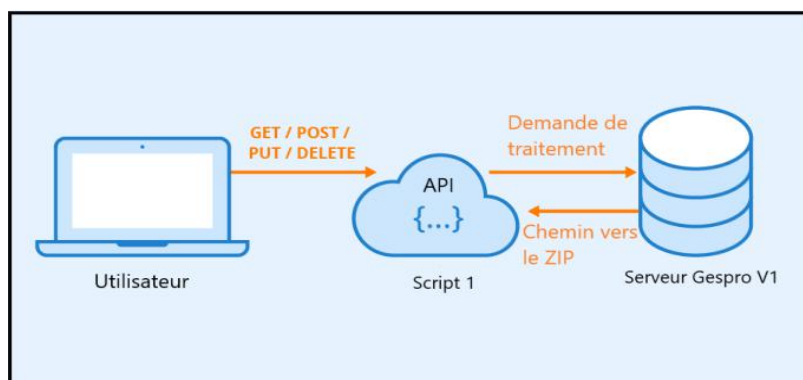
La librairie Termius nous permet d'exécuter des commandes Linux depuis du code PHP, on l'utilise à la ligne 9 pour récupérer la taille disponible sur le serveur.

Intégration des scripts

On a deux scripts à intégrer dans l'application, créés par deux développeurs Gespro. Ces scripts fonctionnent en interrogeant le serveur Gespro V1. Pour des raisons de sécurité, j'ai accès au serveur qu'à travers ces scripts

Script récupérant le projet :

Une fois la vérification validée, on récupère le projet avec le premier script. En effet, ce script agit comme une *API* voulant dire qu'il est utilisé comme un point de relais entre nous et le serveur Gespro V1.



On fait une requête HTTP POST vers le script avec toutes les informations nécessaires :

```
public function handle()
{
    /**
     * On déclenche le script GO avec l'emplacement désiré, l'ID du projet et la fonction post-Callback
     * https://gitlab.gespro.fr/root/backupprojectv1
     */
    $response = Http::post(\Config::get('script.script_go_url').'/dump', [
        'ProjectID' => (int) $this->projet_id,
        'OutputDir' => \Config::get('script.emplacement_archivage'),
        'Callback' => \Config::get('script.script_go_no_port').'/api/PROJET',
    ]);
}
```

La fonction `handle()` est présente dans chaque processus queue, elle est responsable de l'exécution de l'action désirée.

ProjectID : Correspond à l'id du projet que l'on souhaite archiver

OutputDir : On précise le répertoire d'emplacement du ZIP

Callback : Lorsque le serveur a récupéré le projet il contacte notre application en faisant appel au chemin URL précisé, c'est le "endpoint". Dans notre cas le "endpoint" est `www.notre_site.com/api/PROJET`

```
Route::post('/PROJET', "ArchivageController@recuperationProjetCallback");
```

L'application est basée sur le modèle MVC donc lorsque le "endpoint" est appelé le contrôleur "recuperationProjetCallback" est exécuté. Il met à jour la table "Archive" pour le projet correspondant et démarre le prochain processus queue responsable de la récupération du fichier SQL.

```
1 /**
2  * On met à jour la table archive et on déclenche la récupération de la BDD
3  */
4  $archive = Archive::where('projet_id', $event->projet_id)->orderBy('id', 'desc')->first();
5  $archive->etape_id = Etape::where('etape_actuelle', 'recuperation_bdd')->first()->id;
6  $archive->projet_recupere_le = Carbon::now(); //on enregistre la date actuelle
7  $archive->emplacement_projet = $event->path; // on enregistre l'emplacement du ZIP
8  $archive->save(); // on enregistre le nouvel etat de l'archive
9
10 RecupBDD::dispatch($event->projet_id); //on execute le script qui recupere le fichier SQL
```

A la ligne 6, on utilise la librairie Carbon. Une librairie très puissante qui gère les opérations sur les dates.

Script récupérant le fichier SQL:

La vérification validée et le ZIP récupéré, on déclenche ensuite le deuxième script qui récupère le fichier SQL.

```

1  /**
2      * On télécharge le fichier sql
3      */
4      $request = new Client([
5          // Base URI is used with relative requests
6          'base_uri' => \Config::get('script.uri_gespro'),
7      ]);
8
9  |
10     $response = $request->get($path_telechargement);
11
12     $file = $response->getBody()->getContents();    // on enregistre le contenu de la requete
13
14     Storage::put('archivage/projet_id_'. $this->projet_id.'.sql', $file);

```

L'adresse du site est enregistrée dans le fichier *env* (`Config::get('script.uri_gespro')`). On envoie la requête avec paramètre l'id du projet, c'est la variable *\$path_telechargement*, et on sauvegarde la réponse de la requête contenant les données SQL dans un nouveau fichier.

Implementationn du SDK OpenStack

A cette étape on a récupéré le fichier SQL et le projet sous format ZIP; on déplace le fichier SQL dans le ZIP.

```

if(Zip::check(\Config::get('script.emplacement_archivage').$event->projet_id.'.zip')) {
    $zip = Zip::open( \Config::get('script.emplacement_archivage').$event->projet_id.'.zip');
    $zip->add($path.'/projet_id_'. $event->projet_id.'.sql');
    $zip->close();
}

```

Maintenant il reste qu'à envoyer le ZIP chez notre hébergeur de serveur OVH. C'est le SDK OpenStack, vu dans *III.3*, qui va nous permettre de faire cela.

