SIMPLON.CO

Projet Chef d'œuvre

Pour la validation du titre professionnel « Développeur Data IA »

SARLAT Ludivine
07/06/2021

SIGMA

Table des matières

1. Introduction ..... - 2 -
1.1. Présentation de l'entreprise accueil ..... - 2 -
1.2. Cadre de réalisation du projet Chef-Œuvre ..... - 2 -
2. Démarrage du projet ..... - 3 -
2.1. Compréhension du besoin client ..... - 3 -
2.2. Etat de l'art ..... - 3 -
2.3. Eléments de conception technique ..... - 4 -
2.4. Choix techniques liés au projet ..... - 4 -
3. Réponse finale apportée - ce qui a été réalisé ..... - 5 -
3.1 Collecte des données ..... - 5 -
3.2 – Analyse exploratoire des données de sauvegarde ..... - 6 -
3.3 Consolidation des données ..... - 8 -
3.4 Base de données relationnelle ..... - 10 -
3.5 Création d'un modèle IA – Prédiction ..... - 13 -
3.51 Approche classique : ..... - 13 -
3.5.2 Autre orientation pour la modélisation ..... - 14 -
3.5.3 Requêtage de la base pour la modélisation ..... - 15 -
4. Mise en œuvre du projet - Mise à disposition ..... - 16 -
4.1. Organisation technique et l'environnement de développement tout au long de la production - 16 -
4.1.1 Premier livrable (décembre 2020) ..... - 16 -
4.1.2 Second livrable (avril 2021) ..... - 17 -
4.1.3 Ajout d'une fonctionnalité supplémentaire ..... - 21 -
4.2. Gestion de projet ..... - 22 -
4.3. Retours d'expérience sur les outils, techniques et compétences à l'œuvre tout au long du projet - 23 -
4. Bilan du projet et les améliorations envisageables. ..... - 24 -
5. Conclusion. ..... - 24 -
ANNEXES ..... - 25 -

1. Introduction

1.1. Présentation de l'entreprise accueil

Le projet présenté a été réalisé lors d'un contrat d'alternance au sein du Groupe Sigma.

Le Groupe Sigma est une Entreprise de Services du Numérique, spécialisée dans l'édition de logiciels, l'intégration de solutions digitales sur mesure, l'externalisation de systèmes d'information et les solutions cloud. Il compte environ 800 collaborateurs, répartis sur 5 sites dont la majorité est sur le site de La-Chapelle-sur-Erdre. Le groupe est propriétaire de 3 datacenters et compte environ 2200 clients.

Le Groupe Sigma conçoit et développe des solutions logicielles pour répondre aux besoins des Directions des Ressources Humaines et des Directions Administratives et Financières. Il réalise également des solutions logicielles métiers pour la Supply Chain, les Etablissements Consulaires, le Logement Social, la Grande Distribution ; et intervient sur des projets web « B2C » et « B2B » et conçoit des outils métiers. Il accompagne ses clients depuis la phase d'audit jusqu'à la mise en production et maintenance applicative.

Le Groupe Sigma propose à ses clients des offres sur mesure : solutions hébergement, cloud et infogérance avec un support 24/7 et 365/365. Ses collaborateurs accompagnent leurs clients sur la transformation du Système d'Information avec une gamme complète de prestations. Son activité infogérance est certifiée ISO 9001, ISO 27001 et Hébergeur Agréé de Données de Santé (HADS).

1.2. Cadre de réalisation du projet Chef-Œuvre

Le projet que je vais présenter est identifié par l'intitulé « Capacity Planning » et est réalisé en étroite collaboration avec un autre alternant de Simplon. C'est un projet interne à l'entreprise commandité par le service d'Infogérance. Il fait suite à un audit réalisé en janvier 2020 par le cabinet parisien « IA Builders » qui a identifié des cas d'usages pour l'utilisation et de mis à profit d'outils d'intelligence artificielle.

L'audit a mis en évidence plusieurs volets sur lesquels l'IA pouvait avoir une valeur ajoutée importante : notamment sur la gestion des ressources matérielles. En effet, le service d'Infogérance doit anticiper les besoins en matériel (serveurs) pour assurer les services selon les besoins de ses clients. Ces besoins matériels sont étudiés au moment de l'achat qui a lieu en fin d'année. Il est important d'avoir une vision ni trop peu, ni trop important pour ce matériel de sorte à pouvoir assurer le bon fonctionnement des plateformes tout en n'achetant pas du matériel qui ne servirait pas avant l'année si l'achat s'avérait trop important.

Ces achats de matériel doivent être envisagés pour différents services : Service SAUVEGARDE et Service STOCKAGE. Le service SAUVEGARDE propose aux clients de sauvegarder leurs données informatiques régulièrement pour s'assurer que rien ne soit perdu en cas d'incidents majeurs (possibilité de récupérer la dernière sauvegarde). Le service de STOCKAGE concerne le stockage de données plus statiques des clients (comme un service Cloud). Ces 2 services sont des cas d'usage pour lesquels il est nécessaire de réaliser des prévisions sur les besoins clients. Le projet présenté ne s'intéressera qu'à la partie SAUVEGARDE, même si la partie STOCKAGE a été en partie menée en parallèle.

2. Démarrage du projet

2.1. Compréhension du besoin client

Le client, c'est à dire le service Infogérance, a besoin de prévoir ses besoins en matériel physique afin de prévoir ses achats en fin d'année pour l'année à venir. L'achat groupé du matériel lui permet d'avoir un prix plus intéressant et d'éviter la multiplication des commandes au cours de l'année. L'objectif est d'être au plus près des besoins de ses clients pour en acheter ni trop peu, au risque d'en acheter à la dernière minute au prix fort, ni de trop car le matériel non utilisé l'année suivante ne sera alors utilisé que l'année N+2. Ce qui se révèle être une mauvaise pratique connaissant l'obsolescence importante du matériel informatique. Les commandes sont, en général, programmées en novembre de l'année pour l'année à venir. Ceux sont les gestionnaires de plateforme (GPF) qui évaluent les besoins en matériel car ceux sont eux qui assure la supervision des plateformes. L'objectif de ce projet est d'aider ces gestionnaires dans cette tâche qui seront les utilisateurs de notre solution finale.

La phase de cadrage a réuni plusieurs acteurs de l'entreprise lors des ateliers suivants :

- un atelier de Design pour la compréhension de la situation actuelle et la définition du processus cible. Cela a permis de comprendre les missions du gestionnaire de plateforme, ses activités quotidiennes et les enjeux de son métier.

- un atelier de Questions/Cibles pour la formalisation des cibles analytiques et des objectifs de modélisation. Cela a permis de cibler les besoins de l'utilisateur, d'avoir des objectifs qualitatifs pour construire le projet d'IA.

- un atelier Data discovery concernant la compréhension des données existantes, la disponibilité et la qualité des données et leur gouvernance. Celui-ci a permis de prendre connaissance des différents outils à disposition de l'équipe IA et des problèmes liés à l'historisation des données par exemple.

- un atelier architecture pour l'analyse de l'environnement technique existant et l'exigence de sécurité. Différentes solutions sont envisagées qui seront réévaluées au cours du projet en fonction du volume de données à manipuler.

- la validation du cadrage. Le projet a été validé et a été présenté régulièrement à un comité de pilotage.

2.2. Etat de l'art

Comment se fait l'achat de matériel physique pour l'année suivante ?

Cette tâche est réalisée par le gestionnaire de plateforme. Ses missions quotidiennes sont d'utiliser des outils de supervisions, pour contrôler le bon fonctionnement des plateformes. Il consacre un temps conséquent au déplacement de serveurs virtuels sur les différentes baies de sauvegarde (machines physiques). L'objectif est de ne pas avoir de baie saturée car cela génèrerait un incident technique entrainant une coupure du service.

Pour prévoir les achats matériels, le gestionnaire de plateforme doit réunir des données issues de plusieurs outils de supervision. Cela lui demande plusieurs journées de travail, sans compter les missions intermédiaires (mensuelles) qui lui demandent de savoir si la volumétrie de l'ensemble des serveurs dispose de suffisamment de ressources pour ne pas entrainer un incident.

Cette tâche est identifiée comme une situation pouvant être assistée par l'IA. Notre objectif est d'accompagner le gestionnaire de plateforme (utilisateur cible) dans l'évaluation de ses besoins en volumétrie et donc en matériel physique.

2.3. Eléments de conception technique

Les données disponibles :

La plateforme de sauvegarde utilise les 2 outils de supervision NETBACKUP (vouée à disparaitre) et VEEAM (introduite en novembre 2017). Les données de ces outils sont visualisables via l'outil NAGIOS, outil consulté tous les jours par le gestionnaire de plateforme (GPF).

L'outil KIBANA est utilisé par le groupe Sigma et est une couche de visualisation de données pour ELASTICSEARCH. Cet outil permet de visualiser les données de NAGIOS sur un historique de 2 ans. Il a été envisagé d'utiliser ces données mais l'historique s'est révélé insuffisant pour pouvoir faire du machine learning. La recherche des logs antérieurs de cet outil s'est arrêtée au fait qu'une purge des logs s'opérait sur les éléments datant de plus de 2 ans.

Les CRMA (compte-rendu mensuel automatisé) sont des rapports automatisés au format EXCEL générés par un script shell, rapport dont une copie est déposée en début de mois sur les répertoires partagés GSDATA (répertoires internes à Sigma). Ils offrent une vision détaillée par client des serveurs occupés et de leurs volumétries pour les outils Netbakup et Veeam.

Le logiciel SDX (solution Microsoft - Dynamics AX) est utilisé comme ERP (*Enterprise Resource Planning)* chez Sigma depuis 2009 et répertorie les données relatives aux clients et aux fournisseurs de l'entreprise. La partie client/prospect a basculé en 2014 sur le logiciel SALESFORCE et a été envisagée comme source de données. Ce logiciel permet d'enregistrer les clients avant que ceux-ci ne signent de contrats, en tant que prospects. Ces données via les CUBES auraient pu permettre d'avoir une vision anticipée sur les futurs clients susceptibles de bénéficier du service de sauvegarde. La partie facturation SDX nous donne des données sur le chiffre d'affaires et les échéanciers de facturation pour les clients. Plusieurs contraintes dans l'utilisation de ces données ont été identifiées comme le fait que la nomination des clients n'étaient pas cohérentes d'un outil à un autre.

Il a été convenu que les données utiles au projet seraient issues des CRMA.

2.4. Choix techniques liés au projet

Etape de récupération des données – Travail exploratoire

Notre formation Simplon nous ayant donné PYTHON comme langage de programmation pour la manipulation des données et la modélisation, nous avons démarré le projet avec ce langage dans un environnement JUPYTER Notebook. Si ce langage se révèle être adapté au machine learning, une petite difficulté est que ce langage n'était pas encore utilisé par les membres de notre équipe.

Le travail a été réalisé sur Notebook avec la librairie PANDAS adaptée à la manipulation de Dataframe.

La visualisation des données et les transformations avec des méthodes déjà connues s'est aussi faite sur Notebook avec des librairies adaptées (SEABORN, MATPLOTLIB).

Dans un premier temps, la modélisation a été faite sur Notebook grâce à la librairie SCIKIT-LEARN et le modèle choisi a été mis à disposition via un classeur Excel.

Dans ce même temps, une machine virtuelle (VM) a été créée grâce à une ressource Microsoft AZURE, VM sur laquelle le premier livrable (classeur EXCEL) a été déposé et partagé avec les gestionnaires de plateforme.

Mise en base – Modèle – Mise à disposition

Dans un deuxième temps, une VM propre à SIGMA a été créée pour se passer des ressources Microsoft Azure. Le classeur EXCEL (premier livrable) a alors été déposé sur cette VM et partagé avec le gestionnaire de plateforme.

L'étape suivante a été un travail de mise en base de données. Le choix de PostgreSQL a été fait car il semblerait que ce soit une solution plus stable que MySQL, plus adaptée au monde de l'entreprise et déjà utilisée par l'entreprise SIGMA.

Une fois cette mise en base réalisée, le modèle a été réentrainé via un Notebook. Il a été mis à disposition via une interface utilisateur développée en ANGULAR-TYPESCRIPT (second livrable). Ce choix de framework pour le front-end a été fait car ce sont des outils maitrisés par les collaborateurs Sigma. Ce choix en termes de front-end s'est accompagné de la mise en place d'un serveur Node.js dans sa version EXPRESS pour la partie Back-end. Cette version EXPRESS minimaliste et simpliste est suffisante pour le projet tel qu'il est pensé.

La mise à jour des données en base est assurée par le planificateur de tâches Windows (sur la VM SIGMA) qui exécute un script Python.

Le service WEB accessible à l'utilisateur final (GPF) donnant accès à l'application est déployé via le service IIS de Windows (de la VM SIGMA).

L'application web permet d'avoir accès au modèle de prédiction de la volumétrie pour l'année à venir mais aussi à l'historique des données des plateformes (SAUVEGARDE et STOCKAGE) et permet de télécharger les données au format CSV.

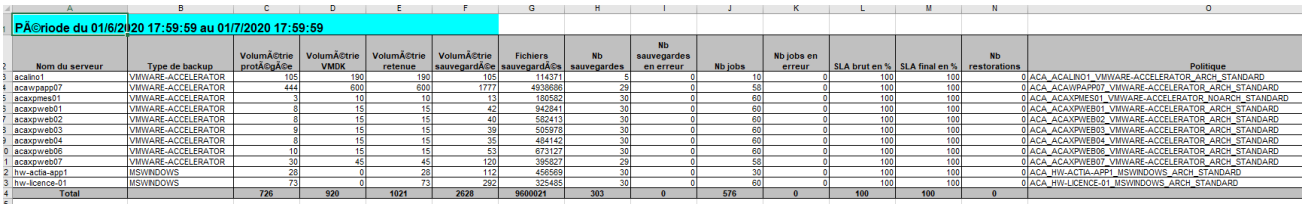
3. Réponse finale apportée - ce qui a été réalisé

3.1 Collecte des données

Les données choisies pour le travail de modélisation se présentent sous la forme de rapports mensuels générés automatiquement et déposés sur un répertoire interne à Sigma. Ils sont au format EXCEL et présentent une feuille par client énumérant les serveurs de ce client et des données comme la volumétrie protégée occupée par chaque serveur(Fig 1 à 3).

Ces classeurs sont générés pour l'outil NetBackup et pour l'outil Veeam dans des répertoires séparés. Veaam est un outil introduit en novembre 2017 et est voué à remplacer totalement NetBackup utilisé depuis février 2015. Un premier travail d'agrégation a été nécessaire. L'union des données de ces 2 outils a été possible en créant des colonnes communes. Le choix des colonnes s'est fait en concertation avec le gestionnaire de plateforme et s'est finalement limité au nom du serveur, la date du classeur Excel, la volumétrie protégée occupée par chaque serveur, la politique associée (ou job pour Veeam), le client (s'il est renseigné) et l'outil associé (Netbackup ou Veeam).

