Master Economiste d'Entreprise

Mémoire de Stage

Développement de logiciels d'impact de perte d'armoire de contrôle-commande et de post-traitement de résultats de simulation

Entreprise

EDF

M. Alexandre PADET
M. Grégoire BOUVET

Corentin DUCLOUX

Université de Tours

M. Vincent PERROLLAZ

8 Avril 2024 — 20 Septembre 2024

Remerciements

Je tiens en premier lieu à remercier Alexandre Padet et Grégoire Bouvet, mes deux tuteurs de stage en entreprise et ingénieurs d'études, pour la confiance qu'ils m'ont accordée dès nos premiers échanges. L'enthousiasme avec lequel ils ont pris le temps de partager leurs connaissances relatives au fonctionnement nucléaire m'a permis de mieux comprendre les données que j'ai eu l'occasion de manipuler et d'appréhender l'impact de mes développements sur les équipes métiers.

Ces remerciements s'adressent aussi à Eric Liret et Grégoire Duhot, dont les points hebdomadaires m'ont poussé à présenter de manière approfondie le fonctionnement du circuit secondaire d'un réacteur à eau pressurisée. Cette démarche m'a été particulièrement bénéfique, eu égard à mon absence de formation sur ces sujets.

Je tiens également à faire part de ma gratitude au Service d'Information et de Documentation (SID) du CNEPE, avec lequel j'ai entretenu des échanges réguliers tout au long de mon stage et qui a toujours été disponible pour répondre à mes très nombreuses interrogations. Je pense notamment à Loïc Tessier, pilote de projet SI, dont l'expertise a été indispensable pour développer des solutions fiables et robustes en contexte de production.

Mes remerciements se tournent évidemment vers l'ensemble du groupe FGP ainsi que les autres groupes du CNEPE avec lesquels j'ai pu travailler au quotidien. Je remercie plus particulièrement Marc Duchamp, chef du groupe FGP, pour m'avoir donné l'occasion de tenir un atelier de présentation à la Direction Technique d'EDF. La bonne humeur au sein du groupe m'a permis de me sentir pleinement intégré dès le début de mon stage. Mention particulière à Amal Rekik, Romane Noublanche et Guillaume Réault, stagiaires du groupe FGP, et à Hind El Harchi, ingénieure d'études, avec lesquels j'ai partagé de nombreux moments de convivialité et d'entraide tout au long de ce stage.

Je tiens à exprimer toute ma reconnaissance à Amandine Thomas, qui a pris le temps de relire entièrement ce mémoire.

Je tiens aussi à remercier chaleureusement Monsieur Vincent Perrollaz, maître de conférences en mathématiques appliquées et professeur d'ingénierie logicielle sous python à l'Université de Tours. Ses enseignements prodigués durant ces deux années de Master ont été véritablement décisifs pour mener à bien mes projets chez EDF. Nos échanges réguliers concernant le stage ainsi que ses recommandations m'ont été d'une grande assistance.

En outre, je remercie l'ensemble de l'équipe pédagogique du Mécen pour ces deux années riches en enseignements.

Je dédie aussi ces lignes à mes parents, ma soeur Amélie et mes amis.

Enfin, je souhaite remercier ma compagne, Charlotte, qui m'a soutenu au quotidien, et sans qui tout cela ne serait pas possible.

* * *

Table des matières

1 | Introduction

La COP28 représente un important marqueur de la reconnaissance du rôle de l'énergie nucléaire dans la lutte contre le changement climatique. A cette occasion, la France signe une déclaration commune appelant à tripler les capacités d'énergie éléctronucléaire dans le monde d'ici à 2050 : il s'agit d'un élément de réponse crédible aux objectifs de sortie des énergies fossiles pour tenter de respecter la trajectoire des 1,5 °C [1] dans le très faible délai imparti.

C'est dans ce contexte qu'EDF signe plusieurs accords de coopération pour décarbonner la production d'électricité et réaffirmer son engagement en faveur du climat. Trois volets sont concernés : le plan « zéro émission nette », les énergies renouvelables et l'énergie nucléaire.

Dans le cadre du master *Mécen* à l'Université de Tours, j'ai souhaité réaliser mon stage de fin d'études dans une entreprise répondant à ces enjeux de décarbonation, tout en poursuivant ma spécialisation dans le domaine de la Data. C'est la raison pour laquelle je me suis naturellement tourné vers EDF, dont la structure et les activités permettent de travailler sur ce type de problématiques.

Dans un premier temps, je débuterai par une présentation d'EDF et du CNEPE, le centre d'ingénierie dans lequel j'ai travaillé. J'aborderai ensuite l'historique de la filière nucléaire en France, avec la présentation d'une centrale REP, avant de détailler les problématiques rencontrées au cours de mon stage. Enfin, je dresserai un bilan de cette expérience professionnelle.

* * *

[1] https://www.ipcc.ch/site/assets/uploads/sites/2/2019/09/SR15_Summary_Volume_french.pdf.

2 | Organisation de mon travail

Pendant mon stage, j'ai été chargé de mener à bien deux projets dans un laps de temps restreint. Afin de faciliter la réussite de ces projets, j'ai adopté une approche itérative, en organisant mon travail sous forme de tâches et fonctionnalités spécifiques à ajouter. Cela m'a permis de mieux prioriser mes efforts et de suivre l'avancement de chaque projet de manière précise, tout en restant flexible face aux ajustements nécessaires.

Pour recueillir les éléments requis, j'ai multiplié les échanges avec les parties prenantes concernées et n'ai pas hésité à poser de nombreuses questions, ce qui m'a aidé à clarifier les besoins et à garantir que les solutions développées répondaient aux attentes des utilisateurs finaux.

Mon rôle a aussi consisté à apporter de meilleures pratiques d'ingénierie logicielle au sein du groupe en maintenant un lien important avec l'équipe informatique. Par exemple, afin d'optimiser mon temps et faciliter la transmission de connaissances, j'ai choisi de documenter mes travaux au fur et à mesure de leur avancement. Cela m'a permis de structurer mes idées et de gagner du temps lors de la phase de passation de projet. En rédigeant la documentation parallèlement au développement, j'ai pu m'assurer que les informations restaient à jour, minimisant ainsi les efforts futurs pour maintenir cette documentation.

* * *

3 | Présentation d'EDF

Electricité De France (EDF) est une société française de production et de fourniture d'électricité. Elle est créée en 1946 en tant que monopole d'Etat – il s'agit plus précisément d'un Etablissement Public à caractère Industriel ou Commercial (EPIC), qui a pour objectif de réunir la production, le transport, la distribution et la fourniture d'électricité. La directive européenne de 1996 portant sur l'ouverture du marché de l'énergie dans les pays membres a vocation à changer le statut d'EDF : en effet, en 2004, EDF devient une Société Anonyme (SA) et n'est plus en situation de monopole sur la fourniture et la production d'électricité. Cependant, en raison de sa nature historique de monopole de fait dans la production d'électricité d'origine nucléaire, le groupe reste aujourd'hui le premier producteur et fournisseur d'électricité en France et en Europe.

Avec un effectif de 179550 collaborateurs, 40,9 millions de clients et générant un chiffre d'affaires de 139,7 milliards d'euros en 2023, le groupe est aussi un des grands leaders mondiaux de l'énergie.

À ce titre, la production d'électricité d'EDF s'élève à 467,6 TWh dans le monde en 2023, dont l'origine principale (77,7%) est issue de l'exploitation de centrales nucléaires. En France, la production d'EDF s'élève quant à elle à 365,9 TWh avec un mix énergétique là aussi largement dominé par le nucléaire (86,9%).

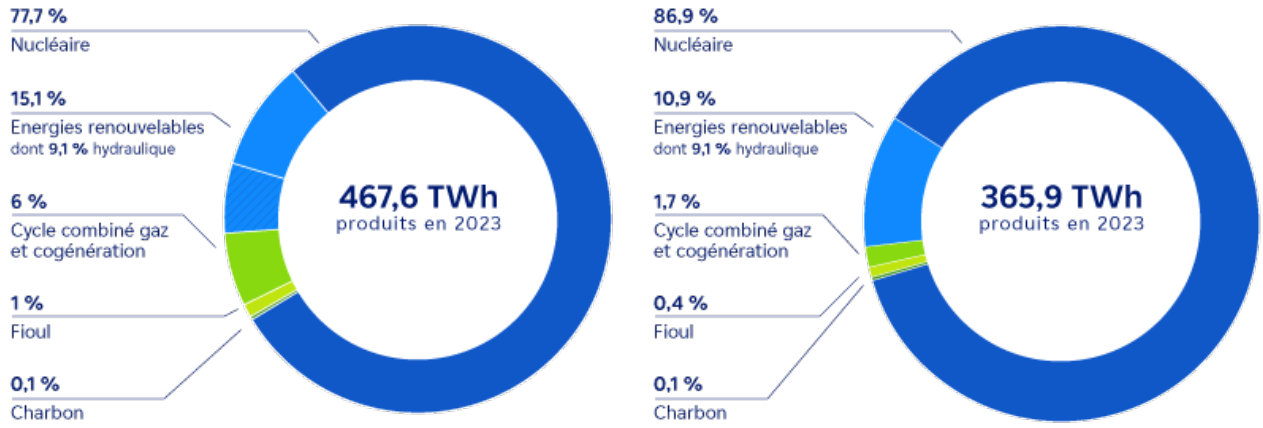

Figure 1 — Mix énergétique d'EDF dans le monde et en France.

Cette très large proportion du nucléaire en France s'explique par le manque de ressources énergétiques (charbon, gaz et pétrole) en métropole. Pendant la crise pétrolière de 1973, afin de garantir la souveraineté énergétique de la France, le gouvernement de Pierre Messmer met en place un vaste programme de construction de centrales nucléaires.

Aujourd'hui, la France dispose de 57 réacteurs ou « tranches » en exploitation répartis sur 18 sites de production. Chaque site de production est appelé un Centre Nucléaire de Production d'Electricité (CNPE). Tous les réacteurs actuels sont issus de la filière à eau pressurisée et ont une puissance électrique de 900 à 1600 MW selon les différents paliers $\underbrace{\text{CP0, CP1, CP2}}_{\text{900 MW}}$, $\underbrace{\text{P4, P'4}}_{\text{1300 MW}}$, $\underbrace{\text{N4}}_{\text{1450 MW}}$ et $\underbrace{\text{EPR}}_{\text{1600 MW}}$.

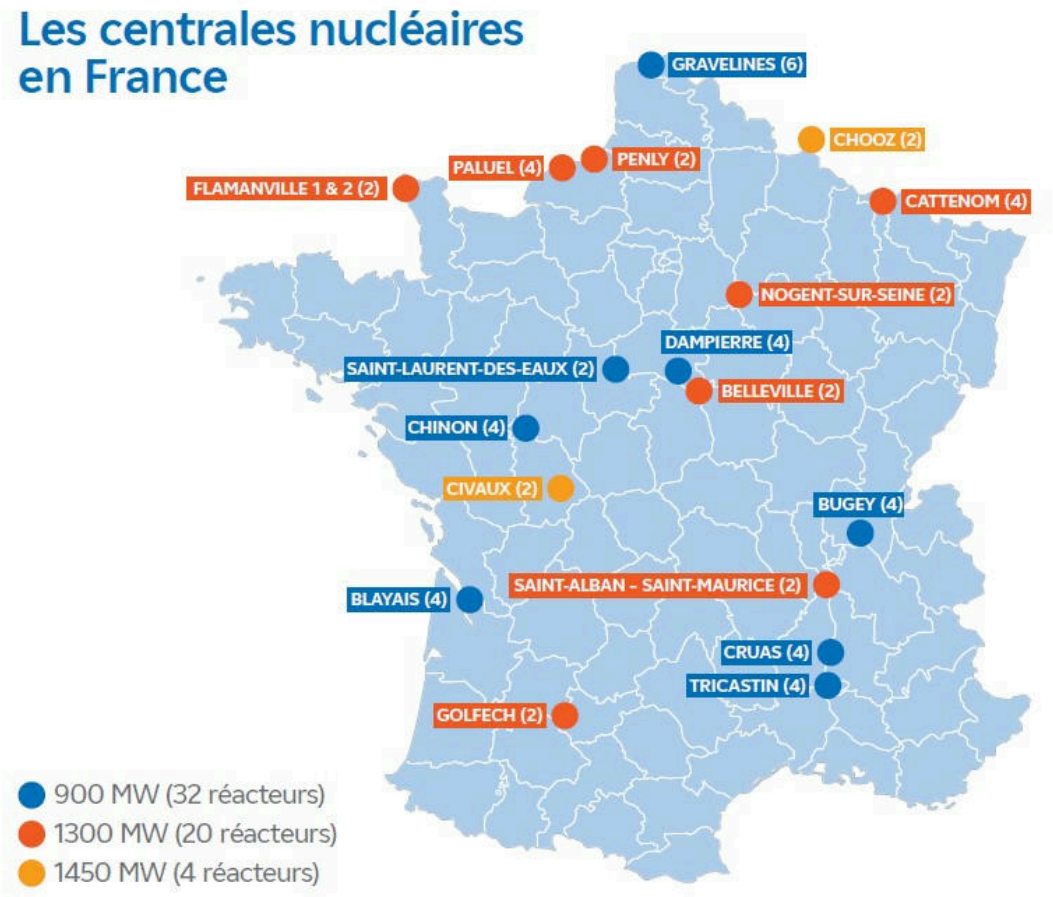

Figure 2 — Carte de la localisation des différents réacteurs en France.