Pour les fichiers en-dessous de 5 Go, l'envoi est fait en un seul coup. Cependant, pour les fichiers supérieurs à 5 Go OpenSwift utilise un concept nommé DLO ou *Dynamic Large Object*. Le fichier est découpé en plusieurs segments et chaque segment est envoyé indépendamment. On peut définir la taille d'un segment; le minimum étant 1 Mo et le maximum 4.99 Go.

```

94     if(octetsToGo($taille_zip_octets) >= 5 ) {
95
96         $options['stream'] = new Stream(fopen(\Config::get('script.emplacement_archivage').$this->projet_id.'.zip', 'r'));
97
98         // optionnel: specifie la taille de chaque segment en octet
99         $options['segmentSize'] = 4294967296; // -> 4 Go
100
101         // optionnel: specifie le conteneur où les segments vont être présents.
102         $options['segmentContainer'] = \Config::get('ovh.ovh_container');
103
104
105         /** @var \OpenStack\ObjectStore\v1\Models\StorageObject $object */
106         $object = $openstack->objectStoreV1()
107             ->getContainer(\Config::get('ovh.ovh_container'))
108             ->createLargeObject($options);

```

\$options['stream'] : Ouvre le fichier ZIP en mode lecture
 \$options['segmentSize'] : On définit la taille qu'aura chaque segment
 \$options['segmentContainer'] : Le nom de notre conteneur OVH

Téléchargement:

Lorsqu'un utilisateur souhaite télécharger une archive, le téléchargement est fait sur son ordinateur. Pour faire cela, notre application va être un point de passage entre le client et l'hébergeur OVH. Lorsqu'on demande le téléchargement, notre application va contacter le serveur OVH. Ensuite le serveur OVH nous envoie l'archive sous format ZIP. A cette étape, au lieu de sauvegarder le ZIP, notre application le retransmet vers le client. C'est ce qu'on appelle un serveur *proxy*.

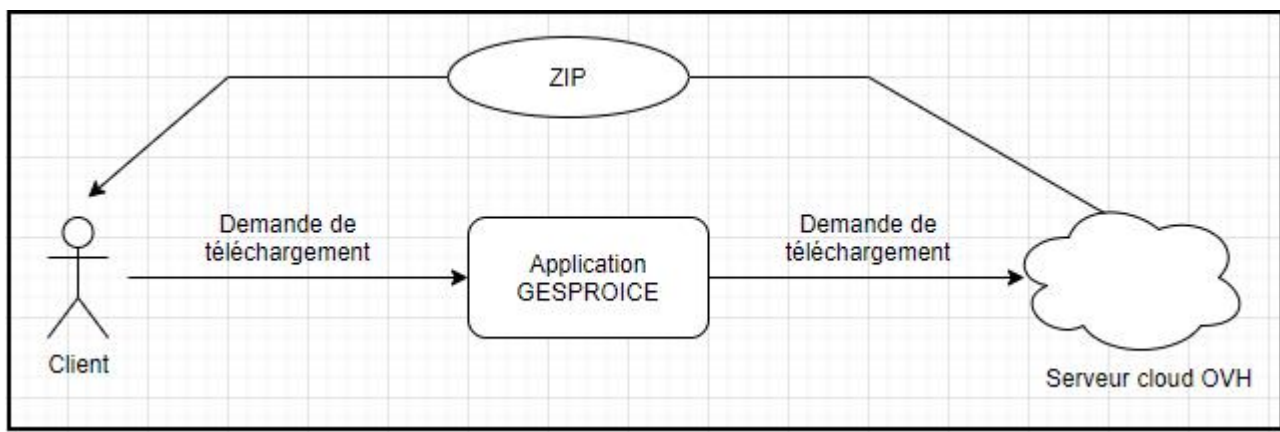


Figure 11 : Téléchargement en mode proxy

Comme mentionné précédemment, le téléchargement n'est pas toujours disponible. OVH a un délai de traitement pour récupérer l'archive. Avant la demande de téléchargement, il faut alors faire une demande de descellement. OVH va nous envoyer par la suite une date de disponibilité pour télécharger le projet. On met à jour la table "Archive" pour le projet correspondant et notifie le client.

```

1 case 'unsealing' :
2 /**
3  * Si une date de disponibilite n'est pas spécifiée, on l'instancie pour la première fois.
4  * OVH règle un archive qui n'est pas récupéré après 24h de demande de dégelassions
5  */
6 if(!$date_disponibilite || $date_disponibilite->addDay()->lessThanOrEqualTo(Carbon::now())){
7
8 $archive_to_download->disponible_le = Carbon::now()->addSeconds($archive->policy_retrieval_delay);
9
10 $archive_to_download->save();
11
12 }
13
14 return back()->withErrors(array($request->projet_id => 'Archive disponible le '.
15                               $archive_to_download->disponible_le));
  
```


Quand l'archive est disponible, on le télécharge la methode download() du SDK OpenStack. On ne doit pas sauvegarder l'archive dans notre serveur, on le redirige vers le client. Pour faire cela, chaque 1 Mo de données qu'on reçoit, on le retransmets directement vers le client. Le navigateur Web client s'occupera à concaténer ces données pour former l'intégralité du ZIP.

```

1 case 'unsealed' : // l'archive est disponible
2
3     $taille_archive = archiveTaille($request->projet_id);
4
5     return response()->streamDownload(function() use ($stream){
6         $body = $stream->download([
7             'guzzle' => [
8                 'stream' => true
9             ]
10        ]);
11
12        while (!$body->eof()) {
13            echo '400\r\n'. $body->read(1024). '\r\n';
14        }
15        echo '0\r\n\r\n';
16    }, 'Archive_'. $request->projet_id. '.zip');|
17 }

```

V.2. Développement frontend

Templates Blades

Pour afficher tous les projets sur la page Web, on ne le fera pas un par un. Cela sera long et fastidieux; il faut donc écrire du code PHP dans les balises HTML. Laravel propose le moteur de template Blade. Les fonctionnalités Blade sont nombreuses mais une des fonctionnalités proposée est l'utilisation des boucles pour afficher des éléments.

```

@foreach($projets as $projet)
    <details id="detail {{$projet->idProjet}}">
        <summary @if($projet->etape_actuelle) onclick="window.load_enCours({{{$projet->idProjet}}}">
        <b>{{$projet->nomProjet}} @if($errors->has($projet->idProjet) || $projet->disponible_le)

```

Blade permet aussi d'hériter une page de l'autre. Avec cette méthode de layout, nous allons économiser pas mal de code, car la structure de base (celle de la page mère) sera héritée sur chacune des pages filles. Les pages filles ne contiendront donc pas les balises répétitives. De cette manière, une simple modification de la page mère modifie toutes les pages filles.

```

1 @extends('master')
2 @section('listage')