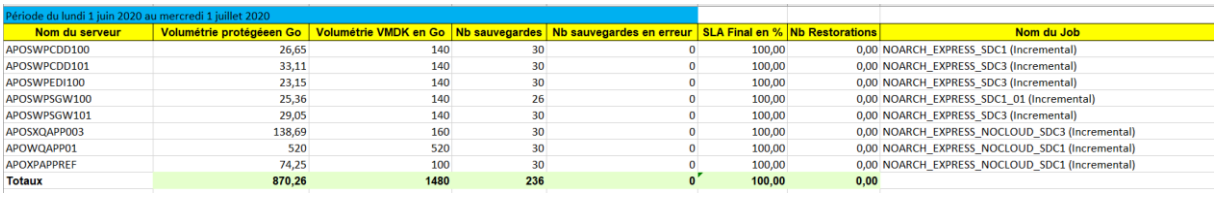
NETBACKUP



PÃ©riode du 01/6/2020 17:59:59 au 01/7/2020 17:59:59														
Nom du serveur	Type de backup	VolumÃ©trie protÃ©gÃ©e	VolumÃ©trie VMDK	VolumÃ©trie retenue	VolumÃ©trie sauvegardÃ©e	Fichiers sauvegardÃ©s	Nb sauvegardes	Nb sauvegardes en erreur	Nb jobs	Nb jobs en erreur	SLA brut en %	SLA final en %	Nb restorations	Politique
acalno1	VMWARE-ACCELERATOR	105	190	190	105	114371	5	0	10	0	100	100	0	ACA_ACALNO1_VMWARE-ACCELERATOR_ARCH_STANDARD
acawpapp07	VMWARE-ACCELERATOR	444	600	600	1777	4938686	29	0	58	0	100	100	0	ACA_ACAWPAPP07_VMWARE-ACCELERATOR_ARCH_STANDARD
acaxpmes01	VMWARE-ACCELERATOR	3	10	10	13	180582	30	0	60	0	100	100	0	ACA_ACAXPMES01_VMWARE-ACCELERATOR_NOARCH_STANDARD
acaxpweb01	VMWARE-ACCELERATOR	8	15	15	42	942841	30	0	60	0	100	100	0	ACA_ACAXPWEB01_VMWARE-ACCELERATOR_ARCH_STANDARD
acaxpweb02	VMWARE-ACCELERATOR	8	15	15	40	582413	30	0	60	0	100	100	0	ACA_ACAXPWEB02_VMWARE-ACCELERATOR_ARCH_STANDARD
acaxpweb03	VMWARE-ACCELERATOR	9	15	15	39	505978	30	0	60	0	100	100	0	ACA_ACAXPWEB03_VMWARE-ACCELERATOR_ARCH_STANDARD
acaxpweb04	VMWARE-ACCELERATOR	8	15	15	35	484142	30	0	60	0	100	100	0	ACA_ACAXPWEB04_VMWARE-ACCELERATOR_ARCH_STANDARD
acaxpweb06	VMWARE-ACCELERATOR	10	15	15	53	673127	30	0	60	0	100	100	0	ACA_ACAXPWEB06_VMWARE-ACCELERATOR_ARCH_STANDARD
acaxpweb07	VMWARE-ACCELERATOR	30	45	45	120	395827	29	0	58	0	100	100	0	ACA_ACAXPWEB07_VMWARE-ACCELERATOR_ARCH_STANDARD
hw-actia-app1	MSWINDOWS	28	0	28	112	456569	30	0	30	0	100	100	0	ACA_HW-ACTIA-APP1_MSWINDOWS_ARCH_STANDARD
hw-licence-01	MSWINDOWS	73	0	73	292	325485	30	0	60	0	100	100	0	ACA_HW-LICENCE-01_MSWINDOWS_ARCH_STANDARD
Total		726	920	1021	2628	9600021	303	0	576	0	100	100	0	

Figure 1 : Données brutes sous Excel de l'outil Netbackup.

VEEAM



Période du lundi 1 juin 2020 au mercredi 1 juillet 2020							
Nom du serveur	Volumétrie protégéeen Go	Volumétrie VMDK en Go	Nb sauvegardes	Nb sauvegardes en erreur	SLA Final en %	Nb Restorations	Nom du Job
APOSWPCDD100	26,65	140	30	0	100,00	0,00	NOARCH_EXPRESS_SDC1 (Incremental)
APOSWPCDD101	33,11	140	30	0	100,00	0,00	NOARCH_EXPRESS_SDC3 (Incremental)
APOSWPEDI100	23,15	140	30	0	100,00	0,00	NOARCH_EXPRESS_SDC3 (Incremental)
APOSWPSGW100	25,36	140	26	0	100,00	0,00	NOARCH_EXPRESS_SDC1_01 (Incremental)
APOSWPSGW101	29,05	140	30	0	100,00	0,00	NOARCH_EXPRESS_SDC3 (Incremental)
APOSXQAPP003	138,69	160	30	0	100,00	0,00	NOARCH_EXPRESS_NOCLOUD_SDC3 (Incremental)
APOWQAPP01	520	520	30	0	100,00	0,00	NOARCH_EXPRESS_NOCLOUD_SDC1 (Incremental)
APOXPAPPREF	74,25	100	30	0	100,00	0,00	NOARCH_EXPRESS_NOCLOUD_SDC1 (Incremental)
Totaux	**870,26**	**1480**	**236**	**0**	**100,00**	**0,00**	

Figure 2 : Données brutes sous Excel de l'outil Veeam.

Voici le résultat de l'agrégat :

serveur	date	vol_pro	politique	client	outil
2008r2_template	2020-11-01	14.82	SIF_NOARCH_EXPRESS_SDC1 (Incremental)	SIG France	VEEAM
2008r2_template	2020-12-01	14.82	SIF_NOARCH_EXPRESS_SDC1 (Incremental)	SIG France	VEEAM
2012r2_template	2020-11-01	50.00	SIF_NOARCH_EXPRESS_SDC1 (Incremental)	SIG France	VEEAM
2012r2_template	2020-12-01	50.00	SIF_NOARCH_EXPRESS_SDC1 (Incremental)	SIG France	VEEAM
2016_en_template	2020-11-01	100.00	SIF_NOARCH_EXPRESS_SDC1 (Incremental)	SIG France	VEEAM

Figure 3 : Dataframe des données de sauvegarde (Netbackup et Veeam)

Il est apparu qu'un même serveur pouvait apparaitre plusieurs fois au cours d'un mois (sur Veeam et Netbackup mais aussi sur le même outil). Ces doublons sont liés au fait qu'un serveur peut être présent sur les 2 outils en même temps le temps du basculement d'une plateforme à l'autre. Afin de ne pas avoir de souci ultérieurement, il a été convenu avec le gestionnaire plateforme que les enregistrements doubles soient agrégés en les sommant.

3.2 – Analyse exploratoire des données de sauvegarde

Au moment de la collecte des données, il y avait 4774 serveurs distincts pour environ 125 661 enregistrements sur un historique de 71 mois.

Une visualisation de la volumétrie globale pour l'ensemble des serveurs permet de voir l'évolution globale de la plateforme sauvegarde (Fig 4).

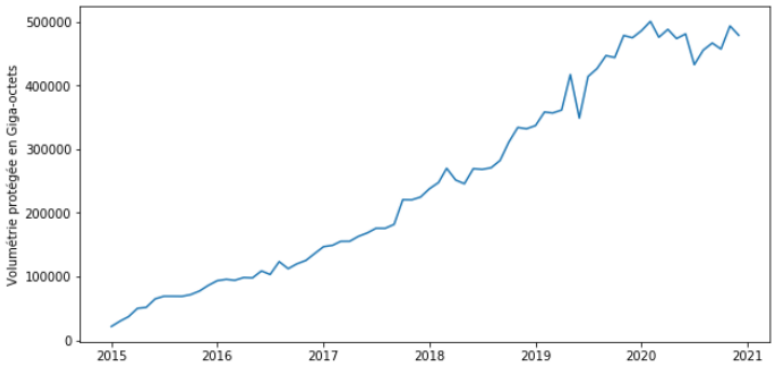


Figure 4 : Evolution de la volumétie protégée pour l'ensemble des serveurs de sauvegarde.

L'évolution semble globalement croissante, des variations plus ou moins importantes sont visibles et correspondent à des évènements plus ou moins connus. Exemple : Novembre 2017 arrivée de l'outil VEEAM.

L'exploration de ces données nous a amenés à enrichir ces données :

- Nous avons enrichi les données avec la création d'une colonne « Offre », à partir de la colonne « Politique ». Effectivement la majorité des politiques contiennent les termes « standard », « express » ou « no_policy ». Cette offre peut être STANDARD, EXPRESS, NO_POLICY ou Inconnu.

Nous pouvons visualiser le type d'offre à laquelle les clients ont souscrit sur le diagramme suivant (Fig 5) :

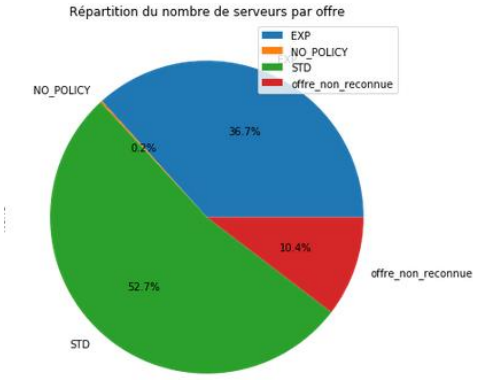


Figure 5 : Répartition des serveurs selon l'offre associée

La majorité des serveurs sont associés à une offre « standard ».

De plus, l'offre associée à un serveur détermine le nombre de jours de rétention, le nombre de sauvegardes effectuées dans le mois et les règles de déduplication[1] (règles propres au métier).

Offre EXPRESS : 7 jours de rétention soit 13 jours glissants avec un taux de déduplication de 1:6. La volumétrie brute conservée sur notre stockage = V x 2 + (V x 0,1 x 11).

[1] La déduplication permet de réduire l'espace mémoire utilisée par des copies de données qui présentent des similitudes.

Offre STANDARD : 31 jours de rétention soit 37 jours glissants avec un taux de déduplication de 1:9. La volumétrie brute conservée sur notre stockage = V x 5 + (V x 0,1 x 29).

- Connaitre l'offre associée à chaque serveur a une grande importance car cela permet de calculer la volumétrie « réellement » occupée par le serveur : « Volumétrie stockée ». Nous avons donc enrichi les données en créant une colonne « Vol_stockée ».

Les règles métiers sont les suivantes :

EXPRESS : Vol_stockée = Vol_protégée * 7,11

STANDARD : Vol_stockée = Vol_protégée * 2,66

NO_POLICY : Vol_stockée = Vol_protégée

3.3 Consolidation des données

Très rapidement, il a été évoqué par le gestionnaire de plateforme que la typologie des serveurs, c'est-à-dire le type de données hébergées par les serveurs avait aussi de l'importance. Par exemple, les serveurs hébergeant des fichiers sont généralement plus volumineux que des serveurs de type applicatif (APP) ou des serveurs de type base de données (DAT).

Nous avons donc cherché à connaitre le type de données hébergées par les serveurs. Pour cela, nous avons réalisé 3 étapes :

- A partir d'un classeur Excel (fourni par l'équipe métier – Fig 6) répertoriant les serveurs selon leur nom et précisant le type de données hébergées sur chacun d'eux (nombre d'applications). Si un serveur héberge en majorité des données de type base de données, alors sa typologie sera DAT (base de données).

Nombre de profil ?	Étiquettes de colonne						
Étiquettes de lignes	AD	appli	base de données	DNS	Messagerie	presentation	virtualisation	(vide)	Total général
mutsxpdat100.srv.sigma.host		1	56						57
mutsxpdat101.srv.sigma.host			54						54
mutsxpdat002.srv.sigma.host		1	40						41
mutxpdat02.dmz.sigma.fr			39						39
mutxpdat01.dmz.sigma.fr			36						36
mutsxpdat001.srv.sigma.host			36						36

Figure 6 : Tableau indiquant le nombre d'application par type de données hébergées pour les différents serveurs.

- Puis nous avons cherché dans le nom même du serveur sa typologie.

APP : app, APP, … type applicatif.

WEB : WEB, Web, web… de type WEB.

DAT : DAT, dat, SQL, sql, … de type base de données.

FIC : FIC, fic, …de type fichiers.

A l'issue de ces 2 étapes, ils restaient encore un grand nombre de serveurs dont nous ne connaissions pas la typologie (environ 37%) (Fig 7); ce qui est un souci pour l'analyse de l'évolution des données en fonction de leur typologie.

type	Nb_Serveurs	Pourcentage
AD	72	1.508169
APP	1706	35.735233
DAT	614	12.861332
DNS	15	0.314202
FIC	133	2.785924
INCONNU	1799	37.683284
MET	31	0.649351
VIR	26	0.544617
WEB	378	7.917889

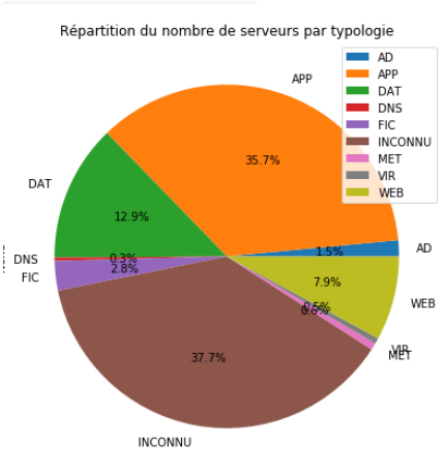


Figure 7 : Répartition des serveurs selon leur typologie

En analysant la volumétrie protégée totale des serveurs selon leur typologie, nous pouvons voir que les serveurs FIC sont moins nombreux mais représentent le plus gros volume de données (Fig 8.1 et 8.2).