À titre de comparaison, un réacteur de 900 MW produit en moyenne chaque mois 500 000 MWh, ce qui correspond à la consommation de 400 000 foyers environ.

Enfin, EDF participe à de grands projets industriels, tels que :

- La construction de la paire d'Evolutionary Power Reactor (EPR) d'Hinkley Point C et de Sizewell C, d'une puissance de 1600 MW par tranche,
- Le programme EPR2, une commande de 6 nouveaux réacteurs en France, dont la première paire serait construite sur le site de Penly, avec une mise en service prévue pour 2035,
- Le développement de la technologie SMR NUWARD™, une petite centrale d'une centaine de MW comprenant deux réacteurs indépendants hébergés dans un bâtiment nucléaire unique.

Afin de faire face à ces nouveaux défis, l'État Français a entrepris la recapitalisation de l'entreprise et, le 8 juin 2023, sa participation dans le capital d'EDF atteint 100% – il est désormais le seul actionnaire du groupe.

> **Note**
>
> Le 8 mai 2024, après l'autorisation accordée par l'Autorité de Sûreté Nucléaire (ASN), la mise en service de l'EPR de *Flamanville* a été réalisée, portant le nombre de réacteurs français à 57.

4 | Présentation du CNEPE

Au sein de la Direction Ingénierie et Supply Chain (DISC) du groupe EDF, il existe, entre autres, plusieurs centres d'ingénierie :

1. Edvance
2. Le Centre National d'Equipements de Production d'Electricité (CNEPE)
3. EDF EPR Engineering UK

Chaque centre travaille sur des sujets spécifiques. Par exemple, Edvance se concentre sur l'îlot nucléaire des projets de nouveau nucléaire tandis que le CNEPE travaille principalement sur l'îlot conventionnel [2] .

Figure 3 — Le **CNEPE** en chiffres

Plus précisément, les activités du CNEPE se composent de deux grands périmètres d'action.

- **L'ingénierie du parc en exploitation** : Le CNEPE prépare et pilote l'intégralité des modifications techniques sur les installations non nucléaires des centrales. De ce fait, les salariés du CNEPE travaillent sur l'ingénierie de l'îlot conventionnel (salle des machines, groupe turbo-alternateur), source froide, poste d'eau, aéroréfrigérants, etc. Par ailleurs, depuis 2014, EDF et le CNEPE sont engagés dans le « Grand Carénage », un programme industriel de rénovation et de modernisation des différentes centrales nucléaires existantes, avec pour objectif l'augmentation de la durée de vie des centrales (de 40 à 60 ans) et l'amélioration de la protection des sites, en particulier le respect des exigences post-*Fukushima*.
- **Le développement des projets Nouveau Nucléaire** : Regroupés sous 8 projets, le *Nouveau Nucléaire* vise à lancer les programmes de construction de réacteurs en France (Flamanville 3, EPR2, SMR) et à l'international (Hinkley Point C, Sizewell C, JNPP, EPR 1200, EXPORT).

En outre, pour répondre à ces objectifs, le centre d'ingénierie est découpé en trois départements :

[2] Le fonctionnement d'une centrale nucléaire de la filière Réacteur à Eau Pressurisée (REP) sera détaillé par la suite dans la Section 5.1.

1. Département Ingénierie du parc en exploitation (DIPE)
2. Département Projets Nouveau Nucléaire (DPNN)
3. Département Etudes (DETU)

Les deux premiers départements sont des pôles « projets » et découlent logiquement des périmètres d'action du CNEPE évoqués ci-dessus, tandis que DETU est un département transverse comprenant les métiers : Génie Civil, Fonctionnement, Contrôle Commande, etc.

4.1 | Le groupe FGP

Le Département Etudes du CNEPE compte plus de 750 ingénieurs et techniciens, soit plus de la moitié de l'effectif total du centre d'ingénierie, et compte de nombreux services, visibles ci-dessous.

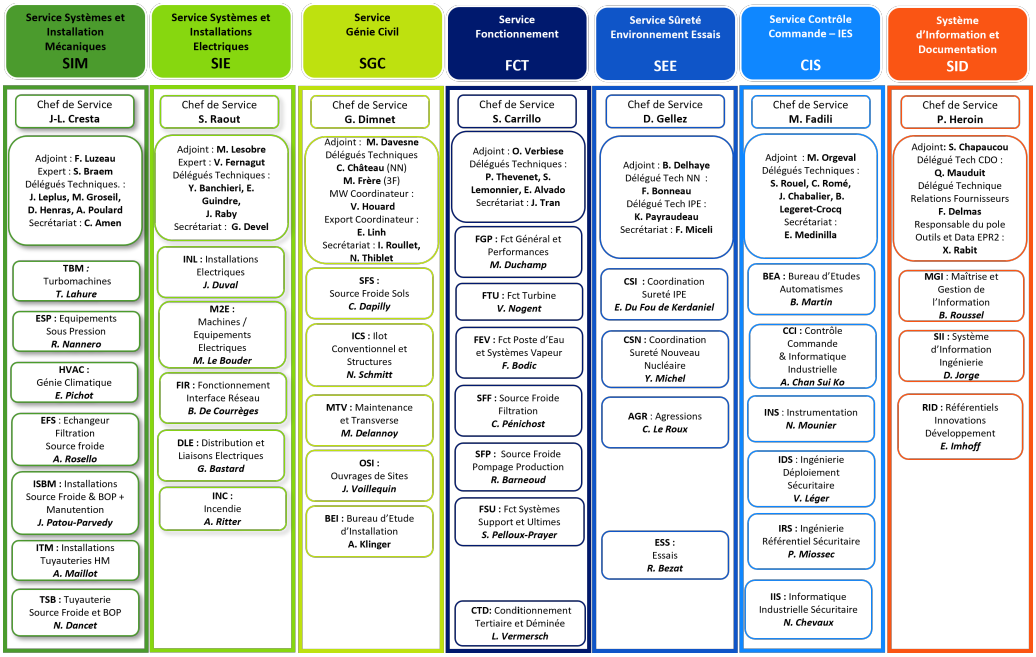

Figure 4 — Services et groupes du Département ETUDES du CNEPE

Pour ma part, j'ai réalisé mon stage au sein du service **FCT** et du groupe Fonctionnement Général et Performance (FGP), dont l'objectif est le suivant :

> *Assurer la cohérence d'ensemble de l'îlot conventionnel des tranches nucléaires pour un fonctionnement performant.*
>
> — Groupe FGP

Une des fonctions du groupe FGP est de travailler sur l'intégration des systèmes de la Salle Des Machines (SDM), avec comme responsabilité le maintien de la cohérence globale de l'installation et le pilotage des pôles systèmes. Les missions principales comprennent la définition et l'optimisation des performances générales de la SDM. Le groupe mène également des études fonctionnelles transverses, qui nécessitent une coordination technique et une attention particulière à la sûreté. De plus, des études de fonctionnement sur les grands transitoires [3] comme un Arrêt Automatique Réacteur (AAR) ou un îlotage [4] , sont réalisés.

[3] Un grand transitoire est un intervalle de temps très court pendant lequel la centrale subit une variation rapide de puissance.

[4] Un îlotage consiste à isoler brutalement une tranche du réseau électrique à un niveau de puissance permettant l'équilibre de la tranche avec alimentation des auxiliaires par la turbine (autoconsommation).

4.1.1 | Aspect Economique de la Performance

Le groupe FGP traite aussi des implications économiques liées à la performance des tranches en exploitation. Définissons quelques termes avant d'exposer les enjeux.

Sûreté (nucléaire) : L'ensemble des dispositions techniques et des mesures d'organisation relatives à la conception, à la construction, au fonctionnement, à l'arrêt et au démantèlement des installations nucléaires de base, ainsi qu'au transport des substances radioactives, prises en vue de prévenir les accidents ou d'en limiter les effets.

Taux de Disponibilité : Ratio entre l'énergie qui aurait pu être produite par la capacité disponible pendant une période donnée et l'énergie qui aurait pu être produite (pendant la même période) à la puissance de référence d'une tranche.

Performance : Maximisation de l'énergie produite par tranche et optimisation de la consommation de combustible pour le MWh produit.

* * *

Dans un CNPE, **la sûreté de l'installation prime sur tout**. Il ne faut en aucun cas dégrader la sûreté pour augmenter le taux de disponibilité de la centrale ou la rendre plus performante. Plus précisément :

$$\text{Sûreté} > \text{Taux de Disponibilité} > \text{Performance}$$

La performance est donc l'aspect le moins prioritaire ; néanmoins, pour donner un ordre d'idée, avec un prix moyen de 50 € du MWh [5] , un MW électrique non produit correspond globalement à ~ 400.000 € d'euros de manque à gagner sur une année pour EDF [6] .

Sur les tranches en exploitation, il peut y avoir quelques MW non produits dus à des problèmes matériels. Le groupe FGP a pour mission d'appuyer les sites de production dans leurs objectifs de récupération de ces MW.

[5] En réalité, la moyenne des prix spot de l'électricité (prix établis sur le marché de l'électricité par les bourses le jour J pour le lendemain) est actuellement autour de 70 € du MWh. **Source** : https://www.rte-france.com/eco2mix/les-donnees-de-marche.

[6] Plus précisément, avec un prix spot de 50 € du MWh, on a : $50 \times 24 \times 365 = 438.000$ €

5 | La filière nucléaire en France

En octobre 1945 est crée le Commissariat à l'Energie Atomique (CEA) pour lancer la recherche et l'industrialisation de l'énergie nucléaire en France. Onze ans plus tard, le premier réacteur nucléaire français de type Uranium Naturel Graphite Gaz (UNGG) est mis en service à Marcoule par le CEA pour des usages militaires. Pour EDF, c'est en 1962 que l'aventure du nucléaire à usage civil débute, avec la mise en service du réacteur *EDF1* à *Chinon*. En tout, six réacteurs de la filière UNGG ont été construits en France par EDF, avec une dernière mise en service d'un réacteur dit de « première génération » d'une puissance de 540 MW à *Bugey* en 1972.

Figure 5 — Réacteur UNGG de *Bugey-1*

Le choix de la filière UNGG s'explique par des contraintes d'indépendance nationale – le graphite et l'uranium naturel étaient les deux seules ressources dont la France était sûre de disposer rapidement en grandes quantités sans avoir recours à des importations.

Au cours des années 1970, les avancées en matière d'enrichissement de l'uranium sur le sol français ouvrent la voie à une filière plus sûre et plus compétitive : celle du Réacteur à Eau Pressurisée, dit de « deuxième génération », qu'EDF compte exploiter. Surtout, le contexte international de l'époque (la guerre du *Kippour*) entraîne un grave choc pétrolier qui va considérablement accélérer le déploiement à l'échelle industrielle et standardisée de ces centrales REP.

Mise en service en 1977, la centrale de *Fessenheim* est la tête de série des centrales REP françaises. Aujourd'hui à l'arrêt, elle disposait d'une puissance de 900 MW par tranche, la rendant beaucoup plus puissante que les centrales issues de la filière obsolète UNGG. En tout, pas moins de 34 tranches de 900 MW (paliers CP0/1/2) seront mises en service entre 1977 et 1988, une véritable réussite industrielle pour EDF. Aujourd'hui, 56 réacteurs de génération II sont encore en exploitation, ainsi qu'un réacteur de génération III, l'EPR de *Flamanville*.

5.1 | Présentation d'une centrale REP

Une centrale REP possède trois circuits indépendants étanches et utilise la chaleur dégagée par la réaction de fission nucléaire pour chauffer de l'eau pressurisée et, au final, faire tourner une turbine puis entrainer un alternateur pour produire de l'électricité.

Une centrale REP dispose par ailleurs de plusieurs bâtiments :

- **Le Bâtiment Réacteur (BR)**, qui contient l'ensemble du circuit primaire, une partie du circuit secondaire et les assemblages de combustible dans la cuve du réacteur, où se produit la réaction de fission nucléaire.
- **Le Bâtiment Combustible (BK)**, dans lequel les assemblages de combustible neufs sont stockés avant leur chargement dans le cœur, et les assemblages usés, qui sont entreposés dans des piscines de désactivation avant leur envoi pour retraitement à l'usine de *La Hague*.
- **La Salle Des Machines (SDM)**, qui abrite le groupe turbo-alternateur, le condenseur, suivi de turbopompes alimentaires et les locaux périphériques d'exploitation.