```

Pages dynamiques avec JavaScript

On utilise JavaScript pour les animations et pour rendre la page dynamique. Cependant, on utilise une fonctionnalité JS très importante côté client : lorsqu'un projet est en cours d'archivage, le client souhaite connaître les étapes réussies et l'étape en cours. On a donc une fonction qui envoie des requêtes *AJAX* chaque seconde pour interroger la base de données et récupérer l'état actuel de l'archivage.

```
/**
 * Fonctions successives Ajax qui vérifie l'état actuel d'un archivage
 * @param projet_id
 */
function check_etat(projet_id) {
    $.ajaxSetup({
        headers: {
            'X-CSRF-TOKEN': '<?php echo csrf_token(); ?>'
        }
    });
    $.ajax({
        url: 'verifEtat/'+projet_id,
        type: 'GET',
        success: async function (data) {

            if(data.message[1] === 'fini') {
                success_icon('envoi_ovh', projet_id);
                removeFadeOut(document.getElementById('progress_bar'),1000);
                window.clearInterval(check);
            }

            if (data.message[0] === 'recuperation_bdd') {
                success_icon('recuperation_projet', projet_id);
            }

            } else if (data.message[0] === 'envoi_ovh') {
                success_icon('recuperation_projet', projet_id);
                success_icon('recuperation_bdd', projet_id);
            }
        }
    });
}
```

En fonction de la réponse retournée, on met à jour l'affichage de progression.

V.4. Contraintes et problèmes rencontrés

Au cours du projet j'ai rencontré quelques obstacles, une grande partie étaient faciles à résoudre grâce à l'étude des risques qu'on a fait durant la phase de conception.

Problème 1 : Interruption de l'archivage

Lors de la phase initiale du développement, on utilisait un contrôleur PHP pour effectuer l'archivage. Le protocole HTTP précise qu'un envoi d'une requête attend une réponse dans les 30 secondes qui suivent, si il n'y a aucun retour du côté serveur le navigateur considère la requête perdue et affiche le message d'erreur 408 "Request Timeout". On avait constamment cette erreur car le traitement d'archivage prenait largement plus de 30 secondes.

Solution

La solution qu'on a intégré était alors d'utiliser les processus queues de Laravel. Lorsque le client demande l'archivage du projet, on envoie un message temporaire indiquant le traitement de la demande. On démarre ensuite les processus queue en arrière-plan sans soucis d'interruption de l'archivage.

Problème 1 :Découper le ZIP téléchargé

En mode proxy, lorsqu'on envoie le ZIP au client on utilise la méthode download() du SDK OpenStack, elle télécharge l'intégralité de l'archive. Le problème est que le téléchargement peut prendre plusieurs minutes et on ne veut pas que le client attende. Il faut donc télécharger bout par bout le ZIP et l'envoyer au client. Malheureusement, OpenStack ne propose pas le téléchargement en partie.

Solution

La solution était d'englober la fonction download() de l'élément Stream : un objet de flux pour représenter les corps des messages de demande et de réponse. L'élément Stream permet de travailler avec différents types de données en les transformant en string.

Avec l'élément Stream on peut alors lire partie par partie le ZIP en téléchargement et l'envoyer au client. L'utilisateur recevra donc chaque seconde une partie du ZIP et ne sera pas obligé d'attendre le téléchargement complet côté serveur.

```
1 case 'unsealed' : // l'archive est disponible
2
3     $taille_archive = archiveTaille($request->projet_id);
4
5     return response()->streamDownload(function() use ($stream){
6         $body = $stream->download([
7             'guzzle' => [
8                 'stream' => true
9             ]
10        ]));
11
12        while (!$body->eof()) {
13            echo '400\r\n'. $body->read(1024). '\r\n';
14        }
15        echo '0\r\n\r\n';
16    }, 'Archive_'.$request->projet_id.'.zip');|
17 }
```

Chaque 1024 octets du ZIP qu'on reçoit, on le retransmet directement au client.

VI. Test

Laravel intègre un module de test unitaire et fonctionnel. Le test fonctionnel permet de tester toutes ou quelques fonctionnalités demandées dans le cahier des charges. On a réalisé les tests sur toutes les fonctionnalités primaires demandées, cela nous garantis que les processus vont s'exécuter comme prévu.

Voici un exemple de test fonctionnel où on vérifie qu'un utilisateur non-authentifié aura pas accès aux pages du site, le code HTTP 302 est une redirection : Laravel le redirige directement vers la page de connexion.

```
1 public function testUnauthorizedAccess()
2 {
3     $response = $this->get('accueil/archiver/12');
4     $response->assertStatus(302);
5     $response = $this->get('accueil/telecharger/12');
6     $response->assertStatus(302);
7     $response = $this->get('historiques');
8     $response->assertStatus(302);
9 }
10
11
12
13
14
15
```

Voici un autre test où on vérifie qu'on ne peut pas se connecter avec des identifiants erronés :

```
1 public function testLdapAuth() {
2 |
3     $connection = Container::getDefaultConnection();
4     $user = 'cn=Alex Boban, ou=users, dc=ldap, dc=gespro, dc=dev';
5     $this->assertTrue($connection->auth()->attempt($user, 'aboban'));
6     $this->assertFalse($connection->auth()->attempt($user, 'false_password'));
7 }
8
9
10
11
```

Conclusion

Le stage dans l'entreprise Gespro du 25/05/2020 jusqu'au 15/05/20 concernée la mise en œuvre d'un système d'archivage. Après avoir analysé les attentes du client j'ai segmenté le projet en trois phases: la conception, le développement et le test. Les trois phases étaient importantes l'une que l'autre, j'ai méticuleusement travaillé sur chacune pour assurer la réussite du stage. Comme tout projet informatique, j'ai rencontré des obstacles que j'ai pu surmonter avec le temps et la recherche.

J'ai pu intégrer toutes mes connaissances que j'ai acquises à l'IUT. Le fait d'appliquer mes compétences scolaires dans un cadre réel m'a amené satisfaction, non seulement d'un point de vue informatique mais aussi de gestion de projet. Toutes les notions vues en cours sont révélées très bénéfiques. Cette première approche professionnelle était enrichissante sur tous les plans. J'ai pu connaître la démarche et les attendus pour réaliser un projet informatique en entreprise.