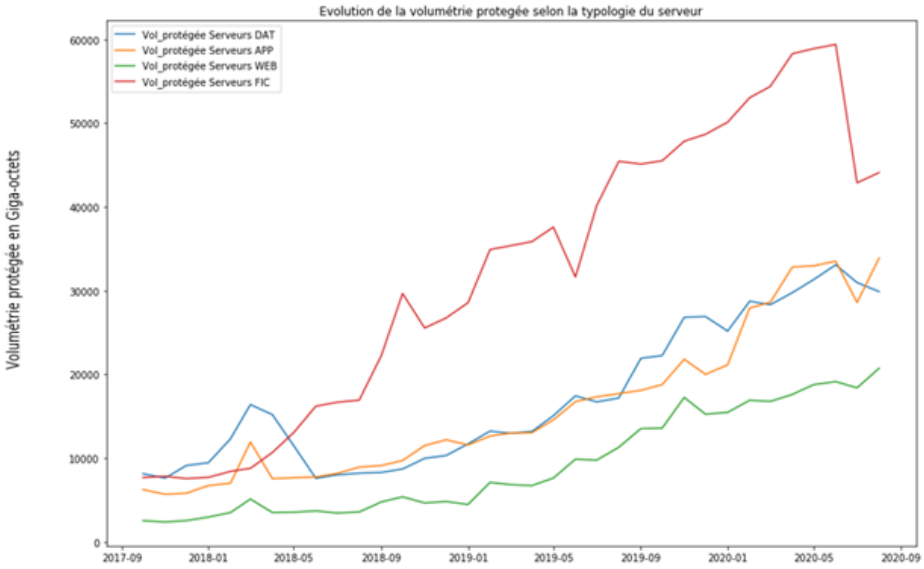


Figure 8.1 : Graphiques représentant la volumétrie protégée

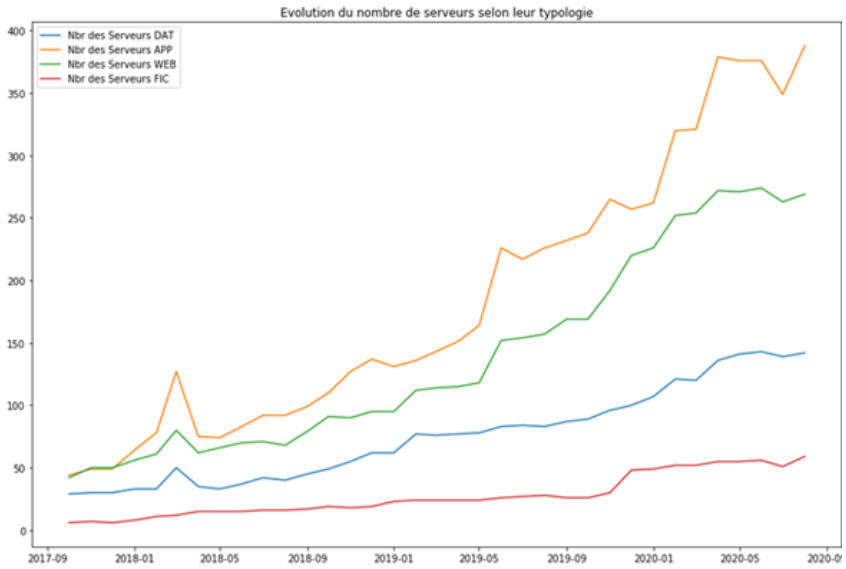
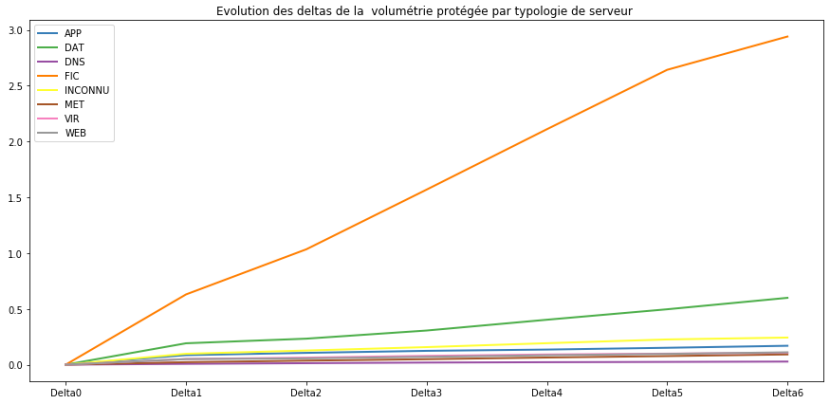


Figure 8.2 : Evolution du nombre de serveurs par typologie

Les analyses complémentaires sur les serveurs dont la typologie est connue ont pu mettre en évidence que les serveurs de type FIC (fichiers) avaient une évolution différente des autres serveurs (Figure 9). En effet, nous pouvons observer que les serveurs de type Fichiers ont une croissance très importante par rapport aux autres types sur leurs 6 premiers mois de vie.



Les deltas sont le rapport entre la différence entre la volumétrie d'un mois et la volumétrie du premier mois de vie (vol6) sur ce premier mois.

delta6 = (vol1 - vol6) / vol6

delta5 = (vol2 - vol6) / vol6

…

delta0 = (vol6 – vol6) / vol6

Figure 9 : Graphique des deltas cumulés sur les 6 premeiers ois de vie.

Afin de simplifier les analyses, nous avons décidé, en concertation avec le gestionnaire de plateforme, de ne travailler qu'avec 3 typologies différentes en assignant la typologie APP aux serveurs initialement reconnus comme APP, AD, WEB, MET, DNS, VIR.

- une 3 ième étape pour déterminer la typologie des serveurs inconnue a été mise en place. Pour se faire, nous avons décidé d'utiliser un algorithme de classification des serveurs à partir de l'évolution de leurs volumétrie protégée sur les 3 premiers mois de vie. Connaissant l'évolution de près de 63% des serveurs et leur typologie, nous avons testé plusieurs modèles de classification. L'algorithme de classification (GradientBoostingClassifier) présentait les meilleurs résultats.

Voici la répartition des serveurs selon leur typologie une fois cette dernière étape appliquée (Fig 10) :

new_type	Nb_Serveurs	Pourcentage
APP	3760	81.899368
DAT	690	15.029405
FIC	141	3.071226

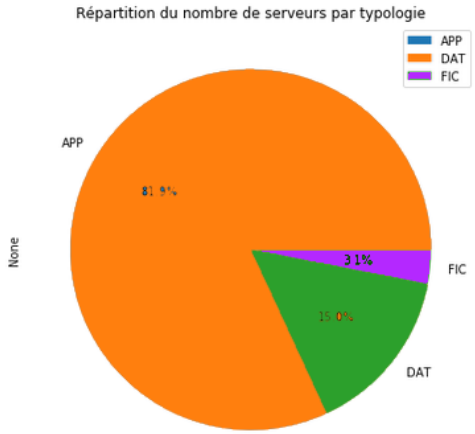


Figure 10:Repartition des serveurs selon leur typologie.

3.4 Base de données relationnelle

La base de données utilisée pour ce projet est une base de données PostgreSQL et a été créée dans un second temps c'est-à-dire après le premier livrable.

Les tables de la base de données ainsi que les vues nécessaires au projet ont été créées via un script PostgreSQL (donné en annexes). Le choix de cette structure de base de données a été fait de sorte que les données soient utilisées (manipulées) par les services/applicatifs et éviter d'être des données « mortes ». Ce qui a demandé de réaliser des transformations et des calculs préalables (Fig 11).

Des vues ont été créées afin de permettre au serveur d'APIs de récupérer directement des données structurées et adaptées aux besoins de l'application. Ces vues ont l'avantage d'être mise à jour dès que de nouveaux enregistrements sont insérés en base.

La table principale est svg_enregistrements et a pour clé étrangère nom_serveur de la table svg_serveurs. Cette table svg_enregistrements a les colonnes suivantes :

- id_enregistrement : entier auto-incrémenté
- nom_serveur : chaine de caractères (clé étrangère de la table svg_serveurs)
- date : date au format AAAA-MM-JJ de l'enregistrement du CRMA
- vol_protegee : nombre décimal correspondant à la volumétrie protégée occupée par le serveur à cette date
- vol_stockee : nombre décimal correspondant à la volumétrie stockée occupée par le serveur à cette date
- politique : chaine de caractères contenant la politique (standard/express/

no_policy,inconnue)

- client : chaine de caractères correspondant au client possédant ce serveur

Cette table pointe sur la table svg_serveurs via la clé étrangère nom_serveur.

La table svg_serveurs a les colonnes suivantes :

- nom_serveur : chaine de caractères correspondant au nom du serveur
- date_crea : date de création du serveur au format AAAA-MM-JJ
- date_maj : date de dernière mise à jour au format AAAA-MM-JJ
- idtypologie : numéro de la typologie du serveur dans la table svg_typologies (clé étrangère de la table svg_typologies)
- idoffre : numéro de l'offre dans la table svg_offres (clé étrangère de la table svg_offres)
- idoutil_sauvegarde : numéro de l'offre dans la table svg_outil_sauvegarde (clé étrangère de la table svg_outi_sauvegardes)

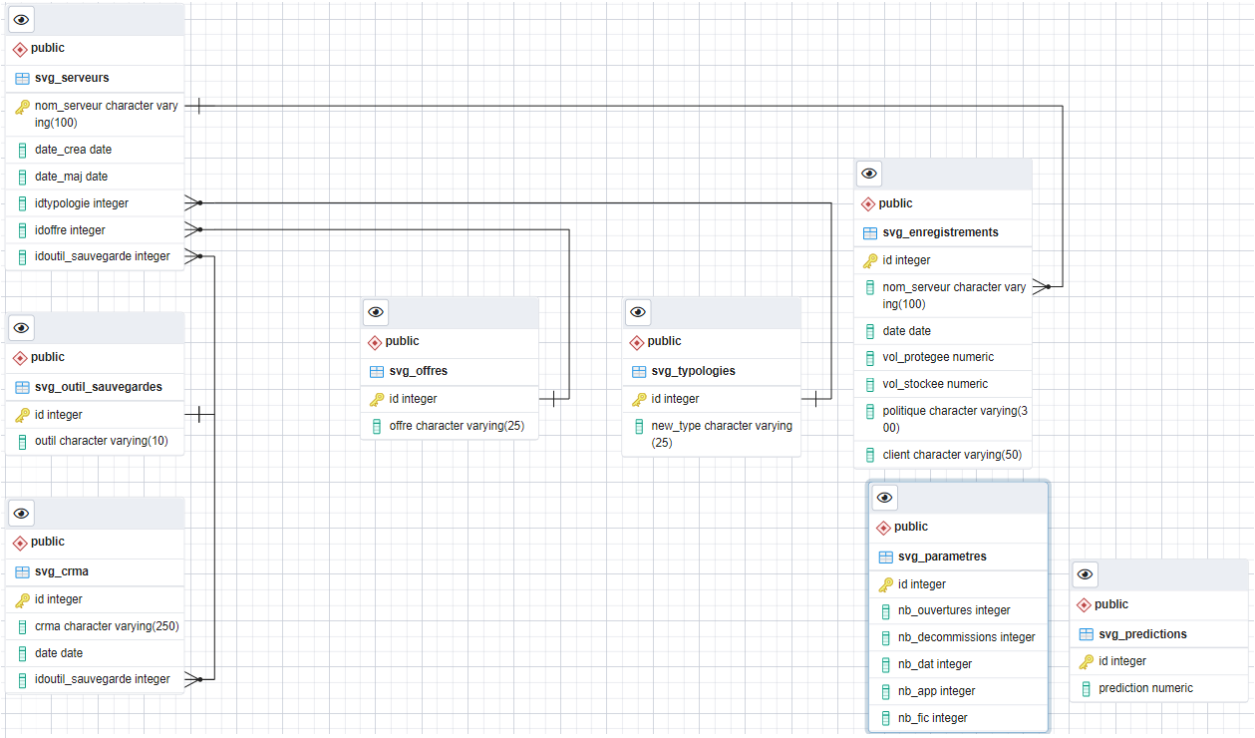


Figure 11 : Schéma des tables de la base de données relative au projet « Capacity planning ».

La table svg_typlogies contient les différentes typologies des serveurs ; la table svg_offres contient les différentes offres (standard/express/no_policy) et la table svg_outil_sauvegarde contient les différents outils de supervision (Netbackup/Veeam).

La table svg_crma contenant les fichiers (CRMA) insérés en base a été créé et permet de connaitre les CRMA déjà insérés en base de données, la date de ces compte rendu ; et a pour clé étrangère l'id de l'outil d'origine de la table svg_outil_sauvegarde.

L'**alimentation initiale** de la base de données a été réalisée via un script Python qui insère, dans les différentes tables, un dataframe issu les données brutes ayant subi des transformations préalables (recherche de l'offre associée au serveur, calcul de la volumétrie stockée, recherche de la typologie du serveur). Il est donné en annexes.

La **mise à jour** de la base est assurée via un script Python qui récupère les données des nouveaux CRMA disponibles en début de mois, effectue les transformations nécessaires et les insèrent en base de données. Ce script s'exécute automatiquement 4 fois par jour pour être sûre que les utilisateurs aient accès aux données le plus rapidement possible.

Pour le projet d'IA, dans sa seconde version (application ANGULAR), nous avons décidé de créer 2 tables supplémentaires qui facilitent la communication entre le serveur et le script python contenant le modèle d'IA. Ces tables sont mises à jour à chaque fois que l'utilisateur renseigne des valeurs dans l'application (svg_parametres) et à chaque fois qu'une prédiction est faite (svg_predictions). La table svg_paramètres contient les valeurs des 3 scénarii complétés par le gestionnaire de plateforme (nb_ouvertures, nb_decommissions, nb_dat, nb_app, nb_fic). La table svg_prédictions permet de stocker les prédictions

des 3 derniers scenarii simulés. Nous verrons plus en détail ces paramètres lors de la phase de modélisation.

3.5 Création d'un modèle IA – Prédiction

3.51 Approche classique :

Dans un premier temps, les données ont été étudiées sous leur forme de séries temporelles ; avec des modèles univariés. Les données sont en effet des enregistrements mensuels réguliers de volumétrie protégée par serveur. Ces données détaillées peuvent être agrégées pour obtenir une volumétrie totale mensuelle comme nous l'avons visualisée précédemment. L'historique des données étaient de 71 mois.

L'évolution de la volumétrie totale est sujette à des fluctuations, comme l'arrivée de nouveaux clients ; le départ ou changement de service souscrit par un client ; ou l'arrivée d'un nouvel outil de supervision. Les méthodes de lissage exponentiel permettent en général de gommer certaines de ces fluctuations. Des modèles de lissage exponentiel simple ou double (Holt et Holt_Winter) ont été testé (Fig 12).

La performance de ces modèles Holt_Winter est évaluée au travers de l'erreur absolue moyenne en pourcentage c'est-à-dire le MAPE (Mean absolute percentage error). Elle a été calculée en comparant les valeurs prédites et les valeurs réellement observées sur une portion de l'historique. Ces valeurs du MAPE ont été calculées pour chaque année et sont données en annexes.

Si la performance du modèle augmente au fur et à mesure des années, mettant en évidence que la prédiction est meilleure d'année en année, celle-ci diminue pour l'année 2020. En effet, il est possible de prédire l'évolution d'une variable si l'avenir se déroule de la même manière que dans le passé. Or 2020 ne semble avoir été une année particulière où l'activité du groupe SIGMA et de leurs clients a été impactés par la crise sanitaire. Les besoins clients, les nouveaux contrats, n'ont pas été conformes à ce qui se passe d'habitude. Ceci reflète les limites intrinsèques du modèle qui ne peut prévoir l'imprévisible.