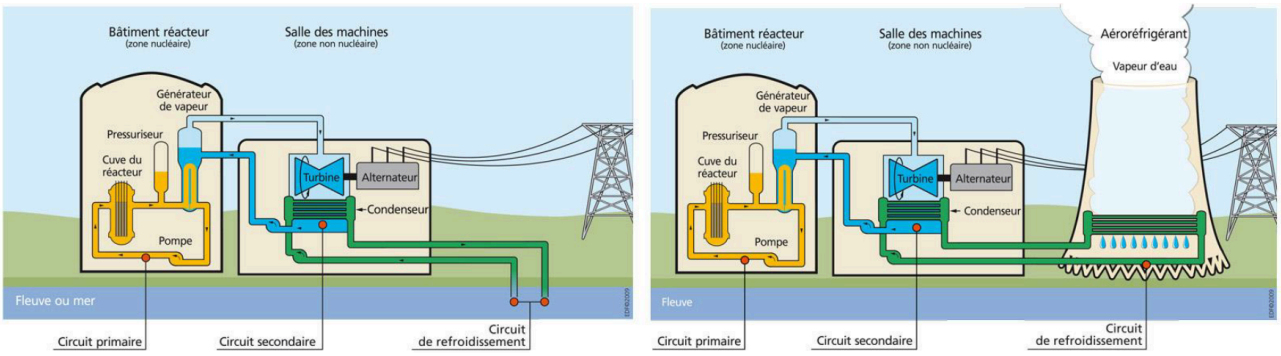

Figure 6 — Schéma simplifié en coupe d'une tranche en circuit ouvert/semi-ouvert

5.1.1 | Le circuit primaire

Le circuit primaire est la source chaude de l'installation, il est installé dans une enceinte de confinement, le BR. L'eau du circuit primaire se réchauffe au contact des éléments combustibles en traversant la cuve du réacteur. Cette eau chaude est ensuite dirigée vers des générateurs de vapeur (3 ou 4 selon les paliers) dans lesquels elle cèdera son énergie. L'eau est ensuite renvoyée dans le réacteur grâce à des motopompes. D'autre part, le circuit primaire est maintenu sous pression par un pressuriseur unique pour éviter la vaporisation de l'eau.

5.1.2 | Le circuit secondaire

Lorsque la centrale est en production, ce circuit représente la source froide du circuit primaire. L'eau du circuit secondaire se vaporise au contact des tubes des générateurs de vapeur. Cette vapeur est dirigée par des tuyauteries hors du bâtiment réacteur et va entraîner l'arbre de la turbine auquel elle est solidaire, ainsi que l'alternateur. Après que la vapeur ait cédé son énergie à la turbine, celle-ci est recueillie dans le condenseur pour y être refroidie et condensée. L'eau est ensuite extraite du condenseur par des pompes qui l'injectent dans les postes d'eau basse pression puis haute pression pour y être réchauffée avant de repartir dans les générateurs de vapeur.

5.1.3 | Le circuit tertiaire

Ce circuit est la source froide du circuit secondaire lorsque celui-ci est en production. Son rôle est d'extraire les calories encore présentes dans la vapeur recueillie après la turbine dans le condenseur. L'eau circule dans des tubes placés à l'intérieur du condenseur. Selon l'implantation des centrales, ce circuit se présente sous des formes différentes :

- Circuit ouvert pour les centrales de bord de mer,
- Circuit ouvert ou semi-ouvert pour les centrales de bord de fleuve.

> **Note**
>
> Les aéroréfrigérants n'équipent que les centrales en bord de fleuve quand la source froide n'a pas un débit suffisant lui permettant d'évacuer la chaleur du Circuit de Refroidissement après son passage dans les conduites forcées du condenseur.

5.1.4 | L'évacuation d'énergie

Lorsque l'alternateur est mis en rotation par la turbine et soumis à un champ magnétique, cela créé un courant qu'il faut évacuer vers le réseau. A la sortie de l'alternateur on trouve un disjoncteur qui permet de coupler et de découpler l'installation du réseau. L'électricité est ensuite dirigée sur des transformateurs principaux pour élever la tension.

Dans la section précédente, il était indiqué que la sûreté de l'installation était l'aspect le plus important dans le fonctionnement d'une centrale REP. Celle-ci est assurée par trois grandes fonctions de sûreté appuyée par différents équipements.

1. **Refroidir le combustible en tout temps**
   - Pompe en service
   - Fluide caloporteur (eau)
   - Source froide
2. **Maîtriser la réactivité**
   - Grappes de contrôle
   - Eau borée
3. **Confiner les produits de fission**
   - La gaine du combustible
   - Le circuit primaire principal
   - Bâtiment Réacteur (enceinte de confinement)

5.2 | Visite de la centrale de Dampierre

Pendant mon stage, j'ai eu la possibilité de visiter le site de production nucléaire de *Dampierre*. Une chance exceptionnelle de découvrir le fonctionnement de ce type d'installation.

5.2.1 | Quelques informations sur la centrale

La centrale nucléaire de *Dampierre* est située dans le département du Loiret, à 50 kilomètres environ d'*Orléans*. Elle dispose de quatre réacteurs de 900 MW électriques, mis en service entre 1980 et 1981 et fait partie du palier CP1. La centrale fonctionne en circuit semi-ouvert, avec 4 aéroréfrigérants, et sa source froide est la Loire.

Figure 7 — Centrale Nucléaire de Dampierre

5.2.2 | Déroulement de la visite

Les accès dans un CNPE sont très réglementés, et une enquête est effectuée avant l'arrivée sur site. A l'intérieur de la centrale, un important dispositif de sécurité est mis en place (barbelés, portiques, capteurs biométriques, etc.) permettant de protéger les différentes zones de l'installation contre les intrusions.

Une fois entrés dans le périmètre de la centrale, nous avons commencé par visiter l'aéroréfrigérant de la tranche 1 à l'arrêt. Entièrement fait de béton et culminant à plus de 170 mètres de hauteur, un aéroréfrigérant est une installation dont le génie civil est particulièrement impressionnant.

* * *

Figure 8 — Aéroréfrigérant de la tranche à l'arrêt

Nous avons ensuite visité la Salle Des Machines de la centrale, qui, particularité des paliers CP0 et CP1, est commune aux 4 tranches et accolée devant les Bâtiments Réacteurs. Nous avons amorcé la visite au niveau des poumons condenseurs pour progressivement arriver au niveau du plancher turbine, à 15 mètres au-dessus du sol. Quelques points m'ont marqué :

- La compacité et l'enchevêtrement des équipements (pompes d'extraction, réchauffeurs, etc.) des étages inférieurs,
- La propreté remarquable de la SDM pour une installation qui a déjà plus de 40 ans à son actif.

Figure 9 — Salle des machines (plancher turbine)

Dans la figure ci-dessus, on aperçoit le groupe turbo-alternateur, d'une longueur de 70 mètres, composé d'un corps Haute Pression (HP), trois corps Basse Pression (BP) et d'un alternateur en bout d'arbre. Pour limiter la propagation des vibrations vers le plancher turbine, le groupe turbo-alternateur est posé sur une dalle en béton indépendante, la table de groupe.

En outre, le plancher turbine est organisé pour permettre le stockage des principaux composants de la turbine en phase de maintenance, à l'image de ce rotor de corps BP.

Figure 10 — Rotor d'une Turbine BP

Un pont principal et un pont auxiliaire permettent les manutentions de montage et de maintenance de ces composants allant parfois jusqu'à plusieurs centaines de tonnes. D'autre part, l'accès aux matériels des niveaux inférieurs se fait au travers de trémies.

Enfin, nous avons terminé la matinée devant la Salle de Commande (SDC) de la tranche 3. Cette salle de commande standardisée (modèle 900) est organisée autour d'un pupitre principal où sont rassemblés tous les moyens de commande et d'information nécessaires à la conduite normale, et d'un tableau arrière pour les opérations de démarrage depuis l'arrêt à froid et en situation accidentelle. Surtout, elle ne dispose presque pas d'outils numériques.

Ma dernière impression fut celle-ci : la décoration rétro-futuriste des années 1970 en SDC a pris un sacré coup de vieux.

* * *

6 | Problématiques du stage

Au cours de mon stage, mes missions ont portées sur deux aspects distincts du domaine nucléaire : le contrôle-commande et les simulateurs. J'ai ainsi pu travailler sur la refonte d'un outil de perte d'armoires de contrôle-commande et sur une application de post-traitement de résultats de simulation. Dans le cadre de mes développements, **R**, **python** et **SQL** ont été utilisés.

6.1 | HPConnect

> **Note**
>
> HPConnect est une application d'analyse de données dediée à la perte d'armoires de contrôle-commande standard pour la centrale EPR d'*Hinkley Point C*. La fonctionnalité principale de cette application est de pouvoir générer des fiches au format Word sur les impacts fonctionnels et sûretés liés à la perte d'une armoire. Ces fiches sont un template de base, elles sont utilisées par le CNEPE et par Edvance pour réaliser les analyses des conséquences fonctionnelles pour le compte du client Nuclear New Build (NNB), une filiale d'*EDF Energy*.
>
> Elles répondent à deux besoins :
>
> - **La gestion des évènements fortuits**, avec pour objectif d'aider l'opérateur à appréhender les conséquences de la perte d'une armoire d'un point de vue fonctionnel et sûreté,
> - **La préparation à la maintenance**, où ces fiches servent de support à l'analyse de risque préalable à la maintenance.

6.1.1 | La centrale EPR d'Hinkley Point C

La centrale nucléaire d'*Hinkley Point* est installée près de *Bridgwater* au Royaume-Uni. Elle comporte en tout 4 tranches définitivement arrêtées, dont les 2 plus anciennes sont actuellement en cours de démantèlement. Depuis 2018, 2 autres tranches EPR d'une puissance de 1600 MW chacune *(Hinkley Point C)* sont en construction, avec une mise en service prévue pour 2029.

Figure 11 — Bâtiment Réacteur en construction

6.1.2 | Architecture I&C

Une centrale nucléaire nécessite la mise en place de moyens particuliers permettant de contrôler le processus physique de fission au sein de l'îlot nucléaire et le bon fonctionnement de l'îlot conventionnel : mesures, régulations, automatismes, etc. Ces moyens sont regroupés dans une architecture d'Instrumentation et Contrôle-Commande (I&C), présentée de manière conceptuelle ci-dessous.

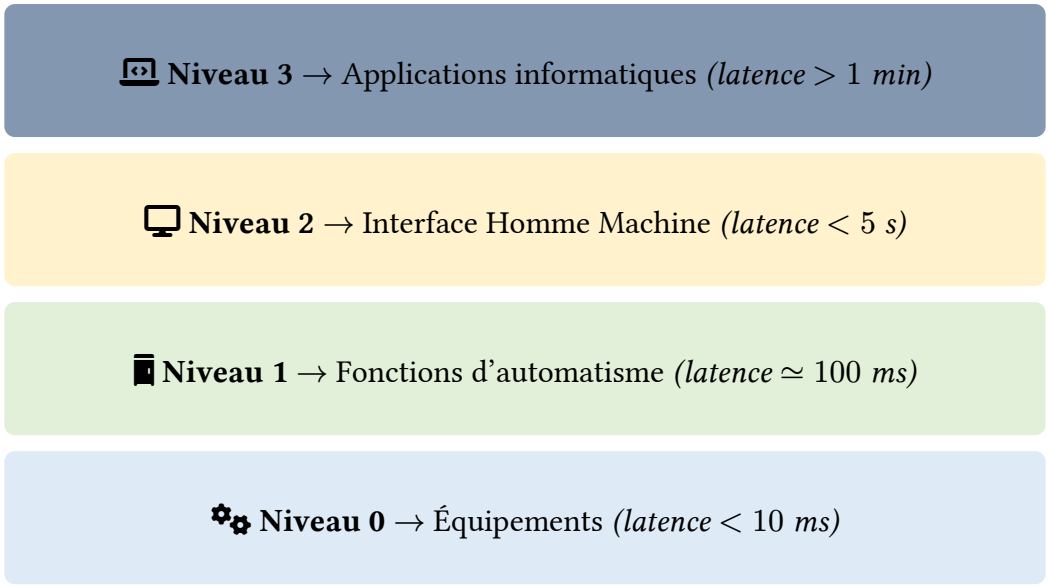

Figure 12 — Base d'une architecture I&C

- **Le niveau 0** regroupe les équipements du procédé, comme les capteurs ou les actionneurs. Ces équipements servent à la prise d'information ou à l'action sur le procédé.
- **Le niveau 1** constitue le niveau central. Il regroupe les équipements réalisant l'acquisition des informations issues des capteurs, le calcul des automatismes logiques et des régulations. Le niveau 1 génère les commandes à destination du niveau 0.
- **Le niveau 2** regroupe les équipements permettant de réaliser les fonctions de conduite et de supervision de l'unité de production. Ce niveau reçoit les ordres des opérateurs de conduite et les informe sur l'état de l'installation en Salle de Commande. Le niveau 2 échange des signaux avec le niveau 1.
- **Le niveau 3** permet le traitement et le stockage de l'information, mais sans action directe sur le procédé.