Le monde de l'entreprise m'a appris que la communication est très importante, que ce soit avec le client ou l'équipe informatique; elle nous permet de livrer un produit qui est fonctionnel et satisfaisant. J'ai aussi appris que la documentation technique est cruciale pour que le client sache comment installer et utiliser un produit, c'était quelque chose que j'ai souvent négligé pendant mes

années à l'IUT. Cependant la plus grande b n fice de ce stage a  t  l'apprentissage du langage Laravel, qui m'a servi pour faire le projet et qui va sans doute me servir dans le future aussi.

En conclusion, il y avait une bonne ambiance dans l'entreprise o  j'ai eu l'opportunit  de conna tre tous les salari s et appris les astuces pour r ussir au sein d'une entreprise, qui est bien  videmment diff rent du monde scolaire. Mon encadrant de stage  tait toujours pr sent pour m'assister et pour me fournir des points d'am lioration lorsqu'il fallait. Laravel  tait un nouveau framework pour moi, il m'a fallu un temps d'apprentissage durant lequel j'ai fait face   certaines difficult s. Toute l' quipe informatique, de m me que mon encadrant  taient pr sents pour me montrer comment surmonter les difficult s de ce framework. Chaque semaine j'avais une s ance de relecture de code ou deux d veloppeurs experts me proposaient des solutions alternatives et des points d'am lioration sur mon code. Cette s ance  tait tr s importante pour mon apprentissage du langage Laravel.

Ma premi re exp rience professionnelle  tait tr s enrichissante, elle m'a permis d' voluer mes comp tences informatiques et en gagner de nouvelles. Je suis satisfait avec le produit rendu, et j'esp re de m me que Gespro va trouver satisfaction dans le travail fait. C'est avec beaucoup d'anticipation que j'attends la continuation de mon ann e en alternance chez Gespro.

R sum 

Mon stage s'est d roul    l'entreprise Gespro sp cialis e dans le domaine de la construction. Le but du stage est de concevoir et fabriquer un syst me qui va permettre   l'entreprise d'archiver des projets de construction.

La connexion au site se fera par l'interm diaire d'un serveur bas  sur le protocole LDAP. L'utilisateur pourra archiver tous les projets qui sont termin s et pourra les t l charger s'il le souhaite. L'h bergeur cloud utilis  est OVH, une entreprise fran aise.

Le projet  tait d coup  en 3 phases pour assurer un bon fonctionnement de l'application : celle de la conception, du d veloppement et du test . Apr s deux mois de travail, l'application a  t  remise   l'entreprise, elle contient toutes les fonctionnalit s primaires demand es.

Ce stage m'a fait gagner plusieurs comp tences, notamment l'utilisation du framework Laravel.

Mots-cl s : archiver, LDAP, OVH, conception, d veloppement, test, fonctionnalit s, Laravel

Abstract

My internship was held in the company Gespro, a company providing online services in real-estate construction. The mission of the internship was to plan and create a site that allows the company to archive construction projects.

The site needs to have a login page using the LDAP protocol. The user should be able to archive all completed construction projects and can download them in the future if needed. The cloud hosting service chosen is OVH.

The internship followed basic rules, it started with a conception step where we laid out the plans for the website. This was followed by a coding phase : in which we wrote and built the site; it was then concluded by the testing phase to make sure our site works properly. The product was successfully delivered after 2 months of work.

Keywords : LDAP, cloud hosting, archive, coding, testing

Glossaire

- ★ SaaS : Le SaaS, ou Logiciel en tant que Service, est un modèle de distribution de logiciel à travers le Cloud. Les applications sont hébergées par le fournisseur de service.
- ★ CLI: Une interface homme-machine dans laquelle la communication entre l'utilisateur et l'ordinateur s'effectue en mode texte.
- ★ LDAP: Protocole qui permet d'accéder à des bases d'informations sur les utilisateurs d'un réseau, via l'interrogation d'annuaires.
- ★ VM: Une machine virtuelle, ou VM (Virtual Machine), est un environnement d'application ou de système d'exploitation installé sur un logiciel qui imite un matériel dédié.
- ★ IDE : Un ensemble d'outils spécifiques dédiés aux programmeurs afin qu'ils puissent optimiser leur temps de travail et améliorer leur productivité.
- ★ SDK: Ensemble d'outils d'aide à la programmation pour concevoir des logiciels, jeux, applications mobiles, etc, pour un système d'exploitation spécifique.
- ★ MVC : Méthodologie ou motif de conception visant à faire le lien entre l'interface utilisateur et les modèles de données sous-jacents.
- ★ API : Solution informatique qui permet à des applications de communiquer entre elles et de s'échanger mutuellement des services ou des données.

Webographie

- ◆ Page Wikipédia -LDAP : https://fr.wikipedia.org/wiki/Lightweight_Directory_Access_Protocol
- ◆ Documentation Laravel : <https://laravel.com/>
- ◆ Documentation LdapRecord : <https://ldaprecord.com/docs/laravel/>

- ◆ Documentation SDK OpenStack : <https://php-openstack-sdk.readthedocs.io/en/latest/>
- ◆ Site OVH : <https://www.ovh.com/fr/>

Annexes

Page de connexion _____
Page d’affichage des projets _____
Demande de téléchargement _____
Date de disponibilité _____
Code source “RecupBDD.php” _____
Code source “ScriptGo.php” _____
Code source “EnvoiOVH.php” _____
Code source “TelechargementController.php” _____
Code source “web.php” _____
Page HTML “listage.blade.php” _____
Code source “listage.js” _____

Page de connexion



Email

Password

Log in

Page d'affichage des projets

Accueil Nom Login Poids ↓ Etat ▼ Date d'archive ▼

Recherche Valider

Projets par page ▼

projet SA ×

Description : Projet de Super Admin permettant la gestion de tous les projets créés sur le serveur.