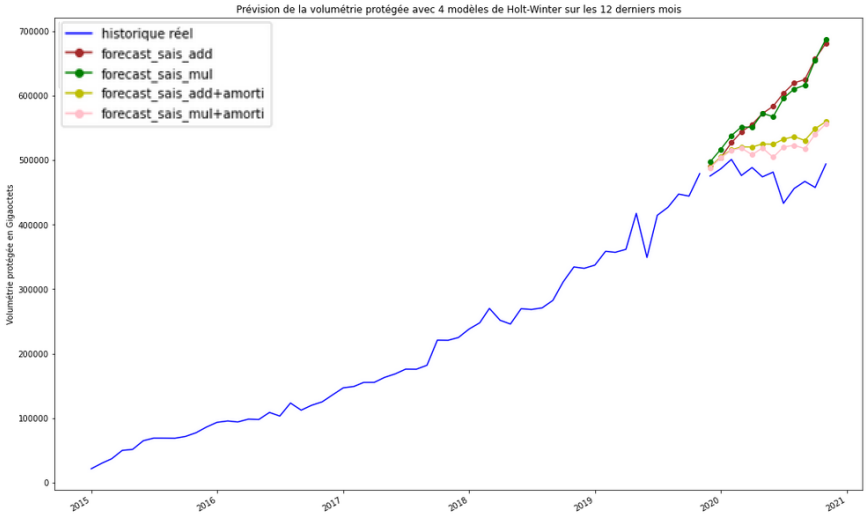
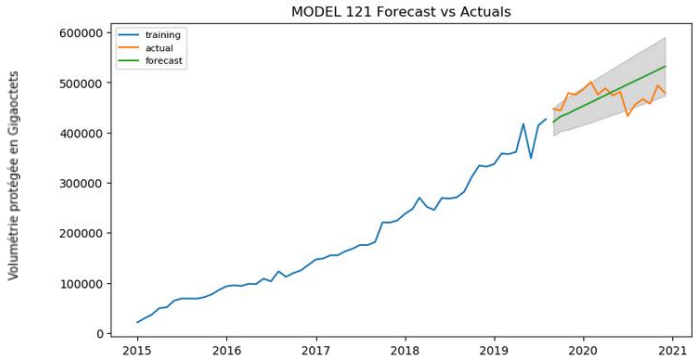


Figure 12 : Visualisation des prédictions des différents modèles Holt-Winter

D'autres modèles comme le modèle ARIMA ont été testés comme le modèle ARIMA(1.2.1) (Fig 13). Si le modèle de ARIMA (1.2.1) se révélait bon les premières années, les données réelles de 2020

sortent de l'intervalle de confiance de celui-ci. Ce modèle auto-regressif ne permet pas non plus de prévoir les chocs possibles comme le climat économique ou les crises sanitaires pouvant influencer l'évolution des besoins client et donc les données de sauvegarde. Ce modèle ne peut prédire l'évolution d'une variable que si l'avenir se déroule de manière identique au passé.



MAPE 2020 = 6.91%

Figure 13 : Visualisation des prédictions de volumétrie protégée avec ARIMA (1.2.1) pour l'année test (2020)

3.5.2 Autre orientation pour la modélisation

Afin de remédier à cette limite propre aux modèles précédents, l'équipe IA s'est orientée vers des modèles de régressions multiples, utilisant des indicateurs susceptibles d'avoir une influence sur l'évolution des données et sur lesquels il serait possible d'avoir la main. Ces indicateurs seraient des variables à part entière.

De plus, la problématique métier a aussi pu être résumée à la prédiction de la volumétrie maximale de l'ensemble des serveurs de la plateforme pour l'année à venir. C'est-à-dire qu'en novembre, nous voulons savoir quel sera le maximum de l'année N+1 à prévoir en termes de volumétrie protégée.

Nous avons cherché de nouveaux indicateurs susceptibles d'être liés à l'évolution de la volumétrie. Des indicateurs tels que le nombre de serveurs ouverts chaque année (nouveaux contrats) et le nombre de décommissions (départ de client, départ sur des serveurs dédiés…) ont été calculés pour l'ensemble de l'historique. Un modèle prenant en entrée les valeurs de la volumétrie totale de l'année en cours et les indicateurs pour l'année à venir a été testé.

A ces premiers indicateurs, l'équipe métier nous a aussi orienté vers la prise en compte du nombre de serveurs par typologie dans les prévisions de l'année à venir. En effet, les serveurs de type FIC (fichiers) sont en général plus volumineux et ont une croissance plus forte, ce qui impacte plus les besoins en ressources (comme nous l'avions observé lors de l'analyse exploratoire).

La communication entre les gestionnaires de plateforme et les avant-ventes, avant-ventes qui ont une vision sur l'arrivée de nouveaux contrats/clients, a aussi été évoquée. L'objectif est de faire en sorte que le gestionnaire de plateforme puisse avoir une vision plus précise sur l'arrivée de futurs clients et du profil de leurs données (plutôt FIC, APP ou DAT).

Nous avons donc testé des modèles de régressions multiples à plusieurs entrées et dont la cible était la volumétrie max N+1 (Figure 14):

- un modèle prenant en compte seulement la volumétrie totale des 10 premiers mois de l'année,
- un modèle intégrant en plus les ouvertures et les décommissions de l'année N+1,
- un modèle en intégrant en plus le nombre de serveurs par typologie de l'année N+1.

Variables explicatives

Variable cible

annee	janvier	fevrier	mars	avril	mai	juin	juillet	aout	septembre	octobre	nb_ouvertures	nb_decommissions	nb_dat	nb_app	nb_fic	Vol_max_N+1
2015	21489.00	30122.90	36869.40	49956.50	51488.60	64912.30	68990.00	68973.60	68833.40	71526.80	397.0	201.0	1237.0	252.0	35.0	135995.00
2016	93471.50	95547.50	94025.70	98419.60	97810.50	108790.00	103167.10	123362.20	112289.00	119838.70	502.0	168.0	1446.0	317.0	37.0	194462.77
2017	146903.00	149027.00	155291.00	155388.00	163121.00	168617.00	175904.00	175752.00	181943.00	194051.95	845.0	418.0	1977.0	449.0	64.0	334323.29
2018	199802.32	205030.39	219944.44	204551.39	198852.18	222240.85	219114.50	218703.18	221332.78	240307.62	1157.0	579.0	2606.0	526.0	109.0	478929.91
2019	337205.48	358591.92	357004.93	361646.74	417433.26	413643.60	414412.39	426839.60	447299.09	444158.14	567.0	627.0	2593.0	537.0	102.0	501074.48

Figure 14 : Jeu de données utilisé par les modèles de regressions multiples

Les modèles sont entrainés sur un ensemble d'échantillons (de 2015 à 2018) et les performances des modèles sont évaluées sur un ensemble de test (ici 2019). Le choix du meilleur modèle est fait en fonction de ses performances observées à travers une métrique. La métrique prise en compte pour ce projet est l'erreur absolue en pourcentage (=MAPE) et est égale à ((Y_true – y_pred) ÷ y_true) *100. Le modèle avec 15 entrées a été finalement choisi (les performances sont données en annexes).

Les valeurs des entrées du modèle n'étant pas du même ordre de grandeur (volumétrie en centaines de milliers versus nombre d'ouvertures en quelques centaines par exemple), nous avons dû appliquer une méthode de mise à l'échelle (scalarisation). Cette méthode permet de transformer les données de telle sorte que leur distribution aura une valeur moyenne de 0 et un écart type de 1. Une fois le schéma de scalarisation entrainé sur les données d'entrainement, il est enregistré au format pickle car il sera nécessaire de l'appliquer aux entrées d'un nouvel échantillon en vue d'une prédiction.

De même, une fois le modèle entrainé, nous l'avons sauvegardé au format pickle, de sorte qu'il soit réappelé par un programme dans le but de faire de nouvelles prédictions.

Ce programme est un script python développé afin de prendre en entrée les valeurs de volumétrie de janvier à octobre, un nombre d'ouvertures et de décommissions, et le nombre de serveurs par typologie pour l'année N+1 (via un csv paramètres) et de donner en sortir pour les prédictions du modèle (sous format csv).

3.5.3 Requêtage de la base pour la modélisation

Le travail de modélisation fait dans un premier temps via un Notebook (prenant en entrée des données au format csv) a été repris dans un second temps, c'est-à-dire lorsque les données ont été mises en base. Le notebook de création de l'IA a simplement été modifié de sorte qu'il récupère les données depuis la base de données PostgreSQL (Fig15). Après la connexion à la base, le code suivant permet de faire des requêtes SQL afin de récupérer les données pour entrainer le modèle.

```
query_read = "SELECT * FROM svg_v_vol_protegee_mois_colonne"

cur.execute(query_read)

raw_enregistrement = cur.fetchall()

conn.commit()

jeu = pd.DataFrame(raw_enregistrement, columns =["annee", "janvier","fevrier" ,"mars", "avril", "mai","juin","juillet","aout", "septembre","octobre","novembre","decembre"])

jeu['Vol_max_N+1']= jeu.max(axis=1).shift(-1)

jeu
```

annee	janvier	fevrier	mars	avril	mai	juin	juillet	aout	septembre	octobre	novembre	decembre	Vol_max_N+1
2015	21489.00	30122.90	36869.40	49956.50	51488.60	64912.30	68990.00	68973.60	68833.40	71526.80	77310.20	86071.90	135995.00
2016	93471.50	95547.50	94025.70	98419.60	97810.50	108790.00	103167.10	123362.20	112289.00	119838.70	125238.00	135995.00	194462.77
2017	146903.00	149027.00	155291.00	155388.00	163121.00	168617.00	175904.00	175752.00	181943.00	194051.95	192686.37	194462.77	334323.29
2018	199802.32	205030.39	219944.44	204551.39	198852.18	222240.85	219114.50	218703.18	221332.78	240307.62	334323.29	332176.98	478929.91
2019	337205.48	358591.92	357004.93	361646.74	417433.26	413643.60	414412.39	426839.60	447299.09	444158.14	478929.91	475292.92	501074.48
2020	486387.88	501074.48	476119.92	488488.10	474004.61	481358.51	432942.12	455933.33	466973.57	457516.67	493931.41	479220.62	NaN

Figure 15 : Jeu de données utilisé pour la modélisation

4. Mise en œuvre du projet - Mise à disposition

4.1. Organisation technique et l'environnement de développement tout au long de la production

Le travail a été réalisé en collaboration étroite avec un autre alternant Simplon. Le partage des fichiers de développement et de documentation a nécessité la mise en place d'un répertoire GitLab assurant la sauvegarde des versions successives du projet. Une formation interne a été assurée par des membres de l'équipe.

4.1.1 Premier livrable (décembre 2020)

Le premier livrable est un classeur EXCEL composé d'une feuille « data » (Fig 16) sur laquelle apparaissent les données de la plateforme sauvegarde de manière agrégée (volumétrie totale par mois).

HISTORIQUE DE DONNÉES

Détail de la **volumétrie protégée** en giga-octets de l'ensemble des serveurs sur la pateforme de Sauvegarde

Année	Janvier	Février	Mars	Avril	Mai	Juin	Juillet	Août	Septembre	Octobre	Novembre	Décembre
2015	21 489	30 123	36 869	49 957	51 489	64 912	68 990	68 974	68 833	71 527	77 310	86 072
2016	93 472	95 548	94 026	98 420	97 811	108 790	103 167	123 362	112 289	119 839	125 238	135 995
2017	146 903	149 037	155 291	155 388	163 121	168 617	175 904	175 752	181 943	220 857	220 615	224 905
2018	238 042	247 760	270 111	251 675	245 807	269 615	268 414	270 910	282 438	311 169	334 323	332 177
2019	337 205	358 592	357 005	361 647	417 433	349 026	414 412	426 840	447 299	444 158	478 930	475 293
2020	486 388	501 074	476 120	488 488	474 005	481 359	432 942	455 933	466 974	457 517	493 931	*en cours*

Figure 17 : Feuille 'data' du classeur EXCEL correspondant à l'historique de la plateforme Sauvegarde

Une deuxième feuille de ce livrable (feuille 'simulation') présente l'historique des indicateurs (nombre d'ouvertures, nombre de décommissions, nombre de serveurs DAT, nombre de serveurs APP et nombre de serveurs FIC) pour les années passées (Fig 17). Il est proposé au gestionnaire de plateforme de compléter un tableau présentant des cellules vides correspondant à de possibles indicateurs pour l'année à venir. 3 scénarii sont possibles simultanément.

SIMULATION POUR L'ANNEE PROCHAINE

Dans cette feuille, vous pouvez tester le modèle de prédiction en paramétrant le nombre de serveurs qui porraient être ouverts, décommissionnés durant l'année prochaine et le nombre de serveurs selon leur typologie pour l'année prochaine.
Pour rappel, vous avez les données des années antérieures.

Veuillez compléter les cases vertes pour tester le modèle et cliquer sur le bouton.

Historique

Année	Nb Ouverture	Nb Décommission	Nb DAT	Nb APP	Nb FIC	Prédiction Max N+1
2016	*389*	*193*	*235*	*1253*	*36*	*135 995*
2017	*643*	*162*	*337*	*1589*	*44*	*224 904*
2018	*701*	*415*	*450*	*2075*	*63*	*334 323*
2019	*888*	*319*	*522*	*2606*	*106*	*478 929*
2020	*309*	*555*	*515*	*2435*	*97*	*519 997*

	Paramètres des scénarii					valeur IA
Scénario	Nb Ouverture	Nb Décommission	Nb DAT	Nb APP	Nb FIC	Prédiction Max N+1
1	1000	0	1515	2435	97	562 147
2	100	0	515	2435	197	532 458
3	0	0	500	2600	120	520 310

Exécuter et afficher les scénarii

Figure 16:Feuille 'simulation' du classeur Excel proposant de compléter 3 scénarii.

Une fois, les cellules complétées, un bouton « Exécuter et afficher les scenarii » permet de lancer une macro développée en VBA (Visual Basic) et affiche les prédictions du modèle dans la colonne « prédiction max N+1 ». Concrètement, cette macro VBA inscrit dans un fichier au format csv « paramètres.csv » les valeurs des indicateurs complétées par le gestionnaire de plateforme ; puis exécute le programme python qui prenait en entrées les valeurs des paramètres « paramètres.csv » et les volumétries de janvier à octobre de l'année écoulée (autres cellules du classeur Excel). Ce programme python enregistre les prédictions dans un fichier au format csv « predictions.csv ». Une fois terminé, la macro récupère ces prédictions dans ce fichier csv et les affiche dans le tableau de paramétrage dans la colonne « prédiction max N+1 ».

Sur cette même feuille EXCEL, des graphes de visualisation sont mis à jour avec ces prédictions (Fig 18 : graphe1 pour l'ensemble de l'historique et graphe2 pour un zoom sur la dernière année).