6.1.3 | L'armoire de contrôle-commande SPPA-T2000

Au sein de l'achitecture présentée dans la Section 6.1.2, les armoires de contrôle-commande occupent une place centrale *(le niveau 1)*, et constituent la plateforme d'automatisme pour les systèmes Plant Automation System (PAS) et Safety Automation System (SAS).

L'armoire de contrôle-commande standard déployée à *Hinkley Point C* est déclinée en plusieurs appellations selon la fonction de l'armoire :

KCM → Îlot conventionnel et distribution électrique ~ 22 armoires (CNEPE),

KCO → Îlot nucléaire ~ 84 armoires (Edvance),

KCQ → Bâtiment de traitement des effluents ~ 5 armoires (CNEPE).

Ces armoires disposent des principaux composants suivants :

- Des cartes d'entrée filaires permettant d'interfacer l'armoire avec les capteurs, les actionneurs et le Moyen de Conduite de Secours (MCS).
- Un processeur/unité de traitement de l'automate.
- Des racks de connexion réseau.

6.1.4 | CabinetLoader

A mon arrivée, l'application *HPConnect* disposait déjà d'une première version développée sur R, mais les sources de données trop nombreuses et leur non-standardisation (notamment des fichiers csv dont la provenance était assez obscure) impliquait des difficultés dans la mise à jour des fiches.

Face à ce constat, ma stratégie a été de remonter aux sources « brutes » et de construire le package R [7] `{cabinetloader}`. Ce package permet, grâce à des fonctions simples, de mettre à jour les données pour leur ingestion dans l'application *HPConnect*.

Pour réaliser ce travail, je me suis largement appuyé sur les packages du **tidyverse** tels que :

- `{dplyr}` pour les opérations de manipulation et de remodelage de données,
- `{janitor}` pour uniformiser le nom des colonnes,
- `{purrr}` pour remplacer les boucles **for** par des fonctions d'itération optimisées.

6.1.4.1 | Les sources de données

Le package `{cabinetloader}` dispose de 5 sources de données principales. La différence la plus notable en terme de standardisation concerne les datapackages, des dossiers *zip* organisés par systèmes élémentaires.

1. **Hardware Signals List (HSL)** → Liste qui représente l'ensemble des entrées/sorties hardware,
2. **Software Signals List (SSL)** → Liste qui représente l'ensemble des entrées/sorties software,
3. **TEC4** → Liste des segments issus de l'unité de traitement d'une armoire,
4. **Datapackages Detailed Functional Diagrams (DFD)** [8]
   - **Set IO** → Répertorie toutes les informations échangées au sein d'un système donné,
   - **External IO** → Répertorie tous les échanges externes d'un système donné,
   - **STE Parameters** → Caractéristiques et paramètres de fonctionnement des équipements,
   - **Fallback Values** → Définit des valeurs de repli pour les signaux échangés,
5. **IHM** → Liste reliant composants et images de conduite.

[7] En prenant en compte les objectifs du projet (génération d'un document computationnel au format Word), les contraintes de l'environnement de déploiement, et le code source déjà existant, R s'est avéré être le meilleur choix.

[8] Les DFD sont les schémas logiques décrivant l'ensemble des fonctions de contrôle-commande d'un système élémentaire.

6.1.4.2 | Architecture du package

L'objectif pour l'utilisateur final est de disposer d'un moyen pour mettre à jour les données facilement à partir de **RStudio**. Pour cela, le script `install.R` va installer toutes les dépendances du package et charge `{cabinetloader}` dans l'Integrated Development Environment (IDE). Une fois chargé, plusieurs options s'offrent à l'utilisateur.

Si on souhaite mettre à jour uniquement certaines données, on peut le faire avec les fonctions de type `*_data_writer`. Le fonctionnement interne de ces fonctions est décrit dans le schéma ci-dessous.

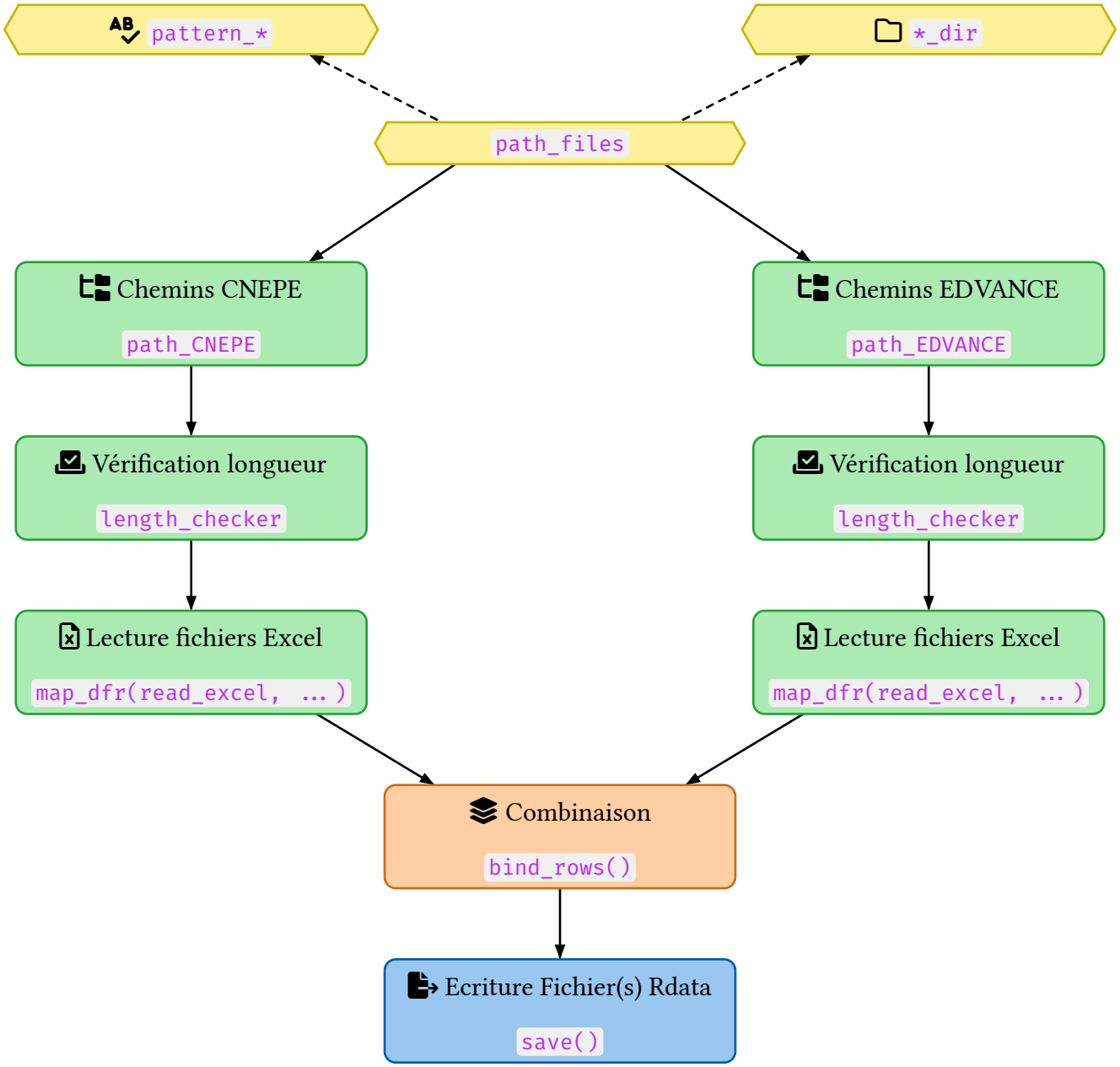

Figure 13 — Fonctionnement interne des fonctions `*_data_writer` pour les datapackages

La fonction `path_files` utilise un *pattern* (expression régulière) et un dossier (datapackages CNEPE/Edvance dézippés) en arguments d'entrée pour filtrer les fichiers présents dans le dossier en fonction du *pattern* choisi — pour récupérer les données Set IO, il existe un *pattern* Set IO spécifique, etc.

Par exemple, pour mettre à jour les données de **Fallback Values** et donc récupérer les valeurs de repli pour les signaux émis, il suffit d'utiliser la fonction `fallback_values_data_writer()`, qui ne nécessite aucun argument.

D'autre part, si on souhaite mettre à jour l'ensemble des données, il faut utiliser la fonction `update_all()`, qui va se charger d'éxécuter les fonctions de mise à jour dans l'ordre suivant :

TEC4 → SSL → HSL → IHM → STE → External IO → Set IO → Fallback Values → Main Datasource

Toutes ces tables intermédiaires créées avant la table de données principale sont nécessaires au bon fonctionnement de la fonction `main_data_join()`, dont le rôle est de combiner les sources HSL, SSL et TEC4 en une seule table regroupant l'ensemble des signaux. Celle-ci est ensuite enrichie grâce à des jointures avec les données IHM pour récupérer les imageries de conduite, les paramètres STE pour récupérer la position sur perte UA (perte de tension automate) des actionneurs, les données External IO pour récupérer le type du signal (filaire/bus) puis la combinaison des données Set IO et Fallback Values pour récupérer les valeurs de repli des signaux.

Enfin, grâce au package `{cli}`, des messages informatifs sur l'avancement et le temps de mise à jour sont fournis à l'utilisateur.

```
-- TEC4 ------------------------------------------------------------------------
i Tâche en cours : import des fichiers HTML TEC4.
v Longueur des chemins de fichiers OK. RAS
v Ecriture du fichier data/tec4.RData effectuée.

-- SSL -------------------------------------------------------------------------
i Tâche en cours : import des fichiers SSL.
v Longueur des chemins de fichiers OK. RAS
--------------------------------------------------------------------------------
i Récursion --- Rebond 1
i Récursion --- Rebond 2
i Récursion --- Rebond 3
i Récursion --- Rebond 4
i Récursion --- Rebond 5
i Récursion --- Rebond 6
--------------------------------------------------------------------------------
i Association des AS Name aux Codes armoire.
v Ecriture du fichier data/ssl.RData effectuée.

-- HSL -------------------------------------------------------------------------
i Tâche en cours : import des fichiers HSL.
v Longueur des chemins de fichiers OK. RAS
v Ecriture du fichier data/hsl.RData effectuée.

-- IHM -------------------------------------------------------------------------
i Tâche en cours : import du fichier de lien IHM.
i Un seul fichier IHM : OK.
v Longueur des chemins de fichiers OK. RAS
v Ecriture du fichier data/ihm.RData effectuée.

-- STE -------------------------------------------------------------------------
i CNEPE - Tâche en cours : import des fichiers STE.
v Longueur des chemins de fichiers OK. RAS
 ➢──────────────────────────────    1% |  ETA:  6m
```

6.1.4.3 | Documentation du package

Des efforts importants ont été mis en oeuvre pour éviter l'effet « boîte noire » et pour décrire les choix d'implémentation mais surtout la logique métier spécifique. Pour ce faire, deux packages ont été utilisés : `{roxygen2}` et `{pkgdown}`.

`{roxygen2}` permet d'écrire la documentation directement dans les fichiers source R à l'aide d'un squelette de commentaires au-dessus d'une fonction. En étant située au plus près du code, la documentation facilite la compréhension, la lisibilité et la maintenance pour les futurs contributeurs. De plus, la documentation est générée automatiquement et transformée en fichiers **.Rd** dans le dossier **man** [9] du package, ce qui simplifie le processus en évitant d'avoir à maintenir une documentation séparée et désynchronisée du code.

`{pkgdown}` permet quant à lui de générer automatiquement un site web customisable de documentation statique à partir des fichiers **.Rd**. Ce site inclut la liste complète des fonctions, leur documentation détaillée et des exemples d'utilisation.