Taille : 0

Login : spadmgestro

Etat : archive

Modules actives

Telecharger

Demande de téléchargement

Aménagement des Berges de la Bourbre
Requête de dégelassions envoyé



Date de disponibilité

Aménagement des Berges de la Bourbre
Archive disponible le 06/08 à 20:32:22



Code source “RecupBDD.php”

```
5 use App\Archive;
6 use App\Etape;
7 use Illuminate\Bus\Queueable;
8 use Illuminate\Contracts\Queue\ShouldQueue;
9 use Illuminate\Foundation\Bus\Dispatchable;
10 use Illuminate\Queue\InteractsWithQueue;
11 use Illuminate\Queue\SerializesModels;
12 use App\Events\ScriptBDDTermine;
13 use GuzzleHttp\Client;
14 use Illuminate\Support\Facades\Storage;
15
16
17 class RecupBDD implements ShouldQueue
18 {
19     use Dispatchable, InteractsWithQueue, Queueable, SerializesModels;
20
21     /**
22      * The number of times the job may be attempted.
23      *
24      * @var int
25      */
26     public $tries = 5;
27
28
29     protected $projet_id;
```

```
34      *
35      * @return void
36      */
37      public function __construct($id)
38      {
39
40          $this->projet_id = $id;
41
42      }
43
44      /**
45       * Execute the job.
46       *
47       * @return void
48       */
49
50      public function handle()
51      {
52
53          /**
54           * On récupère l'emplacement du fichier SQL sur la V1
55           */
56
57          $archive = Archive::where('projet_id', $this->projet_id)->orderBy('id', 'desc')->first();
58
59          $sql_path = $archive->emplacement_sql;
60
61          $path_telechargement = substr($sql_path,1);
62
63      /**
64       * On télécharge le fichier sql
65       */
66
67      $request = new Client([
68          // Base URI is used with relative requests
69          'base_uri' => \Config::get('script.uri_gespro'),
70      ]);
71
72
73      $response = $request->get($path_telechargement);
74
75      $file = $response->getBody()->getContents();    // on enregistre le contenu de la requete
76
77      Storage::put('archivage/projet_id_'.$this->projet_id.'.sql',$file);
78
79
80      //event qui s'occupe de la mise à jour de l'archive
81      event(new ScriptBDDTermine($this->projet_id));
82
83
84      EnvoiOVH::dispatch($this->projet_id);
85
86      }
```

```
88      /**
89       * Handle a job failure.
90       *
91       * @param \Exception $exception
92       * @return void
93       */
94      public function failed( $exception)
95      {
96
97          /**
98           * On met à jour les colonnes 'etat' et 'etape' pour spécifier qu'il y a eu une erreur
99           */
100
101          $archive = Archive::where('projet_id', $this->projet_id)->orderBy('id', 'desc')->first();
102
103          $archive->etat = $exception->getMessage().' - '.$exception->getFile().' - '.$exception->getLine();
104
105          $archive->save();
106
107          if(file_exists(\Config::get('script.emplacement_archivage').$this->projet_id.'.zip')) {
108              unlink(\Config::get('script.emplacement_archivage').$this->projet_id.'.zip');
109          }
110
111      }
112
113  }
```

Code source “ScriptGO.php”

```
15  class ScriptGO implements ShouldQueue
16  {
17      use Dispatchable, InteractsWithQueue, Queueable, SerializesModels;
18
19      /**
20       * Create a new job instance.
21       *
22       * @return void
23       */
24
25      protected $projet_id;
26
27      public function __construct($id)
28      {
29          $this->projet_id = $id;
30
31      }
32
33      /**
34       * Execute the job.
35       *
36       * @return void
37       */
```

```
38 public function handle()
39 {
40
41     /**
42      * On déclenche le script GO avec l'emplacement désiré, l'ID du projet et la fonction post-Callback
43      * https://gitlab.gespro.fr/root/backupprojectv1
44      */
45     $response = Http::post(\Config::get('script.script_go_url').'/dump', [
46         'ProjectID' => (int) $this->projet_id,
47         'OutputDir' => \Config::get('script.emplacement_archivage'),
48         'Callback' => \Config::get('script.script_go_no_port').'/api/PROJET',
49     ]);
50
51
52 }
53
54 /**
55  * Handle a job failure.
56  *
57  * @param Exception $exception
58  * @return void
59  */
60 public function failed($exception)
61 {
62
63     /**
64      * On met à jour les colonnes 'etat' et 'etape' pour spécifier qu'il y a eu une erreur
65      */
66     $archive = Archive::where('projet_id', $this->projet_id)->orderBy('id', 'desc')->first();
67
68     $archive->etat = $exception->getMessage().' | '.$exception->getFile().' | '.$exception->getLine();
69
70     $archive->save();
71
72 }
73 }
```

Code source “EnvoiOVH.php”

```
19 class EnvoiOVH implements ShouldQueue
20 {
21     use Dispatchable, InteractsWithQueue, Queueable, SerializesModels;
22
23
24     /**
25      * The number of times the job may be attempted.
26      *
27      * @var int
28      */
29     public $tries = 5;
30
31     protected $projet_id;
32
33
34     /**
35      * Create a new job instance.
36      *
37      * @return void
38      */
39     public function __construct($id)
40     {
41         $this->projet_id=$id;
42     }
43
44     /**
45      * Execute the job.
46      *
47      * @return void
48      */
49     public function handle()
50     {
51
52
53
54         /**
55          * CREDENTIELS D'AUTHENTIFICATION , A MODIFIER LE FICHIER .ENV EN PRODUCTION
56          */
57         $openstack = new OpenStack\OpenStack([
58             'authUrl' => 'https://auth.cloud.ovh.net/v3',
59             'region' => \Config::get('ovh.ovh_region'), // La region
60             'user' => [
61                 'domain' => ['id' => 'default'],
62                 'name' => \Config::get('ovh.ovh_name'),
63                 'id' => \Config::get('ovh.ovh_id'),
64                 'password' => \Config::get('ovh.ovh_password')
65             ],
66             'scope' => ['project' => [
67                 'name' => \Config::get('ovh.ovh_project'),
68                 'domain' => ['id' => 'default']]
69         ]);
70
71
72
73
74         /**
75          * Options permettant à spécifier le nom attribué au projet et l'emplacement actuel du ZIP à archiver
76          */
77         $options = [
78             'name' => 'Archive_'. $this->projet_id. '.zip',
79         ];
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