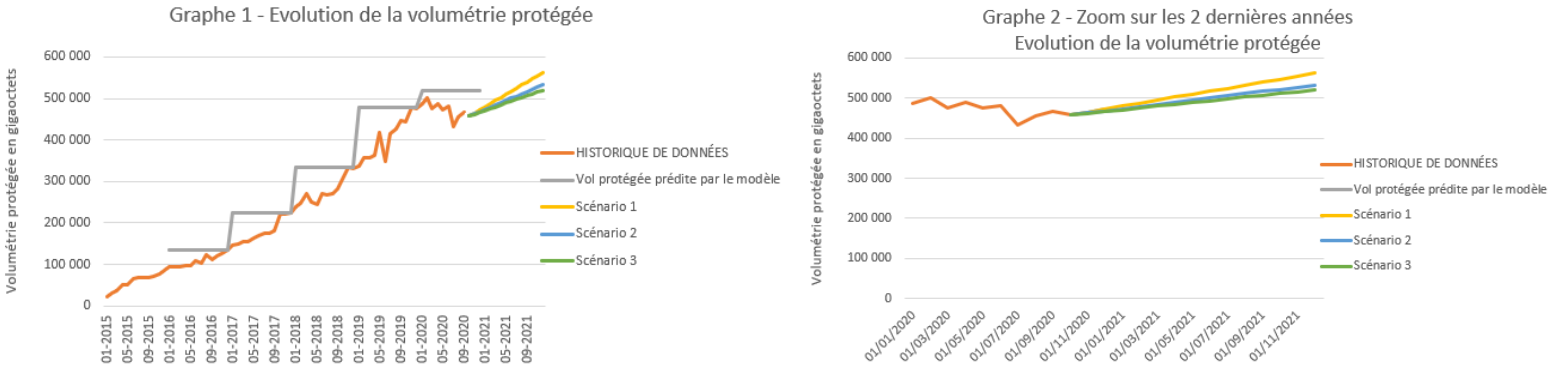


Figure 18 : Graphiques de visualisations des prédictions pour les 3 scénarii.

Le problème de ce premier livrable est qu'il demandait à l'utilisateur de travailler sur la VM sur laquelle le classeur EXCEL est déposé ; mais surtout, les données sont fixées, non mises à jour.

4.1.2 Second livrable (avril 2021)

Architecture du livrable final

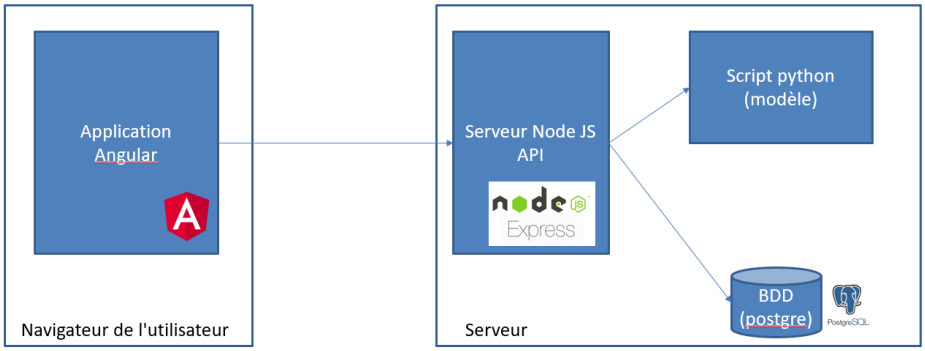


Figure 19 : Architecture du produit final

Cette architecture (Fig 19) a été choisie par l'équipe avec l'appui d'un architecte expert de l'équipe SIGMA. Ce livrable a demandé de monter en compétences sur les technologies ANGULAR et Node.js.

PARTIE BACK-END (coté Serveur)

Nous avons développé un serveur Node.js qui tourne en permanence sur la VM Windows SIGMA signwpapp009. Le serveur est accessible via les services Windows et peut être arrêté et redémarré grâce à ces mêmes services.

L'ensemble des services du projet sont configurés au niveau du fichier principal (index.ts) dans lequel les différents services API REST sont accessibles via une url signwpapp009:3000/nom_du_service. Il y a à la fois des services GET et des services POST.

Le leader technique nous a conseillé d'internaliser les paramètres de connexion à la base via des fichiers spécifiques (database.ini ; config.py ; connect.py).

Les services de type GET

Les méthodes GET interrogent la base de données (via des requêtes PostgreSQL sur les tables ou les vues) et renvoient des données sous format json à l'application qui les appelle.

- Service indicateur : permet d'envoyer au front la vue svg_v_indicateurs pour que le gestionnaire de plateforme puisse voir l'évolution réelle des indicateurs dans le passé.
- Service lastNbServeur : permet de récupérer le nombre de serveur total de l'année en cours (si on est en novembre ou décembre) ou de l'année précédente (si l'on est entre janvier et octobre). Cette valeur est importante dans le cas où l'utilisateur ne remplit pas les champs Nombre DAT, Nombre APP et Nombre FIC afin qu'elles soient calculées par défaut.
- Service vol_protegee : permet à partir de la vue svg_v_vol_protegee de définir deux listes : une liste de dates et une autre de vols (vol_protegée_total). Ces deux listes permettent de tracer le graphique de l'évolution de la capacité en fonction du temps.
- Service vol_protegee_colonne : permet d'extraire la vue svg_v_vol_protegee_mois_colonne. L'affichage de cette vue donne une vision agrégée de la volumétrie protégée totale par mois et par année.
- Service donneesbrutes : permet d'extraire la table svg_enregistrements. Un service permettant de télécharger cette table au format csv est définit coté client dans l'onglet Historique/sauvegarde.
- Service donneesbrutesStk : Service supplémentaire relatif au volet Stockage. Il permet d'extraire la table stk_enregistrements. De même un service permettant de télécharger cette table au format csv est définit coté client dans l'onglet Historique/Stockage.
- Service predictions : permet d'afficher une table représentant la jointure entre les paramètres et prédictions des 3 scenarios.

Les services de type POST

- Service predictions: est appelé en récupérant des paramètres en entrée (valeurs renseignées par l'utilisateur dans l'interface Angular (nombre d'ouvertures, nombre de décommissions, nombre de serveurs DAT, APP et FIC pour l'année N+1) et réalise les actions suivantes:
    - Met à jour la table svg_parametres à partir des valeurs renseignées par l'utilisateur.
    - Exécute le script python (contenant le modèle IA). Le script calcule des prédictions à partir des données de volumétrie et des paramètres saisis pour les 3 scénarii ; et les insère dans la table svg_predictions.
    - Renvoie des données json (intégrant les paramètres des 3 scénarii et leurs prédictions associées).

Le script python appelé via cette méthode prend en entrée les volumétries agrégées (de janvier à octobre) de l'année en cours (si la simulation est faite entre octobre et décembre) ou de l'année passée (si la simulation est faite entre janvier et octobre) de la vue « svg_v_vol_protegee_mois_colonne », le

nombre d'ouvertures, le nombre de décommissions, le nombre de serveurs de type DAT, APP, FIC pour l'année N+1. Il charge le modèle enregistré au format pickle ainsi que le schéma de scalarisation des entrées. Il fait une prédiction avec les valeurs renseignées et insère les nouvelles prédictions de volumétrie N+1 dans la table svg_predictions.

PARTIE FRONT-END (coté application)

La partie front-end a été développée grâce à @ANGULAR/CLI qui est un outil permettant de créer, construire, générer et tester des applications et librairies Angular utilisant le langage Typescript. Elle s'appuie sur Node.js et npm.

Un projet développé en Angular est structuré en plusieurs composants (=components) imbriqués les uns dans les autres. Pour définir la structure de cette ossature, nous avons eu la chance d'interagir avec l'équipe UX de l'entreprise afin d'avoir une maquette de l'application future. Elle a pris en compte nos besoins et s'est inspirée des visuels du premier livrable. L'organisation graphique a permis de donner la structure au projet (et donc de la hiérarchie des components). La maquette UX et sa correspondance en hiérarchie de components sont données en annexes.

Une formation interne sur Angular a été assurée par un expert de l'équipe et a permis de nous lancer dans le développement du projet. Le développement a d'abord visé l'affichage de tous les components, et le « câblage » de tous les composants graphiques et les services associés. Dans un second temps, nous avons réduit l'affichage des components en fonction des options du menu et de l'entête sélectionnés par l'utilisateur. L'application affiche par défaut le component « analyse » du volet Sauvegarde correspondant (Figure 20).

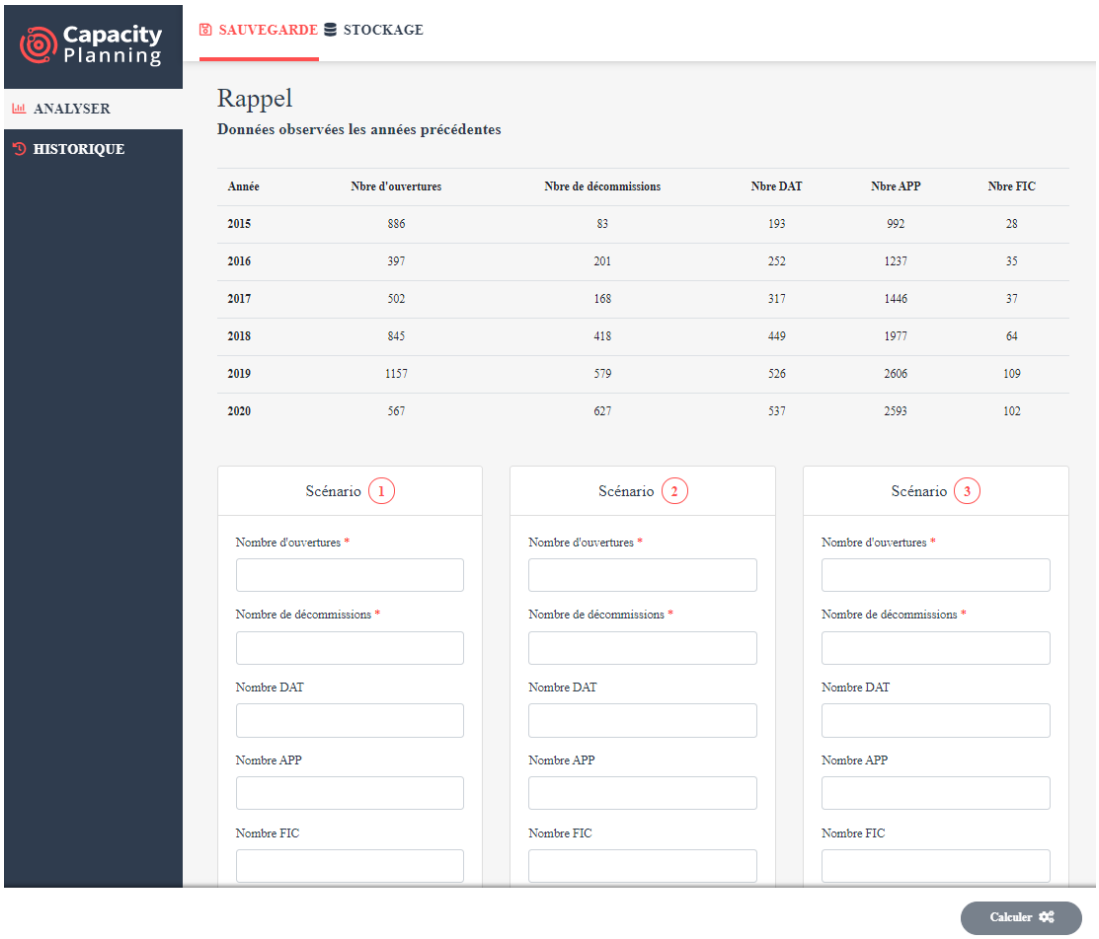


Figure 20:Interface de l'application pour le volet Sauvegarde / Analyser , qui permet le paramétrage de 3 scénarii pour faire des prédictions sur la volumétrie de l'année à venir.

Des échanges avec le gestionnaire de plateforme nous ont amenés à proposer un calcul des valeurs par défaut du nombre de serveurs DAT, APP et FIC, sur la base du nombre d'ouvertures/décommissions donnés par l'utilisateur. En effet, le gestionnaire de plateforme n'a pas toujours le temps, ni les informations des avant-ventes pour compléter précisément ces paramètres. Nous nous sommes appuyés sur le fait que le pourcentage de serveurs par typologie est relativement stable dans le temps (respectivement 81%, 17% et 2%). Ainsi si l'utilisateur doit renseigner les 2 premiers champs des scénarii pour pouvoir accéder au bouton « Calculer », sans être obliger de renseigner les 3 derniers champs des scénarii.

Un fois, ce bouton pressé, l'interface affiche le component « resultats » qui correspond à la figure 21. La prédiction du modèle pour chaque scenario est visible dans le tableau récapitulatif et sur les graphiques. L'utilisateur peut revenir sur la page de paramétrage via le bouton « Modifier scenarios ».

En parallèle du travail de développement de l'application ANGULAR, nous avons eu des échanges avec l'équipe UI de l'entreprise afin de collaborer sur la partie mise en forme de l'application. Leur travail a permis un gain considérable en temps et en qualité de l'interface finale.

Nous avons développé toutes les fonctions nécessaires à l'affichage des données de l'application, fonctions ou components associés à des services httpClient. Ces services sont répertoriés dans un fichier « service1.service.ts » et sont appelés à l'initialisation du component (affichage) ou par un évènement (clic bouton). Ils ont été décrit précédemment dans la partie BACK-END.