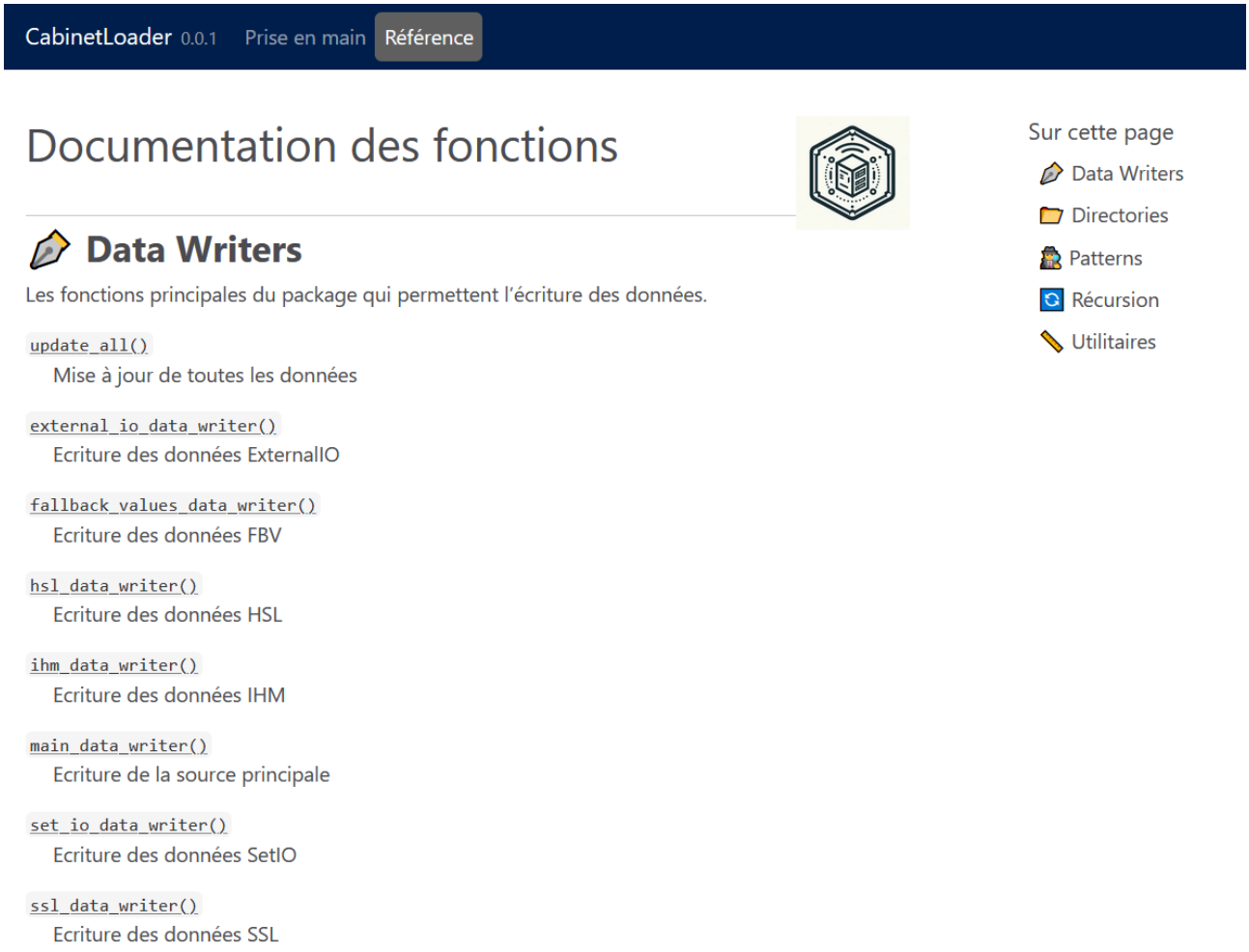

Figure 14 — Référence des fonctions du package `{cabinetloader}`

6.1.5 | L'application HPConnect

HPConnect est déployée sur une machine virtuelle hébergée au CNEPE, accessible à tous les utilisateurs potentiels qui souhaitent générer des fiches chez Edvance et au CNEPE.

[9] Dossier de documentation standard d'un package R.

> **Important**
>
> Par rapport à une rédaction manuelle d'une fiche de perte d'armoire de contrôle-commande, l'application permet un gain de temps considérable, estimé à **40 heures de travail économisé par fiche**. *HPConnect* permet en effet aux ingénieurs rédigeant les fiches de se concentrer sur les conséquences fonctionnelles de la perte d'une armoire plutôt que sur les pertes détaillées par système, tout en évacuant les potentielles erreurs de traitement humaines et en respectant le formalisme requis.
>
> Pour rappel, il est mentionné dans la Section 6.1.3 qu'au total, 111 fiches sont à générer pour le client. L'outil permet donc d'économiser autour de 4400 heures de travail, soit 2,75 personnes en Equivalent Temps Plein (ETP) [10] , ce qui correspond à 330.000 euros.

Sur un aspect plus technique, *HPConnect* est construite avec `{shiny}`, un **framework** permettant de créer des applications web interactives sur R sans avoir besoin de connaissances approfondies en ingénierie logicielle.

Une application *Shiny* dispose de deux, voire trois parties :

- **La partie** `ui`, qui construit l'interface utilisateur de la page web et définit les éléments visuels (widgets) avec lesquels l'utilisateur peut interagir.
- **La partie** `server`, qui contient la logique de l'application et gère les interactions réactives et les outputs basés sur les input widgets.
- **La partie** `global`, *optionnelle*, qui se charge de définir les variables globales et l'importation des packages.

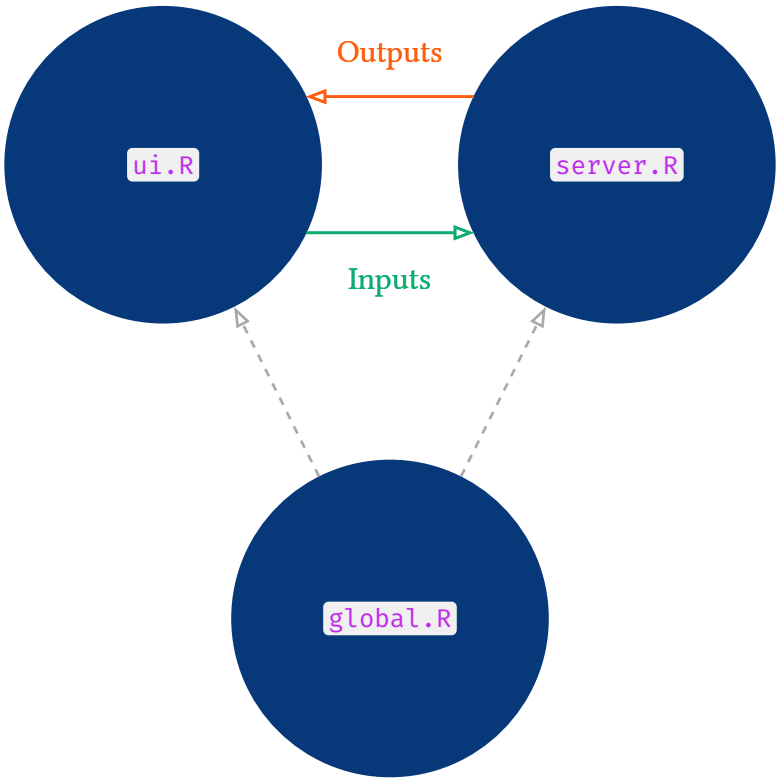

Figure 15 — Interactions des différentes parties d'une application *Shiny*

[10] ETP ≏ 1600 heures par an | Coût ≏ 120k €

> **Avertissement**
>
> Les captures d'écran de l'application sont partiellement masquées par des rectangles noirs afin d'éviter la divulgation d'informations confidentielles.

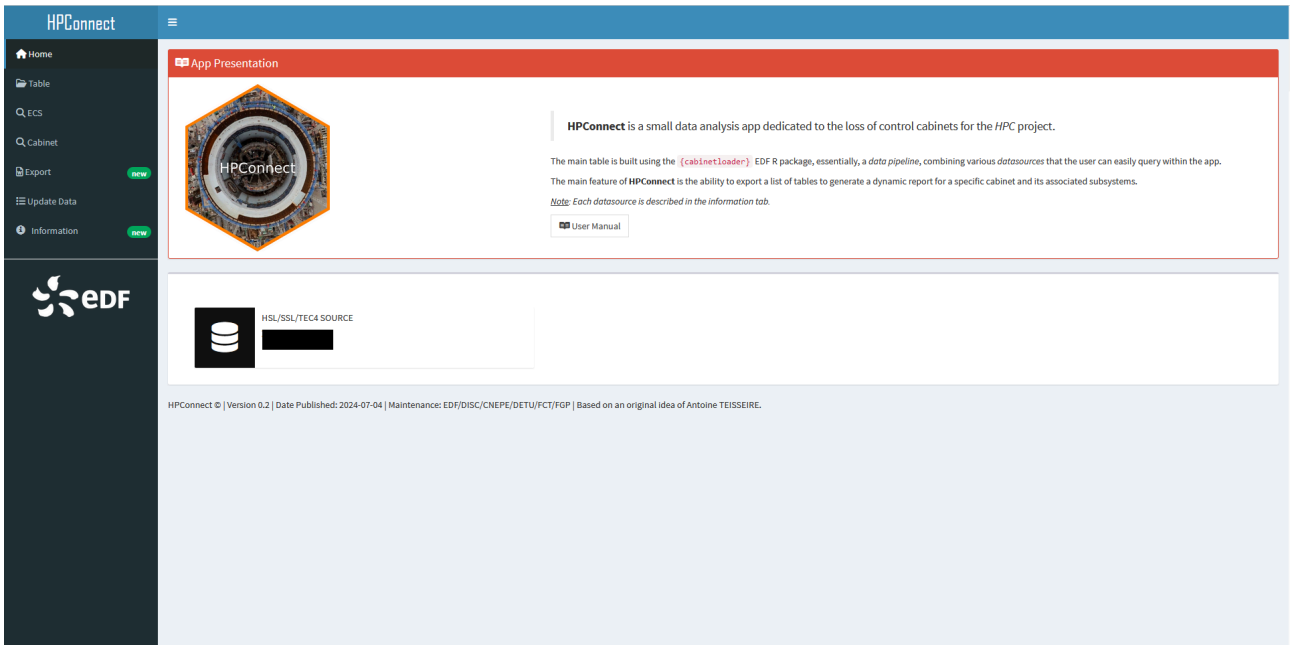

Figure 16 — Onglet Accueil de l'application *HPConnect*

L'onglet d'accueil de l'application décrit brièvement les objectifs de *HPConnect*. La barre latérale permet d'accéder aux autres onglets importants de l'application.

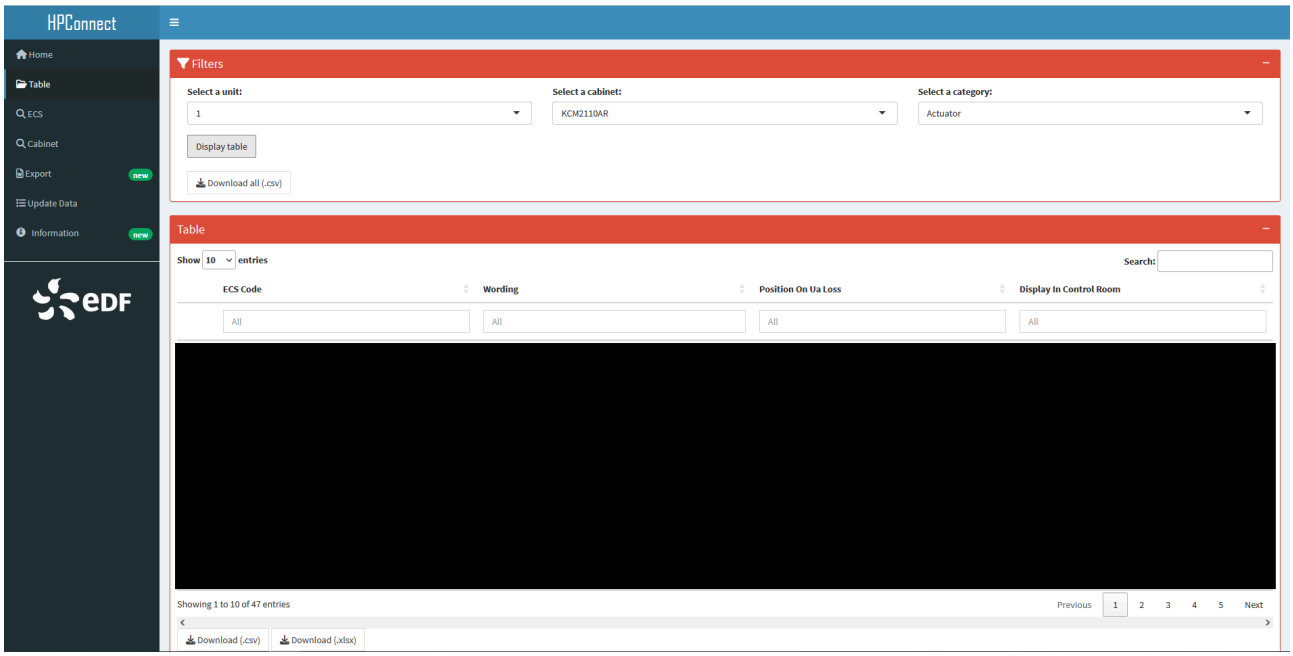

Figure 17 — Onglet Table de l'application *HPConnect*

Ensuite, l'onglet **Table** permet de lister les pertes sur une catégorie pour une armoire donnée : actionneurs, alarmes, commandes groupées, signaux émis, informations analogiques / binaires et capteurs.