```

81     $object = $openstack->objectStoreV1()
82     ->getContainer(\Config::get('ovh.ovh_container'));
83
84
85
86     /**
87     * Si le projet est supérieur à 5 Go, il faut le segmenter en plusieurs parties
88     */
89
90     $taille_projet = Projet::find($this->projet_id)->tailleProjet;
91     $taille_zip_octets = $taille_projet != 0 ? $taille_projet : filesize(\Config::get('script.emplacement_archivage')).$this->projet_id.'.zip');
92
93
94     if(octetsToGo($taille_zip_octets) >= 5 ) {
95
96         $options['stream'] = new Stream(fopen(\Config::get('script.emplacement_archivage')).$this->projet_id.'.zip','r'));
97
98         // optionnel: specifie la taille de chaque segment en octet
99         $options['segmentSize'] = 4294967296; // -> 4 Go
100
101         // optionnel: specifie le conteneur où les segments vont être présents.
102         $options['segmentContainer'] = \Config::get('ovh.ovh_container');
103
104
105         /** @var \OpenStack\ObjectStore\v1\Models\StorageObject $object */
106         $object = $openstack->objectStoreV1()
107             ->getContainer(\Config::get('ovh.ovh_container'))
108             ->createLargeObject($options);
109
110
111     } else {
112
113         $options['stream'] = new Stream(fopen(\Config::get('script.emplacement_archivage')).$this->projet_id.'.zip','r'));
114
115         /** @var \OpenStack\ObjectStore\v1\Models\StorageObject $object */
116         $object->createObject($options);
117
118     }
119
120
121     //event qui s'occupe de la mise à jour de l'archive
122     event(new EnvoiOVHTermine($this->projet_id));
123
124
125 }
126
127 /**
128  * Handle a job failure.
129  *
130  * @param Exception $exception
131  * @return void
132  */
133 public function failed($exception)
134 {
135     /**
136     * On met à jour les colonnes 'etat' et 'etape' pour spécifier qu'il y a eu une erreur
137     */
138
139     $archive = Archive::where('projet_id', $this->projet_id)->orderBy('id', 'desc')->first();
140
141     $archive->etat = $exception->getMessage();
142
143     $archive->save();

```



```

121      //event qui s'occupe de la mise à jour de l'archive
122      event(new EnvoiOVHTermine($this->projet_id));
123
124
125  }
126
127  /**
128   * Handle a job failure.
129   *
130   * @param Exception $exception
131   * @return void
132   */
133  public function failed($exception)
134  {
135      /**
136       * On met à jour les colonnes 'etat' et 'etape' pour spécifier qu'il y a eu une erreur
137       */
138
139      $archive = Archive::where('projet_id', $this->projet_id)->orderBy('id', 'desc')->first();
140
141      $archive->etat = $exception->getMessage();
142
143      $archive->save();
144
145      if(file_exists(\Config::get('script.emplacement_archivage').$this->projet_id.'.zip')) {
146          unlink(\Config::get('script.emplacement_archivage').$this->projet_id.'.zip');
147      }
148  }
149  }

```

Code source “TelechargementController.php”

```

19  class EnvoiOVH implements ShouldQueue
20  {
21      use Dispatchable, InteractsWithQueue, Queueable, SerializesModels;
22
23
24      /**
25       * The number of times the job may be attempted.
26       *
27       * @var int
28       */
29      public $tries = 5;
30
31      protected $projet_id;
32
33
34      /**
35       * Create a new job instance.
36       *
37       * @return void
38       */
39      public function __construct($id)
40      {
41          $this->projet_id=$id;
42      }
43
44      /**
45       * Execute the job.
46       *
47       * @return void
48       */
49      public function handle()
50      {

```

```

49 public function handle()
50 {
51
52
53
54 /**
55  * CREDENTIELS D'AUTHENTIFICATION , A MODIFIER LE FICHIER .ENV EN PRODUCTION
56  */
57 $openstack = new OpenStack\OpenStack([
58     'authUrl' => 'https://auth.cloud.ovh.net/v3',
59     'region' => \Config::get('ovh.ovh_region'), // La region
60     'user' => [
61         'domain' => ['id' => 'default'],
62         'name' => \Config::get('ovh.ovh_name'),
63         'id' => \Config::get('ovh.ovh_id'),
64         'password' => \Config::get('ovh.ovh_password')
65     ],
66     'scope' => ['project' => [
67         'name' => \Config::get('ovh.ovh_project'),
68         'domain' => ['id' => 'default']]
69 ];
70
71
72
73
74 /**
75  * Options permettant à spécifier le nom attribué au projet et l'emplacement actuel du ZIP à archiver
76  */
77 $options = [
78     'name' => 'Archive_'. $this->projet_id. '.zip',
79 ];
80
81
82 $object = $openstack->objectStoreV1()
83     ->getContainer(\Config::get('ovh.ovh_container'));
84
85
86 /**
87  * Si le projet est supérieur à 5 Go, il faut le segmenter en plusieurs parties
88  */
89
90 $taille_projet = Projet::find($this->projet_id)->tailleProjet;
91 $taille_zip_octets = $taille_projet != 0 ? $taille_projet : filesize(\Config::get('script.emplacement_archivage').$this->projet_id.'.zip');
92
93
94 if(octetsToGo($taille_zip_octets) >= 5 ) {
95
96     $options['stream'] = new Stream(fopen(\Config::get('script.emplacement_archivage').$this->projet_id.'.zip','r'));
97
98     // optionnel: specifie la taille de chaque segment en octet
99     $options['segmentSize'] = 4294967296; // -> 4 Go
100
101     // optionnel: specifie le conteneur où les segments vont être présents.
102     $options['segmentContainer'] = \Config::get('ovh.ovh_container');
103
104
105     /** @var \OpenStack\ObjectStore\v1\Models\StorageObject $object */
106     $object = $openstack->objectStoreV1()
107         ->getContainer(\Config::get('ovh.ovh_container'))
108         ->createLargeObject($options);
109
110
111 } else {

```