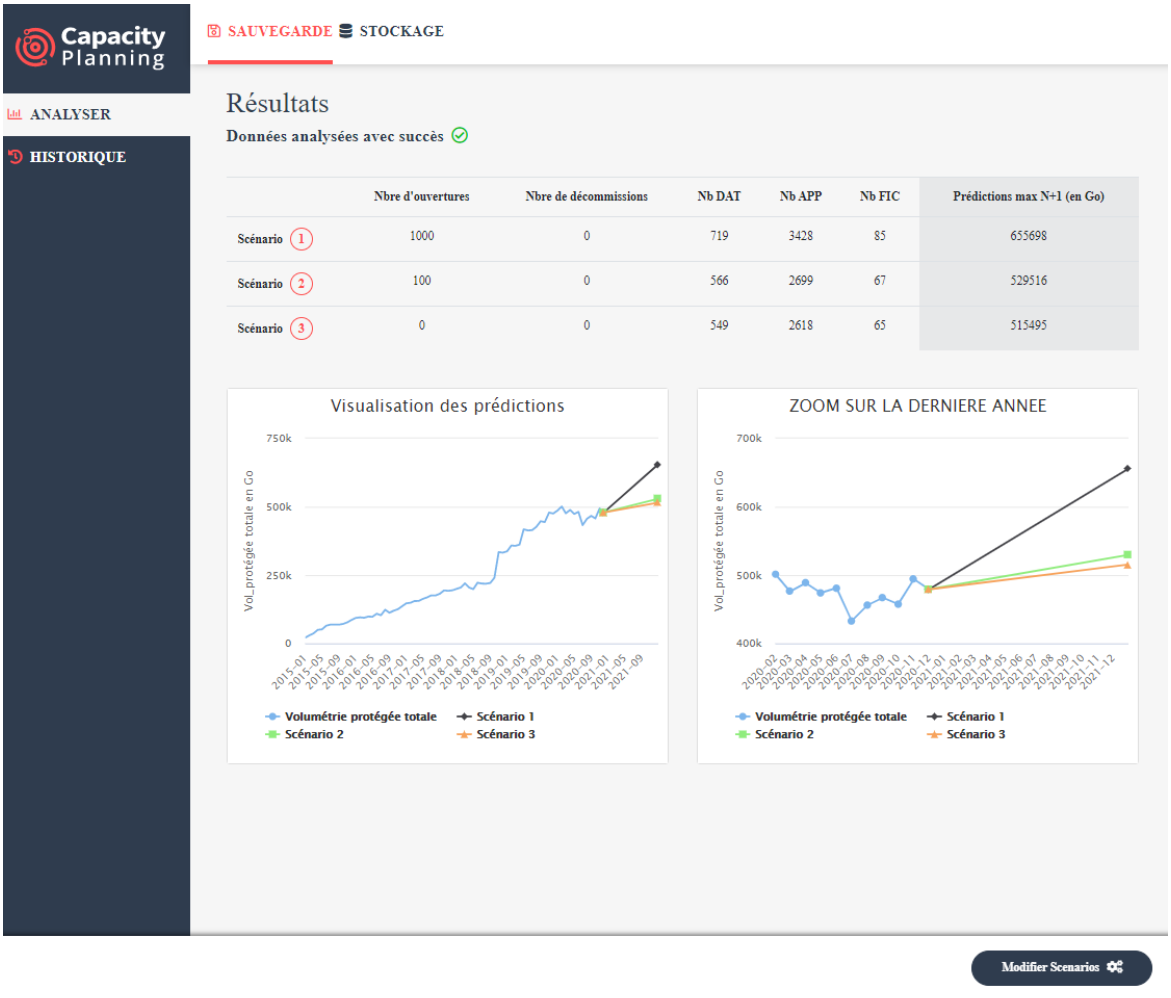


Figure 21 : Interface de l'application dans l'affichage des résultats de prédictions : tableau récapitulatif et graphiques.

Nous avons assuré la partie intégration, avec l'appui de l'expert Angular, de sorte à intégrer les parties HTML (balises, mise en forme) produites par l'équipe UI. Une présentation du produit final à l'équipe UI est permis de rendre compte du travail accompli. Une satisfaction collective s'en dégageait.

L'onglet Historique permet la visualisation des données sous forme de tableau et de graphique via le component « historique » : les indicateurs sur l'ensemble de l'historique (Fig 22).

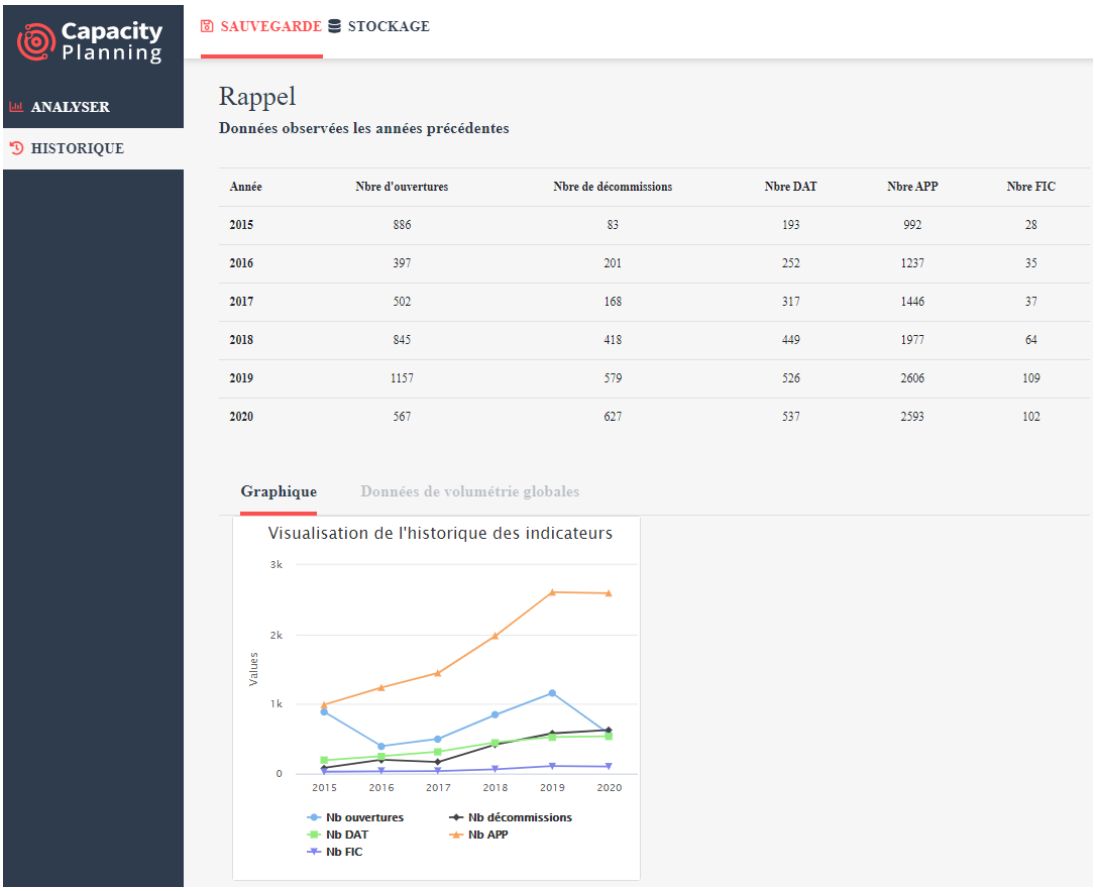


Figure 22 : Interface de l'application sur la partie Historique, affichage de l'onglet 'Graphique'

DEPLOIEMENT

Une fois le développement fini, le projet Node.js Express et le projet Angular (application) doivent être compilés. Cette compilation crée pour chacun des projets un répertoire « dist » dont le contenu constitue le projet en javaScript.

L'application est accessible en interne via une url « **signwpapp009/cp** » grâce au serveur http WINDOWS IIS (gestionnaire de services). Ce service cible le répertoire dans lequel est déposé le projet compilé.

4.1.3 Ajout d'une fonctionnalité supplémentaire

Notre travail sur les données a généré une demande mensuelle de manière indépendante au projet « Capacity Planning » de la part de l'équipe d'infogérance. En effet, la récupération des données et de leur agrégat nous a été demandé en début de chaque mois. Un fichier au format csv leur a été fourni

via un travail indépendant du projet. Nous avons donc profité du développement de l'application ANGULAR pour intégrer une fonctionnalité de téléchargement d'un fichier csv (Fig 23).

Année	Janvier	Février	Mars	Avril	Mai	Juin	Juillet	Août	Septembre	Octobre	Novembre	Décembre
2015	21489	30123	36869	49956	51489	64912	68990	68974	68833	71527	77310	86072
2016	93471	95548	94026	98420	97810	108790	103167	123362	112289	119839	125238	135995
2017	146903	149027	155291	155388	163121	168617	175904	175752	181943	194052	192686	194463
2018	199802	205030	219944	204551	198852	222241	219115	218703	221333	240308	334323	332177
2019	337205	358592	357005	361647	417433	413644	414412	426840	447299	444158	478930	475293
2020	486388	501074	476120	488488	474005	481359	432942	455933	466974	457517	493931	479221

Figure 23 : Interface de l'application sur la partie Hsitorique / Onglet Données de volumétrie globale

Le travail de récupération et d'agrégat de données a aussi été fourni pour le service de Stockage. Cela a été possible par le fait que les données de stockage soient fournies par le gestionnaire de plateforme concerné. En effet, si l'accent a été mis sur le volet Sauvegarde, le volet Stockage nécessite le même travail de récupération de données, d'agrégat, de nettoyage pour l'insertion en base de données. Les scripts de création de tables et d'alimentation initiale sont donnés en annexes, ainsi que le diagramme structural de ces tables.

4.2. Gestion de projet

Le planning défini initialement n'a pas été complètement respecté car les compétences techniques de l'équipe se sont développées au fur et à mesure du projet. Si les étapes principales ont été parcourues, chacune d'elles a demandé des temps de formation, d'auto-formation, de concertations avec des membres de l'équipe pour la maitrise des outils utiles au projet :

formation GIT, formation TYPESCRIPT, formation ANGULAR, Atelier Node.js (EXPRESS), auto-formation sur les séries temporelles (ARIMA, Lissage exponentiel simple/double).

Le projet s'est déroulé sur 9 mois et se présente avec les étapes ci-dessous via l'outil Web Notion (Fig 24) :

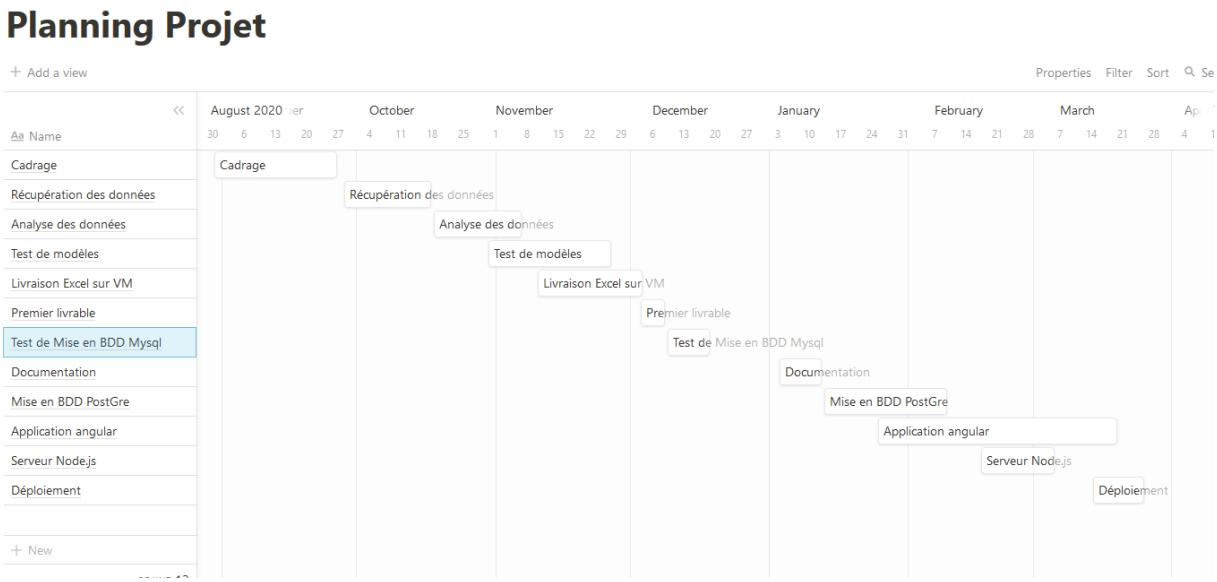


Figure 24 : Planning du projet présentant les différentes étapes parcourues

Si le projet a démarré avec une organisation AGILE, en sprints de 2 semaines (étendus à 3 semaines dans un premier temps) celui-ci a été abandonnée car ne convenait pas au rythme de l'équipe de développement. Le projet s'est déroulé avec la mise en place de « Stand-up », point de 15 minutes tous les matins, pour évoquer ses actions de la veille, ses actions pour le jour à venir, et les difficultés potentielles. Ces points quotidiens sont un point important dans l'élaboration du projet car ce fonctionnement permet de savoir quelles sont les difficultés et problèmes à résoudre.

Une aide substantielle du leader technique a été apporté tout au long du projet, et particulièrement lors de la réalisation du second livrable. Ce qui a été un élément déterminant dans la réussite de ce projet.

La complétude de ce projet nous a demandé de réaliser une multitude de missions spécifiques propres à différents métiers. Nous avons été à la fois développeur, concepteur, data analyst, chef de projet, data scientist, intégrateur, assistant maitre d'ouvrage (pour le recueil du besoin). Nous avons donc appris énormément de choses dans ces différentes missions.

Des réunions ponctuelles avec les gestionnaires de plateforme ont été nécessaires afin de valider la bonne orientation du développement du projet et d'envisager des pistes d'amélioration, ajustements pour être sûre de répondre aux besoins métiers.

Des réunions du COPIL (comité de pilotage) ont aussi permis de rendre compte de l'avancée du projet auprès des parties-prenantes.

Le projet s'est clôturé par des échanges avec les gestionnaires de plateforme afin de l'initier à l'application et de recueillir leurs remarques pour d'éventuelles améliorations.

4.3. Retours d'expérience sur les outils, techniques et compétences à l'œuvre tout au long du projet

Le projet s'est déroulé en collaboration étroite avec mon collègue alternant de Simplon. Notre travail a gagné en efficacité à partir du moment où nous avons mis en place le répertoire GITLAB pour le partage de fichiers de développement. Le partage des fichiers s'est fait dans un premier sous TEAMS. Mais

cet outil n'est des plus pertinent dans le travail de développement collaboratif. En effet, travaillant conjointement sur les mêmes fichiers nous obligeait à être vigilant sur la version utilisée pour continuer le travail de développement sur notre propre machine. Nous avons réellement gagné en expérience sur ce point-là.

Le travail d'exploration des données s'est fait sous python via les notebooks. Or, la mise en base de données nous a permis de voir que les transformations de données faites sous python sont aussi très facilement réalisables via des requêtes SQL et il est vrai que ce langage commun à d'autres membres de l'équipe nous a permis d'être épaulés plus concrètement sur cette partie.

Notre manque de vision globale d'un projet nous a fait retarder la mise en base des données alors même que cette étape était nécessaire pour le fonctionnement et la pérennité de l'application finale. Le livrable intermédiaire aurait pu être éviter. En effet, le besoin de mise à jour en continu des données n'avait pas été bien visualisé au départ. Or la base de données est un moyen efficace pour stocker les données et réalisé l'ajout des données générées tous les mois.

4. Bilan du projet et les améliorations envisageables.

Ce projet s'est révélé être un projet comportant toutes les étapes importantes d'un projet d'IA depuis la récupération des données jusqu'au déploiement de l'application. C'est pourquoi, même si nous avons rencontrées des difficultés, je suis très fière d'avoir pu mener à bien ce projet avec l'appui de mes collègues.

Une petite déception m'habite quant au fait de ne pas avoir eu le temps de déployer un modèle d'IA propre au volet STOCKAGE. Un temps devra être donné pour l'analyse des données et la recherche d'un modèle performant. Les données sont déjà en base et sont accessibles par le gestionnaire de plateforme via le téléchargement d'un csv. Heureusement le produit final peut aisément être complété par l'intégration d'un modèle de prédiction propre à la partie STOCKAGE.