L'intérêt de cet onglet est de pouvoir visualiser les données associées à ces catégories pour faire des vérifications à la volée sans avoir à générer une fiche entière pour tous les systèmes d'une armoire.

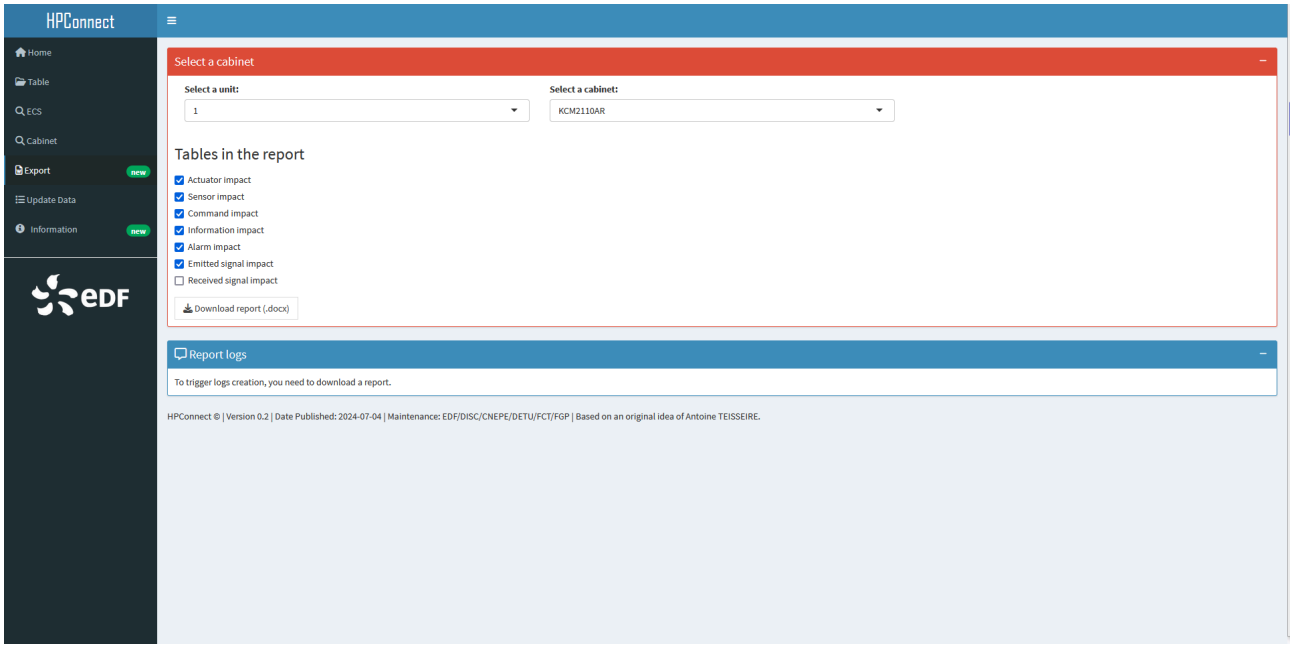

Figure 18 — Onglet Export de l'application *HPConnect*

L'onglet Export est essentiel, il se charge de la génération des fiches au format Word.

En sortie, un rapport par système entièrement paramétré listant les pertes pour une armoire spécifiée est généré. La création de la fiche peut prendre de 20 secondes à 3 minutes selon la volumétrie des données de l'armoire et il est possible de suivre l'avancement de la tâche dans **Report Logs**.

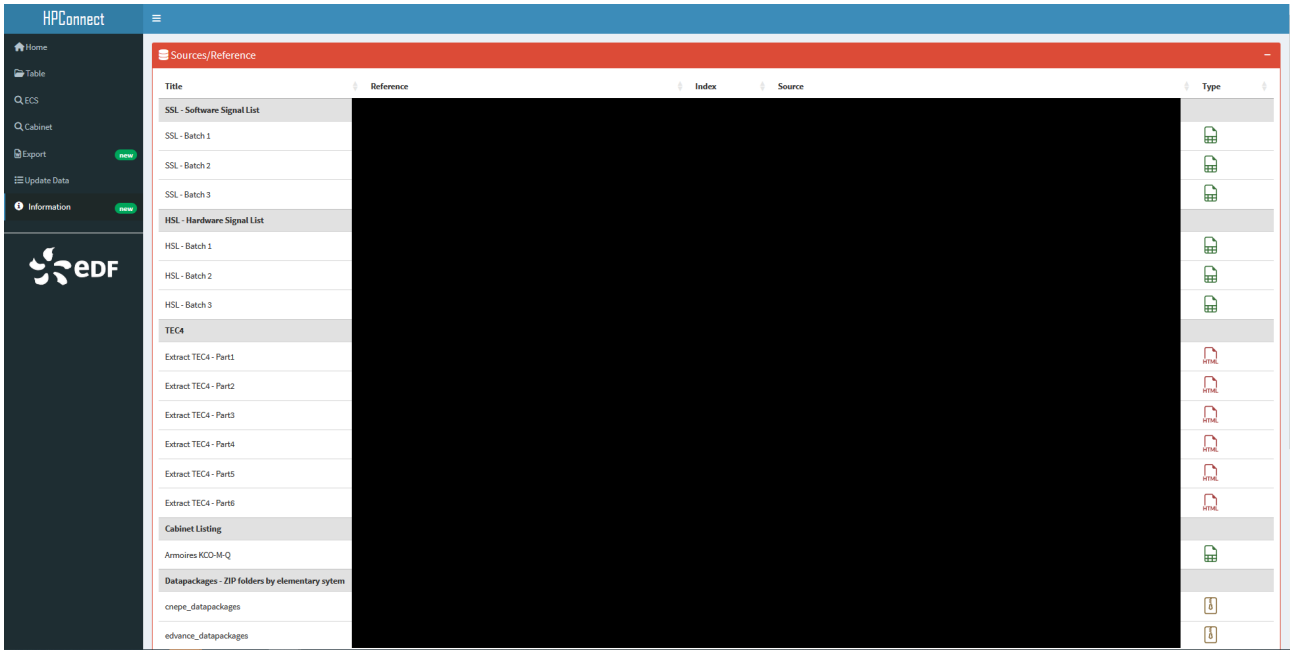

Figure 19 — Onglet Information de l'application *HPConnect*

Enfin, l'onglet **Information** liste l'ensemble des sources de données utilisées dans l'application *HPConnect*, leur format et leur indice de révision.

Loss of cabinet 9KCQ0111AR

1 TABLE OF CONTENTS

2 ABBREVIATIONS

AP	Automation Processor
AS	Automation System
AS name	Reference of automation system run by an AP cabinet
I&C	Instrumentation & Control
KIC	Computerised Operating System
KSC	Main Control Room Instrumentation
KCM	Instrumentation & Control System - Conventional Island & Electrical distribution
KCO	Instrumentation & Control System - Nuclear Island
KCQ	Instrumentation & Control System - Effluent Treatment Buildings
OTS	Operational Technical Specifications
PAS	Processing Automation System
SAS	Safety Automation System
SAS PL	Safety Automation System – Preventive Line
SAS RRC-B	Risk Reduction Category B Safety Automation System
Ua	I&C voltage (loss of I&C cabinet => loss of Ua of actuators associated to I&C cabinet)

3 PURPOSE OF THE DOCUMENT

The aim of this sheet is to describe the functional consequences of the loss of KCQ0111AR I&C cabinet.

A functional analysis per system of the impact of the cabinet loss is performed next. Detailed loss of equipment is available in appendix B, per system and equipment type.

Scope, context, input data, abbreviations and methodology of the analysis are detailed in note [redacted]

3.1 I&C CABINET INFORMATION

Name of the Cabinet: KCQ0111AR
AP or Extension Cabinet: [redacted]
AS Name: [redacted]
Cabinet Technology: [redacted]
Division: [redacted]
Building: [redacted]
Impacted Systems: [redacted]

Figure 20 — Exemple de la partie introductive d'une fiche générée par *HPConnect*

Chaque fiche est structurée de la façon suivante :

- Un tableau listant les abbréviations utilisées dans le document,
- Un court résumé rappelant le contexte de l'analyse,
- Des informations générales sur l'armoire de contrôle-commande (comme sa désignation, le type d'armoire, sa localisation dans la tranche, et enfin les systèmes impactés),
- La perte détaillée des équipements par système (Figure 21).

7 9LKB

7.1 ACTUATOR IMPACT

9LKB Actuator Impact

ECS Code	Wording	Position On Ua Loss	Display In Control Room
[redacted]			

7.2 SENSOR IMPACT

9LKB Sensor Impact

ECS Code	Wording	Display In Control Room
[redacted]		

7.3 COMMAND IMPACT

No impact

7.4 INFORMATION IMPACT

9LKB Analog/Binary Information Impact

ECS Code	Wording	Display In Control Room
[redacted]		

7.5 ALARM IMPACT

9LKB Alarm Impact

ECS Code	Wording	Fallback Value
[redacted]		

7.6 EMITTED SIGNAL IMPACT

No impact

Figure 21 — Perte des équipements détaillée par système

Ici, nous obtenons la perte détaillée du système **9LKB** de l'armoire : actionneurs, capteurs, informations et alarmes sont impactés.

Voici par exemple une fonction générique dont le but est de restituer les données pour les tableaux des actionneurs dans l'application et dans la fiche au format **Word**.

Pour documenter la fonction, `{roxygen2}` a été utilisé. Les intêrets de ce package sont décrits dans la Section 6.1.4.3. Nous ne sommes pas ici dans le cadre de la création d'un package, mais la documentation reste essentielle pour les futurs contributeurs.

actuator.R

```r
#' @title Crée le dataframe des impacts actionneurs
#'
#' @description `actuator` permet de filtrer le dataframe principal préfiltré
#' sur une armoire en fonction des trigrammes des actionneurs.
#'
#' Les positions sur perte Ua (perte de tension automate)
#' sont traduites en anglais pour les notes.
#'
#' *Note* : fonction générique utilisée à la fois dans l'application
#' et dans la génération des rapports Word.
#'
#' @param main_datasource_filtered Le dataframe principal préfiltré
#'
#' @returns L'impact de la perte d'une armoire sur les actionneurs de cette
armoire.
actuator <- function(main_datasource_filtered) {
  list_actuator <- c("1SM", "ACER", "ACT1", "ACT2", "ACT3", "ADA", "ADC",
"ARE", "ARF", "ARP", "FAC", "VME", "VPT")

  actuator <- main_datasource_filtered |>
    filter(typical_diagram %in% list_actuator & type == "HSL") |>
    select(ecs_code, wording, position_on_ua_loss, display_control_room) |>
    mutate(position_on_ua_loss = case_when(
      position_on_ua_loss == "decl" ~ "Tripped",
      position_on_ua_loss == "maint" ~ "Maintained on Initial Position",
      position_on_ua_loss == "open" ~ "Opened",
      position_on_ua_loss == "close" ~ "Closed",
    )) |>
    distinct()

  return(actuator)
}
```

6.2 | ViZiR

> **Note**
>
> ViZiR est une application de post-traitement et de visualisation de données. L'apport majeur de cette application est de pouvoir importer des fichiers csv issus de sessions sur simulateur pour en extraire des graphiques dynamiques, multiaxes et personnalisables.

6.2.1 | Historique de ViZiR

Le simulateur d'une tranche permet d'étudier le comportement dynamique théorique d'une unité de production. Pour que le modèle soit le plus réaliste possible, tous les systèmes de la tranche sont modélisés, la logique contrôle-commande est répliquée et les modèles hydrauliques sont chargés. Le simulateur permet ainsi une représentation immédiate de l'installation et des phénomènes physiques.

Au CNEPE, le groupe FGP a pour rôle de piloter et gérer le développement et l'utilisation des simulateurs. *Grégoire Bouvet* est responsable de ces activités et a supervisé le développement de *ViZiR*.

Les utilisateurs principaux des simulateurs sont les pilotes système, les ingénieurs essai et les ingénieurs fonctionnement général du CNEPE, qui réalisent entre autres des simulations pour vérifier les modifications, le déroulement de procédures d'essai, etc. Une fois la session sur simulateur terminée, il est possible d'exporter les résultats sous forme de fichier csv standardisé avec une colonne de pas de temps en millisecondes et des colonnes avec les valeurs de différents capteurs au pas de temps $t = 100, 300, \ldots$ Un exemple d'export est disponible ci-dessous :

TpsSimu	3AGL7851.TOTFLOW	3GFT4122.iValeur	3ERD9637.Modele_measure
100	81,1125	2175,14	12,0274
300	80,9957	2175,14	12,0273
500	80,4528	2175,14	12,0272
…	…	…	…
5000	79,1539	2175,14	12,0129

Note : Données totalement fictives

Table 1 — Export d'un jeu de données en csv

En utilisant ce fichier, les ingénieurs du CNEPE pouvaient générer des courbes par capteur sur Excel, mais se retrouvaient rapidement confrontés à plusieurs difficultés.