```

114     $options['stream'] = new Stream(fopen(\Config::get('script.emplacement_archivage').$this->projet_id.'.zip','r'));
115
116     /** @var \OpenStack\ObjectStore\v1\Models\StorageObject $object */
117     $object->createObject($options);
118 }
119
120
121 //event qui s'occupe de la mise à jour de l'archive
122 event(new EnvoiOVHTermine($this->projet_id));
123
124
125 }
126
127 /**
128  * Handle a job failure.
129  *
130  * @param Exception $exception
131  * @return void
132  */
133 public function failed($exception)
134 {
135     /**
136      * On met à jour les colonnes 'etat' et 'etape' pour spécifier qu'il y a eu une erreur
137      */
138
139     $archive = Archive::where('projet_id', $this->projet_id)->orderBy('id', 'desc')->first();
140
141     $archive->etat = $exception->getMessage();
142
143     $archive->save();
144
145     //event qui s'occupe de la mise à jour de l'archive
146     event(new EnvoiOVHTermine($this->projet_id));
147
148 }
149
150 /**
151  * Handle a job failure.
152  *
153  * @param Exception $exception
154  * @return void
155  */
156 public function failed($exception)
157 {
158     /**
159      * On met à jour les colonnes 'etat' et 'etape' pour spécifier qu'il y a eu une erreur
160      */
161
162     $archive = Archive::where('projet_id', $this->projet_id)->orderBy('id', 'desc')->first();
163
164     $archive->etat = $exception->getMessage();
165
166     $archive->save();
167
168     if(file_exists(\Config::get('script.emplacement_archivage').$this->projet_id.'.zip')) {
169         unlink(\Config::get('script.emplacement_archivage').$this->projet_id.'.zip');
170     }
171 }
172
173 }
174
175 }

```

Code source “web.php” (Table des chemins)

```

27 Route::get('/deauth', "Auth\LoginController@deconnexion")->name('deconnexion');
28
29 Route::get('/historiques', "HistoriquesController@afficheHistoriques")->name('historiques')->middleware('auth');
30
31 Route::group(['prefix' => 'accueil', 'middleware' => ['auth']], function (){
32     Route::get('/', ['as' => 'accueil', 'uses' => "ListageController@tri" ]);
33     Route::get('telecharger/{projet_id}', "TelechargementController@telecharger")->name('telecharger');
34     Route::get('supprimer/{projet_id}', "SuppressionController@suppressionArchive")->name('supprimer');
35     Route::get('verifEtat/{projet_id}', "ArchivageController@verifEtat")->name('verif');
36     Route::get('verifEtatTelechargement/{projet_id}', "TelechargementController@verifEtatTelechargement")->name('verif-telechargement');
37     Route::get('archiver/{projet_id}', "ArchivageController@archiver")->name('archiver');
38     Route::get('{?tri={param1?}&etat={param6?}&recherche={param2?}&par_page={param5?}&order={param7?}}', "ListageController@tri")->name('tri');
39 });
40
41 Auth::routes();
42

```

Page HTML “listage.blade.php”

```

1 @extends('master')
2 @section('listage')
3     <html>
4     <meta name="csrf-token" content="{{csrf_token()}}">
5     {{-- Selection nombre de projets par page --}}
6     <div id="loading" class="loading">Loading&#8230;</div>
7     <div class="dropdown par-page" style="margin-left: 50px;">
8         <button class="btn btn-primary dropdown-toggle dropdown-par-page " type="button" data-toggle="dropdown">Projets par page
9         <span class="caret"></span></button>
10        <ul class="dropdown-menu" style="margin-left: -22px;">
11            @php
12                $pages_keys = [ 5, 10, 15, 20, 25, 30 ];
13            @endphp
14            @foreach($pages_keys as $key)
15                <li><a href="{{ route('tri', ['param5' => $key, 'param1'=>$_GET['tri'],
16                    'param2'=>$_GET['recherche'], 'param6'=>$_GET['etat']] }}">{{ $key }}</a></li>
17            @endforeach
18        </ul>
19    </div>
20    {{-- Definition de l'icone "x" et autre pour animer la fermeture et ouverture graphique d'un projet --}}
21    <div class="kd" >
22        <div style="visibility: hidden; position: absolute; width: 0px; height: 0px;">
23            <svg xmlns="http://www.w3.org/2000/svg">
24                <symbol viewBox="0 0 24 24" id="expand-more">
25                    <path d="M16.59 8.59L12 13.17 7.41 8.59 6 10.16 6 6-6z"/>
26                    <path d="M0 0h24v24H0z" fill="none"/>
27                </symbol>
28                <symbol viewBox="0 0 24 24" id="close">
29                    <path d="M19 6.41L17.59 5 12 10.59 6.41 5 5 6.41 10.59 12 5 17.59 6.41 19 12 13.41 17.59 19 17.59 13.41 12z"/>
30                    <path d="M0 0h24v24H0z" fill="none"/>
31                </symbol>
32            </svg>
33        </div>
34        {{-- Definition d'un projet pour chaque projet dans la BD --}}

```