5. Conclusion.

La réalisation de ce projet a permis de couvrir un maximum de compétences attendus pour la validation du titre professionnel « Développeur Data IA ». Ce projet a été un des premiers projets IA réalisés de bout en bout par l'entreprise SIGMA. J'ai été ravie de réaliser toutes les étapes de ce projet : compréhension du besoin, collecte de données, transformation et nettoyage des données, analyse et visualisation, choix et entrainement d'un modèle, mise à disposition du modèle de prédiction, livraison et formation des utilisateurs.

Cette expérience a été très enrichissante que ce soit pour la résolution des problèmes fonctionnels ou techniques et pour la collaboration avec les autres membres de l'équipe. Ce projet fut la première étape vers une montée en compétences dans le domaine de l'intelligence artificielle et m'a apporté un immense sentiment de satisfaction.

Je remercie infiniment l'entreprise SIGMA de m'avoir accueillie en tant qu'alternante pour la réalisation de ce projet et espère que l'équipe IA du groupe continuera sur de nouveaux projets IA passionnants.

ANNEXES

Script de création des tables du projet SAUVEGARDE

NB : Les tables sont créées dans un ordre qui respecte les relations qui les relient.

```
CREATE TABLE svg_offres(
        id serial PRIMARY KEY,
        offre VARCHAR(25) NOT NULL    );
CREATE TABLE svg_typologies (
        id serial PRIMARY KEY,
        new_type VARCHAR(25) NOT NULL   );
CREATE TABLE svg_outil_sauvegardes (
        id serial PRIMARY KEY,
        outil VARCHAR(10) NOT NULL );
CREATE TABLE svg_crma (
        id serial PRIMARY KEY,
        crma VARCHAR(250) NOT NULL,
        date DATE NOT NULL,
        idoutil_sauvegarde INTEGER NOT NULL,
        FOREIGN KEY (idoutil_sauvegarde)
           REFERENCES svg_outil_sauvegardes(id)
           ON UPDATE CASCADE ON DELETE CASCADE );
CREATE TABLE svg_serveurs (
        nom_serveur VARCHAR(100) PRIMARY KEY,
        date_crea DATE NOT NULL,
        date_maj DATE NOT NULL,
        idtypologie INTEGER REFERENCES svg_typologies(id) NOT NULL,
        idoffre INTEGER REFERENCES svg_offres(id) NOT NULL,
        idoutil_sauvegarde INTEGER REFERENCES svg_outil_sauvegardes(id) NOT NULL );
CREATE TABLE svg_enregistrements (
        id serial PRIMARY KEY,
        nom_serveur VARCHAR(100) REFERENCES svg_serveurs(nom_serveur) NOT NULL,
        date DATE NOT NULL,
        vol_protegee numeric not null,
        vol_stockee numeric not null,
        politique VARCHAR(300),
```

```
    client VARCHAR(50) );
```

Script de création de tables pour le modèle IA :

```
CREATE TABLE svg_parametres(
    id serial PRIMARY KEY,
    nb_ouvertures INTEGER NOT NULL,
    nb_decommissions INTEGER NOT NULL,
    nb_dat INTEGER NOT NULL,
    nb_app INTEGER NOT NULL,
    nb_fic INTEGER NOT NULL    );
CREATE TABLE svg_predictions (
    id serial PRIMARY KEY,
    prediction numeric NOT NULL    ) ;
```

Script de création des vues

```
-------------------------création de la vue svg_v_vol_protegee_mois_colonne -------------------------
create view svg_v_vol_protegee_mois_colonne as (
with data as (
        select date_part('year', date) annee, date_part('month', date) num_mois, sum(vol_protegee)
vol_pro
        from svg_enregistrements
        group by date
)
select annee,
        sum(vol_pro) filter (where num_mois = 1) as janvier,
        sum(vol_pro) filter (where num_mois = 2) as fevrier,
        sum(vol_pro) filter (where num_mois = 3) as mars,
        sum(vol_pro) filter (where num_mois = 4) as avril,
        sum(vol_pro) filter (where num_mois = 5) as mai,
        sum(vol_pro) filter (where num_mois = 6) as juin,
        sum(vol_pro) filter (where num_mois = 7) as juillet,
        sum(vol_pro) filter (where num_mois = 8) as aout,
        sum(vol_pro) filter (where num_mois = 9) as septembre,
        sum(vol_pro) filter (where num_mois = 10) as octobre,
```

```
            sum(vol_pro) filter (where num_mois = 11) as novembre,
            sum(vol_pro) filter (where num_mois = 12) as decembre
from data
group by annee
order by année );
-------------------------création vue svg_v_vol_pro_par_mois--------------------
CREATE VIEW svg_v_vol_protegee_par_mois as
SELECT date, sum(vol_protegee) as vol_protegee_total
       FROM public.svg_enregistrements
          group by date
          order by date DESC;
----- ------------------Création de vue svg_v_nombres_serveurs-----------------------
create view public.svg_v_nombres_serveurs as (
with data as (
            select date_part('year', date) annee, svg_enregistrements.nom_serveur, svg_serveurs.idtypologie
from svg_enregistrements
            inner join svg_serveurs on svg_serveurs.nom_serveur = svg_enregistrements.nom_serveur)
                        select annee,
                        count(distinct data.nom_serveur) filter (where idtypologie=0) as nb_app,
                        count(distinct data.nom_serveur) filter (where idtypologie=1) as nb_dat,
                        count(distinct data.nom_serveur) filter (where idtypologie=2) as nb_fic
                        from data
                        group by annee);
--------------------vue ouvertures--------------------
CREATE OR REPLACE VIEW public.svg_v_ouvertures
 AS
 WITH data AS (
        SELECT date_part('year'::text, svg_serveurs.date_crea) AS annee_crea,
          date_part('year'::text, svg_serveurs.date_maj) AS annee_maj
         FROM svg_serveurs
        WHERE (svg_serveurs.date_crea > ( SELECT min(svg_serveurs_1.date_crea) AS max
```

```
            FROM svg_serveurs svg_serveurs_1))
    )
 SELECT data.annee_crea AS annee,
    count(data.annee_crea) AS nb_ouvertures
   FROM data
  GROUP BY data.annee_crea
  ORDER BY data.annee_crea;
----------------------vue decommissions--------------------
CREATE OR REPLACE VIEW public.svg_v_decommissions
 AS
WITH data AS (
        SELECT date_part('year'::text, svg_serveurs.date_crea) AS annee_crea,
           date_part('year'::text, svg_serveurs.date_maj) AS annee_maj
          FROM svg_serveurs
         WHERE (svg_serveurs.date_maj < ( SELECT max(svg_serveurs_1.date_maj) AS max
                FROM svg_serveurs svg_serveurs_1))
    )
 SELECT data.annee_maj AS annee,
    count(data.annee_maj) AS nb_decommissions
   FROM data
  GROUP BY data.annee_maj
  ORDER BY data.annee_maj;
---------------------- vue indicateurs-----------------------------
CREATE OR REPLACE VIEW public.svg_v_indicateurs
 AS
 WITH data AS (
        SELECT svg_v_ouvertures.annee,
           svg_v_ouvertures.nb_ouvertures,
           svg_v_decommissions.nb_decommissions,
           svg_v_nombres_serveurs.nb_dat,
           svg_v_nombres_serveurs.nb_app,
```

```
        svg_v_nombres_serveurs.nb_fic
       FROM ((svg_v_ouvertures
         JOIN svg_v_decommissions ON ((svg_v_decommissions.annee = svg_v_ouvertures.annee)))
         JOIN svg_v_nombres_serveurs ON ((svg_v_nombres_serveurs.annee = svg_v_ouvertures.annee)))
     )
SELECT data.annee,
   data.nb_ouvertures,
   data.nb_decommissions,
   data.nb_dat,
   data.nb_app,
   data.nb_fic
  FROM data;
```

Script python d'insertion initiale en base de données :

```
import pandas as pd
import psycopg2
import psycopg2.extras as extras
from io import StringIO
import datetime
from config import config
from connect import connect

# read database configuration
params = config()
# connect to the PostgreSQL database
connection = psycopg2.connect(**params)
# create a new cursor
cur = connection.cursor()

# Etape 1 : création de dataframes au format adapté (selon le schemas des tables de la bdd)

# 1- svg_enregistrement
enregistrements = pd.read_csv("df_sum.csv")
enregistrements.Date = enregistrements.Date.map(lambda x: datetime.datetime.strptime(x, "%Y-%m-%d"))
enregistrements = enregistrements.rename(columns={"Serveur":"nom_serveur", "Date": "date", "Vol_protegee": "
vol_protegee", "Vol_stockee":"vol_stockee", "Politique":"politique", "Client":"client"})
enregistrements = enregistrements[["nom_serveur", "date", "vol_protegee", "vol_stockee", "politique", "clien
t"]]
enregistrements = enregistrements.reset_index()
enregistrements = enregistrements.rename(columns={"index": "id"})

# 2- svg_serveurs
```

```
table_serveurs = pd.read_csv("table_serveurs.csv")
table_serveurs = table_serveurs.rename(columns={"Serveur":"nom_serveur", "Date_crea": "date_crea", "Date_maj
":"date_maj", "New_Type": "new_type", "Offre": "offre", "Outil":"outil"})

# 3 - svg_outil_sauvegarde
outil_sauvegardes = pd.DataFrame(table_serveurs.outil.unique(), columns=["outil"])
outil_sauvegardes = outil_sauvegardes.reset_index()
outil_sauvegardes = outil_sauvegardes.rename(columns={"index": "idoutil_sauvegarde"})

# 4 - svg_crma
df_crma_veeam = pd.read_csv("df_crma_veeam.csv")
df_crma_veeam["outil"] = "Veeam"
df_crma_netbackup = pd.read_csv("df_crma_Netbackup.csv")
df_crma_netbackup["outil"] = "NBU"
df_crma = pd.concat([df_crma_netbackup, df_crma_veeam], ignore_index=True)
df_crma = df_crma.reset_index()
df_crma = df_crma.rename(columns={"index": "id"})
df_crma.date = df_crma.date.map(lambda x: datetime.datetime.strptime(x, "%Y-%m"))

df_crma = df_crma.merge(outil_sauvegardes[['idoutil_sauvegarde', 'outil']], on="outil")
df_crma = df_crma[['id','crma', 'date', 'idoutil_sauvegarde']]

# 5 - svg_typologies
typologies = pd.DataFrame(table_serveurs.new_type.unique(), columns=["new_type"])
typologies = typologies.reset_index()
typologies = typologies.rename(columns={"index": "idtypologie"})

# 6 - svg_offres
offres = pd.DataFrame(table_serveurs.offre.unique(), columns=["offre"])
offres = offres.reset_index()
offres = offres.rename(columns={"index": "idoffre"})

# 7 merge des tables avec la table serveurs
table_serveurs = table_serveurs.merge(typologies[['idtypologie','new_type']], on="new_type")
table_serveurs = table_serveurs.merge(offres[['idoffre','offre']], on="offre")
table_serveurs = table_serveurs.merge(outil_sauvegardes[['idoutil_sauvegarde','outil']], on="outil")
table_serveurs = table_serveurs[["nom_serveur", "date_crea", "date_maj", "idtypologie", "idoffre", "idoutil_
sauvegarde"]]

# renommage des colonnes des 3 tables pour l'insertion. sinon probleme
outil_sauvegardes = outil_sauvegardes.rename(columns={"idoutil_sauvegarde": "id"})
offres = offres.rename(columns={"idoffre": "id"})
typologies = typologies.rename(columns={"idtypologie": "id"})

# Etape 2 : Insertion des dataframes dans les tables respectives

def insert_table(conn, df, table):
        # Create a list of tupples from the dataframe values
        tuples = [tuple(x) for x in df.to_numpy()]
```

```
        # Comma-separated dataframe columns
        cols = ','.join(list(df.columns))
        # SQL quert to execute
        query = "INSERT INTO %s(%s) VALUES %%s" % (table, cols)
        cursor = conn.cursor()
        try:
            extras.execute_values(cursor, query, tuples)
            conn.commit()
        except (Exception, psycopg2.DatabaseError) as error:
            print("Error: %s" % error)
            conn.rollback()
            cursor.close()
            return 1
        print("insert_table() done")
        cursor.close
# Insertion des df dans leurs tables respectives
insert_table(connection, outil_sauvegardes, 'svg_outil_sauvegardes')
insert_table(connection, df_crma, 'svg_crma')
insert_table(connection, offres, 'svg_offres')
insert_table(connection, typologies, 'svg_typologies')
insert_table(connection, table_serveurs, 'svg_serveurs')
insert_table(connection, enregistrements, 'svg_enregistrements')

###ETAPE 3 : Création de dataframes avec des valeurs par défaut pour les tables svg_parametres et svg_predic
tions
df_parametres = pd.DataFrame(list(zip(['1', '2', '3'], [200, 300, 400], [50, 10, 150], [200, 300, 400], [200
0, 2500, 3000], [30, 50, 70])),columns=["id", "nb_ouvertures", "nb_decommissions", "nb_dat", 'nb_app', "nb_f
ic"])

df_predictions = pd.DataFrame(list(zip(['1', '2', '3'], ['200', '300', '400'])), columns=["id", "prediction"
])

# Insertion de valeurs par défaut dans les tables du modèle IA
insert_table(connection, df_parametres, 'svg_parametres')
insert_table(connection, df_predictions, 'svg_predictions')
```

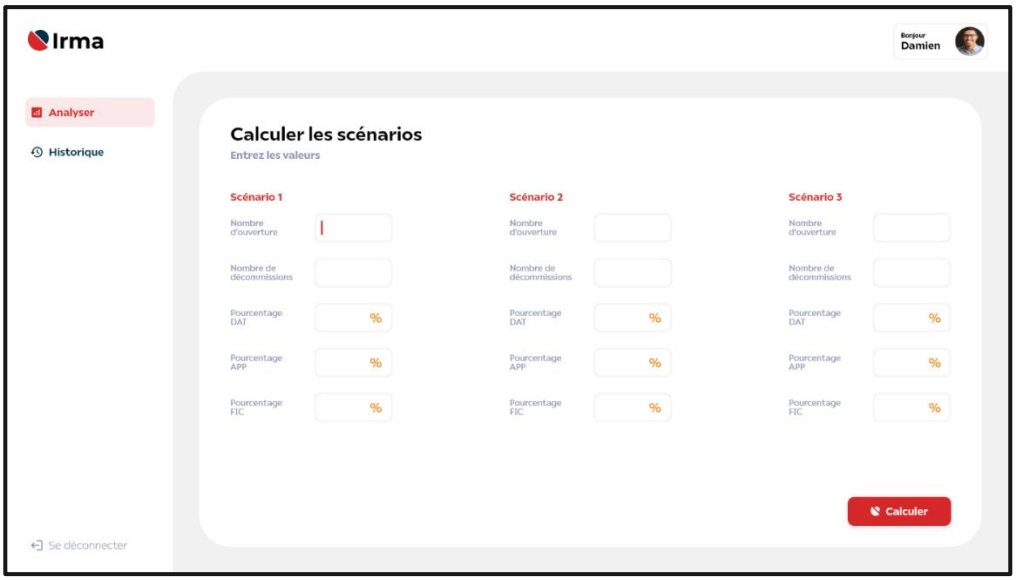
Performances des modèles Holt-Winter en fonction du type de saisonnalité (additive/multiplicative) et d'amortissement intégré ou non :