> **Problèmes rencontrés**
>
> - Temps de paramétrage,
> - Les données extraites n'ont pas de métadonnées associées (libellé, unité, etc.),
> - Absence de formattage pour l'axe x,
> - Pas d'axe y secondaire,
> - Un graphique non-dynamique, avec impossibilité de zoomer à un instant t.

ViZiR a donc été imaginé pour répondre à ces problématiques et, dans sa première version, a été développé sous R en utilisant le package `{shiny}`. Ce choix s'explique car un certain nombre d'applications avaient déjà été réalisées en R dans le groupe FGP *(Voir Section 6.1)*.

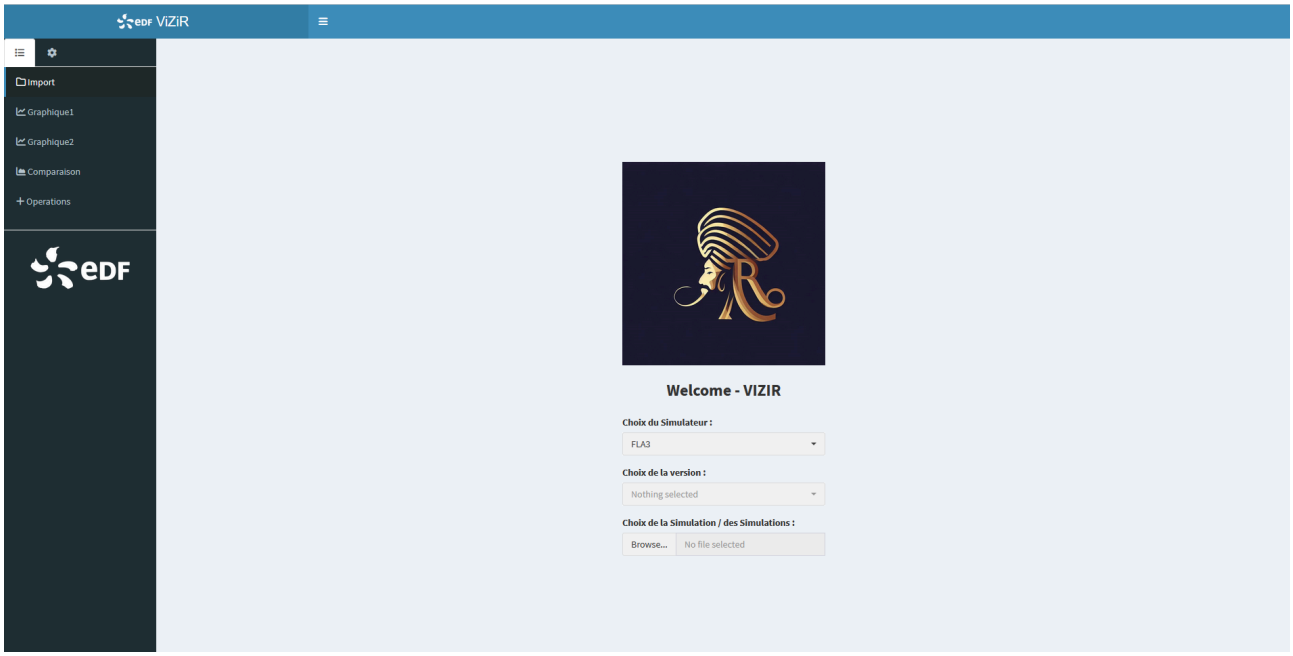

Figure 22 — Interface Utilisateur de *ViZiR 1.0*

Le *workflow* de *ViZiR* est simple, et il suffit de quelques étapes pour visualiser les résultats de simulation.

Choix du simulateur
↓
Choix de la version du simulateur [11]
↓
Choix de la simulation
↓
Résultats graphiques

Figure 23 — Workflow de *ViZiR 1.0*

Cette première version, déployée sur une machine virtuelle du CNEPE, est plus proche d'un Proof Of Concept (POC) que d'une version stable respectant les normes plus contraignantes d'une application en production.

Elle a néanmoins rencontré un certain succès et s'est illustrée dans le cadre d'études comme **l'impact d'une modification sur la régulation d'un sytème fluide de *Flamanville 3***. Toutefois, l'import de certains fichiers csv non pris en charge occasionnait des déconnexions difficiles à déboguer pour les utilisateurs, ce qui pouvait les décourager dans l'utilisation de *ViZiR*.

[11] Correspond à la base de données associée au simulateur, qui va permettre de récupérer les libellés, les unités, etc.

6.2.2 | ViZiR 2.0

Un certain nombre d'objectifs ont été fixés pour corriger les problèmes rencontrés avec *ViZiR*, améliorer la stabilité de l'application et lui ajouter des fonctionnalités.

> **Important**
>
> - Le *refactoring* du code source de R vers **python**,
> - Une application dont les fonctions sont documentées, testées et avec versionning adapté,
> - Une amélioration de l'ergonomie et de l'interface utilisateur,
> - La possibilité de sauvegarder des paramètres graphiques pour les recharger plus tard,
> - L'élargissement du *scope* initial de *ViZiR* → les fichiers **csv** importés peuvent provenir de différentes sources hétérogènes de données, à partir du moment où ils possèdent une colonne de pas de temps,
> - Un système de multigraphe,
> - Une fonctionnalité permettant de suivre le trafic sur l'application.

* * *

6.2.2.1 | Refactoring du code source

Le refactoring du code source s'explique par la volonté du groupe FGP de déployer l'application sur la plateforme **Datatools**, une offre de services « clé en main » proposée par la DSI groupe d'EDF, qui permet de développer tout type d'application, sans avoir à se soucier de la gestion des éléments d'infrastructure comme les démarches d'authentification, de gestion des filesystem, du stockage, etc. Cependant, **Datatools** ne permet pas le déploiement d'applications dont le code source n'est pas du **python**. (R n'étant pas un langage qualifié par la DSI)

La plateforme utilise *Kubernetes*, un logiciel d'orchestration de conteneurs open source qui permet de gérer les applications conteneurisées.

> **Note**
>
> Un conteneur est une manière de virtualiser un Operating System (OS) pour exécuter une application. On peut facilement y packager le code source d'une application, ses configurations et ses dépendances. Enfin, on peut construire et lancer une image s'exécutant sur n'importe quelle infrastructure. Par rapport à une machine virtuelle, les conteneurs sont plus légers et plus rapides.

Kubernetes place ensuite ces conteneurs dans des pods et les exécute sur des nœuds *(worker nodes)*. Un cluster contient au minimum un nœud principal exécutant un pod de conteneurs et un plan de contrôle *(control plane)* qui gère le cluster.

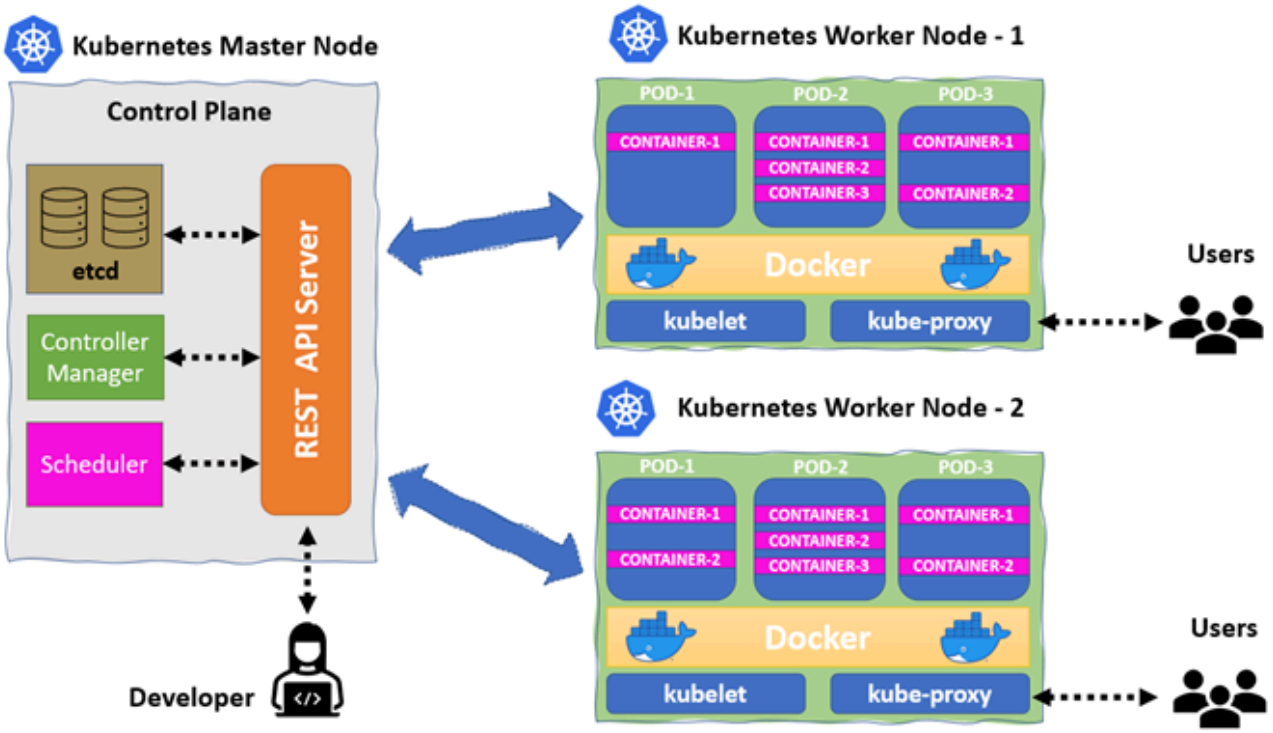

Figure 24 — Exemple de cluster *Kubernetes*

6.2.2.2 | Industrialisation de la solution

Plusieurs conditions sont nécessaires pour faire tourner une application robuste en cycle de production.

Par exemple, le POC de *ViZiR* souffrait entre autres d'une utilisation de **Git** inadaptée. En effet, chaque modification du code source était réalisée sur une seule branche avec un message d'un mot sans explication supplémentaire. On aboutit à l'historique suivant :

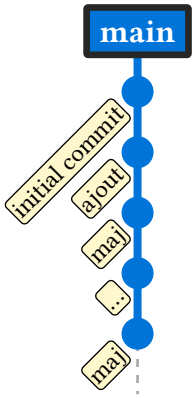

Figure 25 — Repository mono-branche de *ViZiR* 1.0

Sans aucun test unitaire [12] ni aucun test d'intégration [13] , cette pratique est risquée, car, à chaque modification, il n'y a aucune garantie que le code fonctionne toujours de manière optimale. Dans le cadre d'un POC, ce n'est pas forcément un problème, mais à mesure que l'application grandit, accumule les fonctionnalités et les utilisateurs, il faut un mode alternatif de gestion des branches.

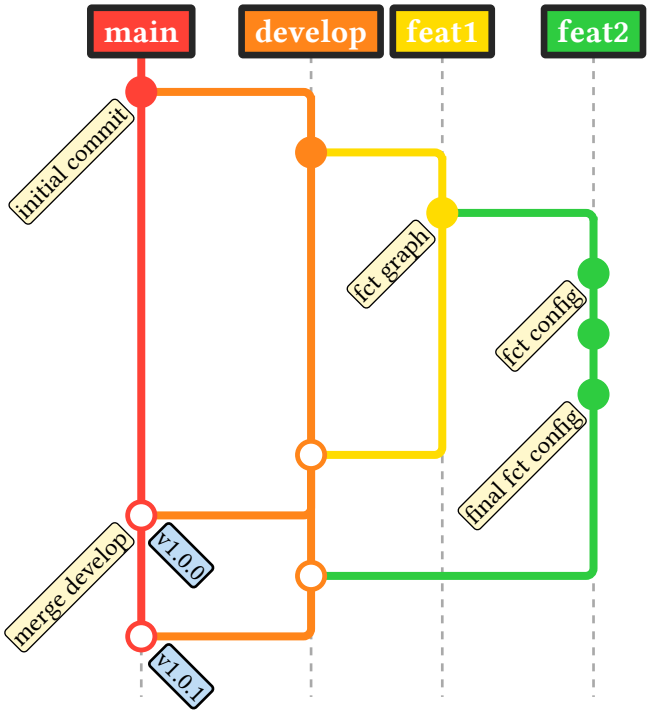

Figure 26 — Extrait du repository de *ViZiR* 2.0

[12] Un **test unitaire** doit tester une caractéristique et une seule. On ne définit pas un scénario de test complexe dans un test unitaire.