```

35     {{-- S'il y a un résultat retourné, on affiche tous les projet. NB: Aucun résultat est retourné si le paramètre de filtrage ou de recherche corespond à aucun
36
37     @if($projets)
38         <div id="container-projets" style="padding-bottom: 50px;">
39             @foreach($projets as $projet)
40                 <details id="detail" {{ $projet->idProjet }}>
41                     <summary @if($projet->etape_actuelle) onclick="window.load_enCours('{{ $projet->idProjet }}'" @endif>
42                         <b>{{ $projet->nomProjet }} @if($errors->has($projet->idProjet) || $projet->disponible_1e) <p style="color: red;"> {{ ($projet->disponible_1e ? $
43
44                         <svg class="control-icon control-icon-expand" width="24" height="24" role="presentation">
45                             <use xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href="#expand-more" />
46                         </svg>
47                         <svg class="control-icon control-icon-close" width="24" height="24" role="presentation">
48                             <use xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href="#close" />
49                         </svg>
50                     </summary>
51                     <br>
52                     <p>Description : {{ $projet->description }} </p>
53                     <br>
54                     <p>Taille : @if($projet->tailleProjet){ {{ afficherPoids($projet->tailleProjet) }} } @else 0 @endif </p>
55                     <p>Login : {{ $projet->login }} </p>
56                     {{-- On définit la couleur de la bordure de chaque projet en fonction de son état; comme définidans le CDC --}}
57                     @if($projet->etat === "actif")
58                         <p style="border: 5px solid yellow">Etat : {{ $projet->etat }} </p>
59                     @elseif($projet->etat === "bloque")
60                         <p style="border: 5px solid red">Etat : {{ $projet->etat }} </p>
61                     @elseif($projet->etat === "termine")
62                         <p style="border: 5px solid darkgreen">Etat : {{ $projet->etat }} </p>
63                     @elseif($projet->etat === "a_archiver")
64                         <p class="light_green">Etat : {{ $projet->etat }} </p>
65                     @elseif($projet->etat === "archive")
66                         <p class="green">Etat : {{ $projet->etat }} </p>
67                     @endif
68                 <div id="archivage_precondition_message" {{ $projet->idProjet }}></div>
69                 {{-- On affiche chaque module d'un projet --}}
70                 <div id="container_bas">
71                     <div id="container-bas" {{ $projet->idProjet }}>
72                         <p>Modules actifs </p>
73                         <ul>
74                             @if($projet->centrinfo)
75                                 <li> CentrInfo</li>
76                             @endif
77                             @if($projet->difdoc)
78                                 <li> Dif Doc</li>
79                             @endif
80                             @if($projet->profi)
81                                 <li> Profi</li>
82                             @endif
83                             @if($projet->oreli)
84                                 <li> Oreli</li>
85                             @endif
86                             @if($projet->bimpro)
87                                 <li> Bimpro</li>
88                             @endif
89                         </ul>
90                     </div>
91                     @if($projet->etat === "a_archiver")
92                         <a id="archiver_button" {{ $projet->idProjet }} onclick="window.archiver('{{ $projet->idProjet }}'" class="btn btn-primary-3" > <h5>Arch
93                     @elseif($projet->etat === "archive" && !$projet->attente)
94                         <a id="desarchiver_button2" {{ $projet->idProjet }} href="{{ route('telecharger', ['projet_id' => $projet->idProjet]) }}" onclick="retu
95
96                     {{--Bouton suppression--}}
97                     {{-- @else--}}
98                     <a id="desarchiver_button2" {{ $projet->idProjet }} href="{{ route('supprimer', ['projet_id' => $projet->idProjet]) }}" onclick="re
99
100                 @endif
101             </div>
102         </details>
103     @endforeach
104 </div>
105 </div>

```

```
106 <!-- The Modal -->
107 <div id="myModal" class="modal">
108
109     <!-- Modal content -->
110     <div class="modal-content">
111         <div class="modal-header">
112             <span class="close">&times;</span>
113             <h3 id="modal-title" > </h3>
114         </div>
115         <div class="modal-body">
116             <p id="modal-error"> </p>
117         </div>
118     </div>
119
120 </div>
121
122
123
124 @if($projets)
125     <div class="paginator"> {{ $projets->links() }} </div>
126 @endif
127 @show
128 </html>
```

Code source “listage.js”

```
1  var loader = false;
2  var uploading =false;
3  var archive = false;
4  var check;
5
6
7  /**
8   * Fonction qu envoie une requête Ajax pour démarrer l'archivage
9   * @param projet_id
10  */
11  window.archiver = function (projet_id) {
12      let confirme = confirm("Confirmez l'archivage de ce projet ");
13
14      if(confirme){
15          clear(projet_id);
16          success_content(projet_id);
17          $.ajaxSetup({
18              headers: {
19                  'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')
20              }
21          });
22          $.ajax({
23              url: 'archiver/'+projet_id,
24              type: 'GET',
25              success: async function (data) {
26
27              },
28              error: function (e) {
29                  console.log(e);
30                  let datas = JSON.parse(e.responseText);
31                  modal(datas.message, 'Erreur');
32                  clearInterval(check);
33                  document.getElementById('progress_bar').remove();
34
35              }
```



```
36     });
37     check = window.setInterval(function(){
38         check_etat(projet_id);
39     }, 1000)
40
41 }
42
43 }
44
45 /**
46  * Charge tous Les projets qui étaient en archivage lors de la réouverture d'une nouvelle fenêtre
47  * @param projet_id
48  */
49 window.load_enCours = function(projet_id) {
50     clear(projet_id);
51     success_content(projet_id);
52     check = window.setInterval(function(){
53         check_etat(projet_id);
54     }, 1000)
55 }
56
57 /**
58  * Enlève La partie inférieure du conteneur projet pour laisser place à la progression
59  * @param projet_id
60  */
61 function clear(projet_id) {
62     let element = document.getElementById("container-bas "+projet_id);
63
64     if(element) {
65         removeFadeOut(element, 2000);
66         sleep(2000);
67
68         let divd = document.createElement("div");
69         divd.className = "lds-ring";
70         divd.id = "loader";
```

```
72     for(let i=0; i<4; i++){
73         let div = document.createElement("div");
74         divd.appendChild(div);
75     }
76
77     let container = document.getElementById("archivage_precondition_message "+projet_id);
78     container.appendChild(divd);
79 }
80
81 }
82
83 /**
84  * Enlève l'objet {el} avec une transition de {speed}
85  * @param el
86  * @param speed
87  */
88 function removeFadeOut( el, speed ) {
89     var seconds = speed/1000;
90     el.style.transition = "opacity "+seconds+"s ease";
91
92     el.style.opacity = 0;
93
94     setTimeout(function() {
95         el.parentNode.removeChild(el);
96     }, speed);
97
98 }
99
100 /**
101  * Charge Le Loader spinner pendant Le Login
102  */
103 function chargement_icone() {
104     if(loader) {
105         document.getElementById("loader").style.visibility="hidden";
```

```
177     document.getElementById("modal-title").innerHTML = etat;
178
179     modal.style.display = "block";
180
181
182     span.onclick = function() {
183         modal.style.display = "none";
184     }
185
186     window.onclick = function(event) {
187         if (event.target == modal) {
188             modal.style.display = "none";
189         }
190     }
191 }
192
193
194 /**
195  * Fonction créant dynamiquement les tables de progressions
196  * @param projet_id
197  */
198 function progression(projet_id) {
199
200     let etat_archivage = {
201         recuperation_projet: 'Récupération du projet',
202         recuperation_bdd: 'Récupération de la base de données',
203         envoi_ovh : 'Envoi au cloud OVH',
204     }
205
206
207     // create ul element and set its attributes.
208     let ul = document.createElement('UL');
209     ul.setAttribute('id', 'list-progression');
210     ul.setAttribute('class', 'list-group list-group-flush');
```