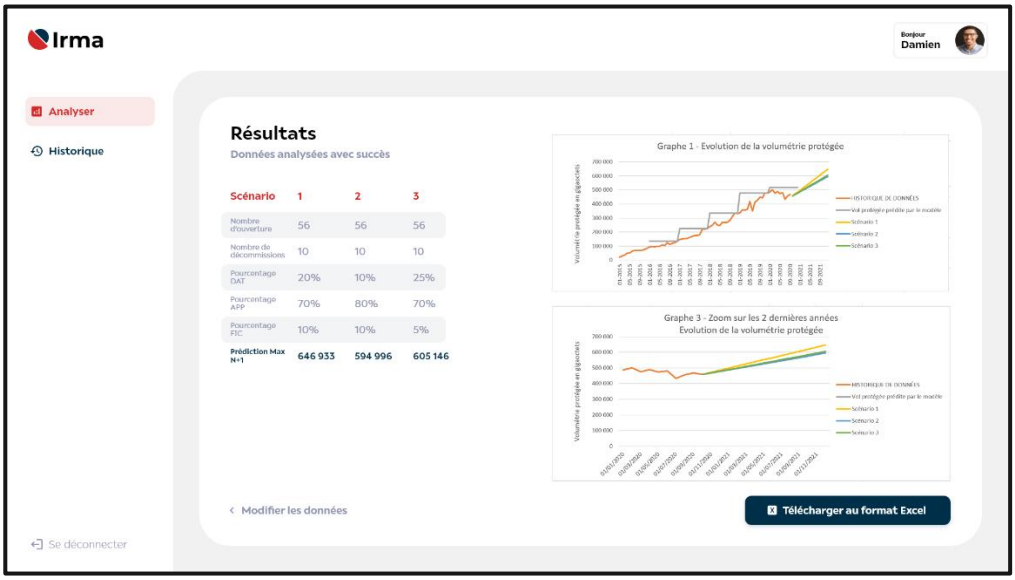
Modèles Holt Winter **(MAPE – Mean absolute percentage error)** **= (Y_true – y_pred) ÷ y_true) *100**	**2015**	**2016**	**2017**	**2018**	**2019**	**2020**
Modèle 1 (trend additive , season additive)	58.16	24.93	14.92	13.32	10.34	13.70
Modèle 2 (trend additive , season multiplicative)	50.05	14.57	16.54	12.04	11.28	17.28
Modèle 3 (trend additive , season additive + damped)	84.88	28.59	15.39	13.27	10.38	13.29
Modèle 4 (trend additive, season multiplicative + damped)	54,48	13,93	16,71	10,98	10,93	13,92

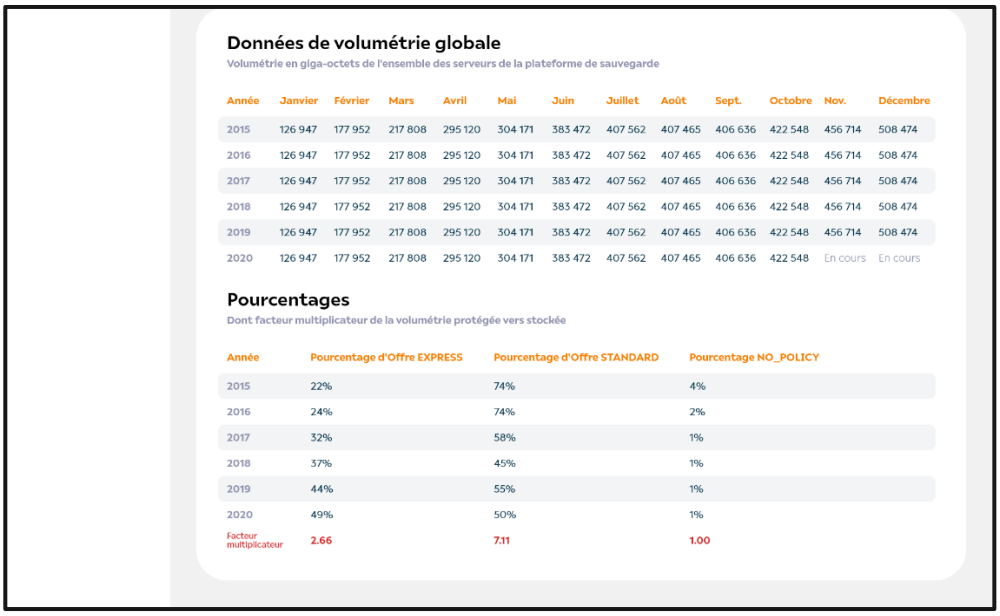
Performances des modèles de régression multiple :

	Modèle 1 à 10 entrées (= Vol de janvier à octobre)	Modèle 2 à 12 entrées (= Vol de janvier à octobre + Nb d'ouvertures et Nb de décommissions)	Modèle 3 à 15 entrées (= Vol de janvier à octobre + Nb d'ouvertures et Nb de décommissions + Nb de DAT, Nb de APP, Nb de FIC)
Prédiction (Go)	682 084.80	460 352.52	486 212.89
Volumétrie réelle (Go)	501 074.48	501 074.48	501 074.48
Performances des modèles (MAPE)	36%	8%	3%

Proposition de l’équipe UX - Maquette de l’application







Données de volumétrie globale

Volumétrie en giga-octets de l'ensemble des serveurs de la plateforme de sauvegarde

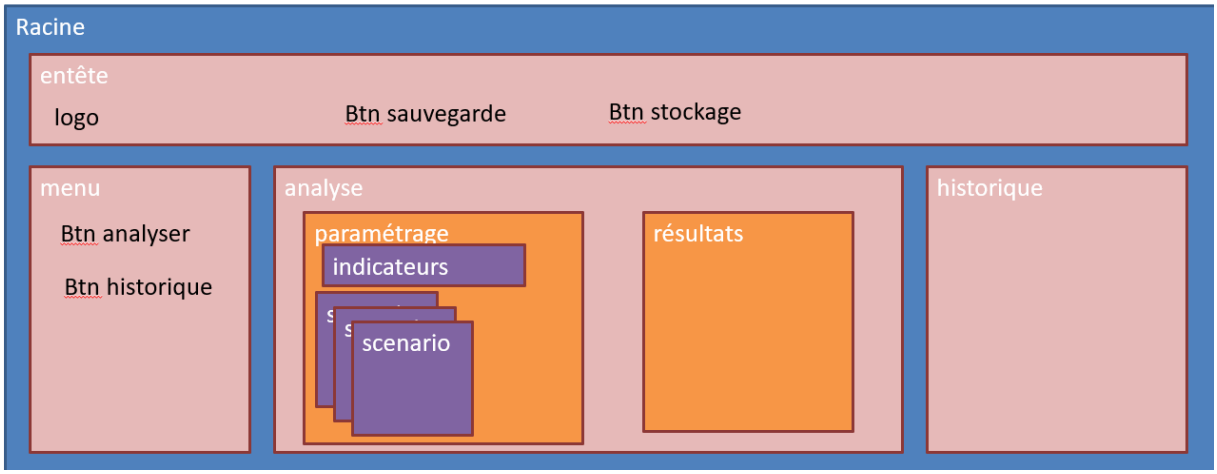
Année	Janvier	Février	Mars	Avril	Mai	Juin	Juillet	Août	Sept.	Octobre	Nov.	Décembre
2015	126 947	177 952	217 808	295 120	304 171	383 472	407 562	407 465	406 636	422 548	456 714	508 474
2016	126 947	177 952	217 808	295 120	304 171	383 472	407 562	407 465	406 636	422 548	456 714	508 474
2017	126 947	177 952	217 808	295 120	304 171	383 472	407 562	407 465	406 636	422 548	456 714	508 474
2018	126 947	177 952	217 808	295 120	304 171	383 472	407 562	407 465	406 636	422 548	456 714	508 474
2019	126 947	177 952	217 808	295 120	304 171	383 472	407 562	407 465	406 636	422 548	456 714	508 474
2020	126 947	177 952	217 808	295 120	304 171	383 472	407 562	407 465	406 636	422 548	En cours	En cours

Pourcentages

Dont facteur multiplicateur de la volumétrie protégée vers stockée

Année	Pourcentage d'Offre EXPRESS	Pourcentage d'Offre STANDARD	Pourcentage NO_POLICY
2015	22%	74%	4%
2016	24%	74%	2%
2017	32%	58%	1%
2018	37%	45%	1%
2019	44%	55%	1%
2020	49%	50%	1%
Facteur multiplicateur	2.66	7.11	1.00

Structure – Hiérarchie de components ANGULAR



App-component => module-sauvegarde => menu

=> entete

=>analyse => parametrage => Indicateurs

=>Scenario

=> resultats

=>historique

Script de création des tables Stockage

```
CREATE TABLE "stk_origines" (
        "id_origine" INTEGER NOT NULL,
        "origine" VARCHAR NULL DEFAULT NULL,
        PRIMARY KEY ("id_origine") );
-----------------------------------------------------------------------
CREATE TABLE "stk_typologies" (
        "id_typologie" INTEGER NOT NULL,
        "name_typologie" VARCHAR NULL DEFAULT NULL,
        PRIMARY KEY ("id_typologie") );
-----------------------------------------------------------------------
CREATE TABLE "stk_fichiers" (
        "nom_fichier" VARCHAR NOT NULL,
        "id_origine" INTEGER NULL DEFAULT NULL,
```

```
	PRIMARY KEY ("nom_fichier"),
	CONSTRAINT "id_origine" FOREIGN KEY ("id_origine") REFERENCES "public"."stk_origines" ("id_origine") ON UPDATE NO ACTION ON DELETE NO ACTION );

-----------------------------------------------------------------------

CREATE TABLE "stk_offres" (
	"id_offre" INTEGER NOT NULL,
	"name_offre" VARCHAR NULL DEFAULT NULL,
	PRIMARY KEY ("id_offre") );

-----------------------------------------------------------------------

CREATE TABLE "stk_pools" (
	"id_pool" INTEGER NOT NULL,
	"name_pool" VARCHAR NOT NULL,
	"id_offre" INTEGER NOT NULL,
	PRIMARY KEY ("id_pool"),
	CONSTRAINT "id_offre" FOREIGN KEY ("id_offre") REFERENCES "public"."stk_offres" ("id_offre") ON UPDATE NO ACTION ON DELETE NO ACTION );

-----------------------------------------------------------------------

CREATE TABLE "stk_enregistrements" (
	"id_enreg" INTEGER NULL DEFAULT NULL,
	"name_volume" VARCHAR NULL DEFAULT NULL,
	"capacity" NUMERIC NULL DEFAULT NULL,
	"used_capacity" NUMERIC NULL DEFAULT NULL,
	"date" DATE NULL DEFAULT NULL,
	"client" VARCHAR NULL DEFAULT NULL,
	"id_pool" INTEGER NULL DEFAULT NULL,
	"id_typologie" INTEGER NULL DEFAULT NULL,
	"id_origine" INTEGER NULL DEFAULT NULL,
	CONSTRAINT "id_origine" FOREIGN KEY ("id_origine") REFERENCES "public"."stk_origines" ("id_origine") ON UPDATE NO ACTION ON DELETE NO ACTION,
	CONSTRAINT "id_pool" FOREIGN KEY ("id_pool") REFERENCES "public"."stk_pools" ("id_pool") ON UPDATE NO ACTION ON DELETE NO ACTION,
	CONSTRAINT "id_typologie" FOREIGN KEY ("id_typologie") REFERENCES "public"."stk_typologies" ("id_typologie") ON UPDATE NO ACTION ON DELETE NO ACTION );
```

Script d'alimentation initiale des données de STOCKAGE

```
import pandas as pd
import numpy as np
import datetime
import os
from config import config
from connect import connect
import psycopg2
import psycopg2.extras as extras

# Paramètre de connexion à la base de donnée
params = config()
# connect to the PostgreSQL server
print('Connecting to the PostgreSQL database...')
connection = psycopg2.connect(**params)

# Mise en base
df = pd.read_csv('df_volume.csv')
df =
df.rename(columns={"name":"name_volume",'pool':'name_pool',"typologie":"name_typologie"}
)

# pour la création des tables
offres = pd.DataFrame(df.offre.unique(), columns=['offre'])
offres = offres.reset_index()
offres = offres.rename(columns={"index": "id_offre", "offre":"name_offre"})

pools = pd.DataFrame(df.name_pool.unique(), columns=['name_pool'])
pools['name_offre'] =
["T0","T0","T2","T2","T2_V7000","T0_V7000","T2_V7000","T2_TMP","T2_TMP","T2_FCM","T2_FCM
","T2","T2"]
pools = pools.reset_index()
pools = pools.rename(columns={"index": "id_pool"})

pools_plus = pools.merge(offres, on="name_offre")

# créer une table typologies et une colonne identifiant
typologies = pd.DataFrame(df.name_typologie.unique(), columns=["name_typologie"])
typologies = typologies.reset_index()
typologies = typologies.rename(columns={"index": "id_typologie"})

origines = pd.DataFrame(df.origine.unique(), columns=['origine'])
origines = origines.reset_index()
origines = origines.rename(columns={"index": "id_origine"})

df_fichiers_stockage =pd.read_csv('df_fichiers_stockage.csv')
df_fichiers_stockage =df_fichiers_stockage.merge(origines , on="origine")

df = df.merge(pools_plus[["id_pool","name_pool"]], on="name_pool")
df = df.merge(typologies, on="name_typologie")
df = df.merge(origines, on="origine")
df = df.reset_index()
df = df.rename(columns={"index": "id_enreg"})

def insert_table(conn, df, table):
    tuples = [tuple(x) for x in df.to_numpy()]
    cols = ','.join(list(df.columns))
    query = "INSERT INTO %s(%s) VALUES %%s" % (table, cols)
    cursor = conn.cursor()
    try:
        extras.execute_values(cursor, query, tuples)
        conn.commit()
    except (Exception, psycopg2.DatabaseError) as error:
        print("Error: %s" % error)
        conn.rollback()
        cursor.close()
        return 1
```

```
    print("insert_table() done")
    cursor.close

# Insert les différents df dans les tables associées
insert_table(connection, origines, 'stk_origines')
insert_table(connection,df_fichiers_stockage[["nom_fichier","id_origine"]],
'stk_fichiers')
insert_table(connection, offres, 'stk_offres')
insert_table(connection, pools_plus[["id_pool","name_pool","id_offre"]], 'stk_pools')
insert_table(connection, typologies, 'stk_typologies')
insert_table(connection, df[['id_enreg','name_volume','capacity','used_capacity',
'date', 'client','id_pool', 'id_typologie', 'id_origine']], 'stk_enregistrements')
```

Structure de la table principale des données STOCKAGE