[13] Un **test d'intégration** vérifie que les différents modules d'un logiciel fonctionnent correctement ensemble.

C'est exactement ce que va permettre `gitflow`. Au lieu d'une seule branche **main**, ce workflow utilise deux branches pour sauvegarder l'historique du projet. La branche **main** stocke l'historique officiel des versions, et la branche **develop** sert de branche d'intégration pour les fonctionnalités (**feat1**, **feat2**, **etc.**), comme montré ci-dessus avec la Figure 26.

Le second point concerne le manque de clarté dans la documentation. Certaines fonctions disposaient de commentaires explicatifs tandis que d'autres n'en avaient pas. De plus, aucune convention de nommage n'avait été définie préalablement, donc, certaines fonctions avaient été écrites avec un titre en français/anglais et ne décrivaient pas nécessairement le rôle de celle-ci.

Sur **python**, pour donner des indications claires sur le comportement d'une fonction, on peut utiliser conjointement les annotations de type et les *docstrings*. Par exemple :

sqlite_conn.py

```python
def get_table(db_path: WindowsPath, query: str) -> pl.DataFrame:
    """
    Permet de charger une base de données sqlite
    sous forme de DataFrame Polars.

    - Ignore les potentielles erreurs d'encodage à l'importation de la table
    pour éviter une exception.
    - `infer_schema_length=None` permet de forcer polars à scanner
    toute la table pour éviter de mal inférer le type des données.

    Note : la notation `with` permet d'ouvrir la connexion à la DB puis
    de la refermer après son utilisation, empêchant les problèmes de cache.

    Args:
        db_path (WindowsPath): Le chemin vers la base de données
        query (str): Une reqûete SQL

    Returns:
        pl.DataFrame: Le Dataframe résultant de la requête
    """
    with sqlite3.connect(db_path) as conn:
        conn.text_factory = lambda b: b.decode(errors="ignore")
        table = pl.read_database(query, conn, infer_schema_length=None)
    return table
```

→ La fonction `get_table` est entièrement documentée (partie grise avant le code) et les annotations de type sont précisées à droite de chaque argument et derrière la flèche pour le type de retour de la fonction. En lisant uniquement la signature [14] de la fonction, on peut facilement comprendre que celle-ci récupère une table d'une base de données à partir d'un chemin et d'une requête SQL, puis qu'elle renvoie un dataframe. Si on souhaite plus de détail, il suffit de lire la *docstring*.

Pour améliorer la qualité et la lisibilité du code, j'ai aussi utilisé un *linter* (`ruff`), dont le rôle va être d'analyser le code à la recherche d'erreurs, de vulnérabilités et de problèmes stylistiques.

---

[14] La signature d'une fonction correspond à l'ensemble de ses paramètres (noms, positions, valeurs par défaut et annotations), ainsi que l'annotation de retour.

6.2.2.3 | Amélioration de l'ergonomie et de l'interface utilisateur

ViZiR 2.0 est construit avec le package `streamlit` sur **python**.

L'architecture de `streamlit` permet de concevoir des applications de la même manière qu'en écrivant des scripts **python**. Pour permettre cela, `streamlit` dispose d'un fonctionnement particulier : il n'y a pas de concept de *callback* — chaque fois qu'un utilisateur interagit avec un widget, tout le script est ré-éxécuté.

> **Note**
>
> Il est cependant possible de limiter le chargement d'opérations chronophages en utilisant le cache du navigateur grâce au décorateur `@st.cache`, c'est par exemple le cas avec la fonction `get_table` de la page précédente.

Figure 27 — Onglet Accueil de *ViZiR 2.0*

L'interface a été entièrement repensée afin de garantir une meilleure expérience utilisateur et un *workflow* plus intuitif. L'objectif est de garantir une navigation simple avec un minimum d'assistance. Pour cela :

- Toute la partie liée à l'import de fichiers et la configuration est réunie à un seul endroit, dans la barre latérale,
- Un guide de démarrage rapide et un manuel utilisateur sont mis à disposition sur la première page,
- Un fichier Cascading Style Sheets (CSS) a été utilisé pour customiser plusieurs composants de l'interface,
- Tout le système de personnalisation du graphique est regroupé dans un expandeur pour ne pas surcharger l'interface.

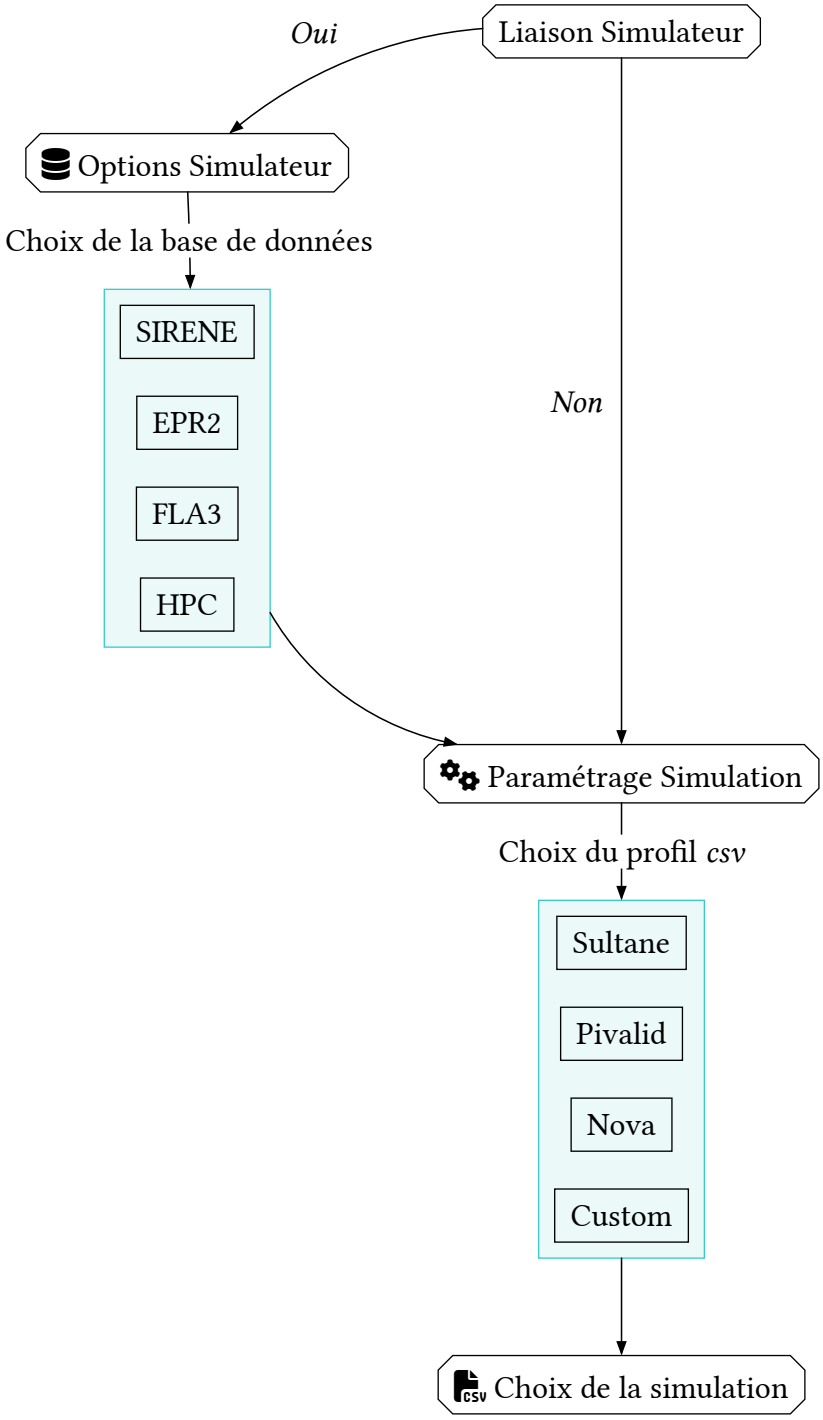

Figure 28 — *Workflow* d'import de *ViZiR 2.0*

Le *workflow* d'import de *ViZiR 2.0* est expliqué dans la figure ci-dessus.

---

1. L'utilisateur peut choisir s'il souhaite ou non un lien avec les bases de données des simulateurs.
2. Si oui, il peut alors sélectionner le type de simulateur (HPC, FLA3, etc.) et la version de la base de données **sqlite**.
3. Si non, il peut directement passer à l'étape de configuration du profil *csv*.
4. Enfin, la dernière étape concerne l'import du fichier *csv* de simulation.

Note

Ces étapes n'ont pas forcément besoin d'être réalisées dans cet ordre précis. L'utilisateur peut tout à fait décider d'importer son fichier *csv* avant de le configurer, mais risque de se retrouver face au message suivant :

Attention, ViZiR n'a pas réussi à lire le fichier *csv* actuellement importé.

Etes-vous sûr d'avoir sélectionné le bon profil de fichier *csv* dans **Paramétrage du fichier csv** ?

Si le profil n'existe pas, il faut étudier plus en détail le fichier *csv* et choisir l'option **Custom** pour configurer manuellement le séparateur, les lignes à retirer et le pas de temps.

De plus, **ViZiR** accepte uniquement les *csv* structurés de la manière suivante :

⇒ Le pas de temps de la première colonne est variable : date, secondes, millisecondes, etc.

Colonne de Temps	Capteur_1	Capteur_2	Capteur_N
1000	0.1	25.65	8900.2
...	...	...	...
10000	0.2	15.24	12589.36

Une fois le fichier *csv* importé, il est possible d'obtenir un compteur des capteurs analogiques et Tout Ou Rien (TOR), et d'effectuer des transformations d'unités pour les capteurs analogiques [15] , ce qui peut s'avérer utile avant la visualisation avec l'onglet **Grapheur**.

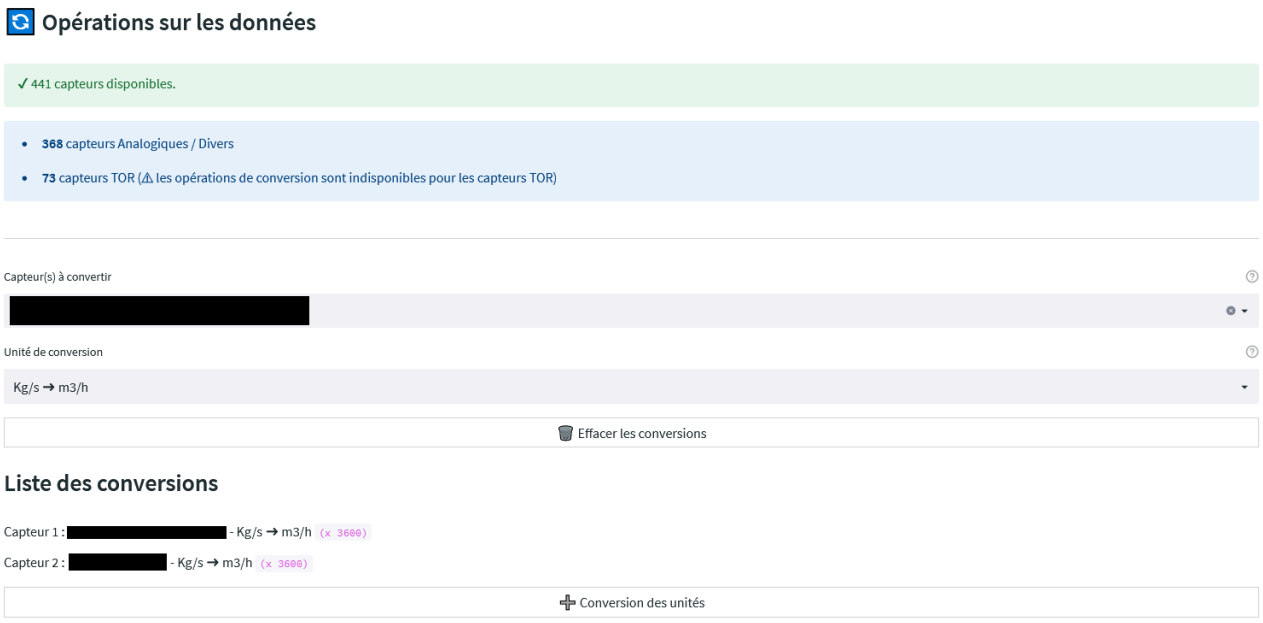

Figure 29 — Onglet Données de *ViZiR 2.0*

[15] Une conversion d'unité pour les capteurs TOR n'aurait pas de sens, puisque les valeurs de ces capteurs sont $\{0, 1\}$.

Une fois que le fichier csv est chargé, il est possible de sélectionner les capteurs analogiques et TOR à afficher dans une liste déroulante.

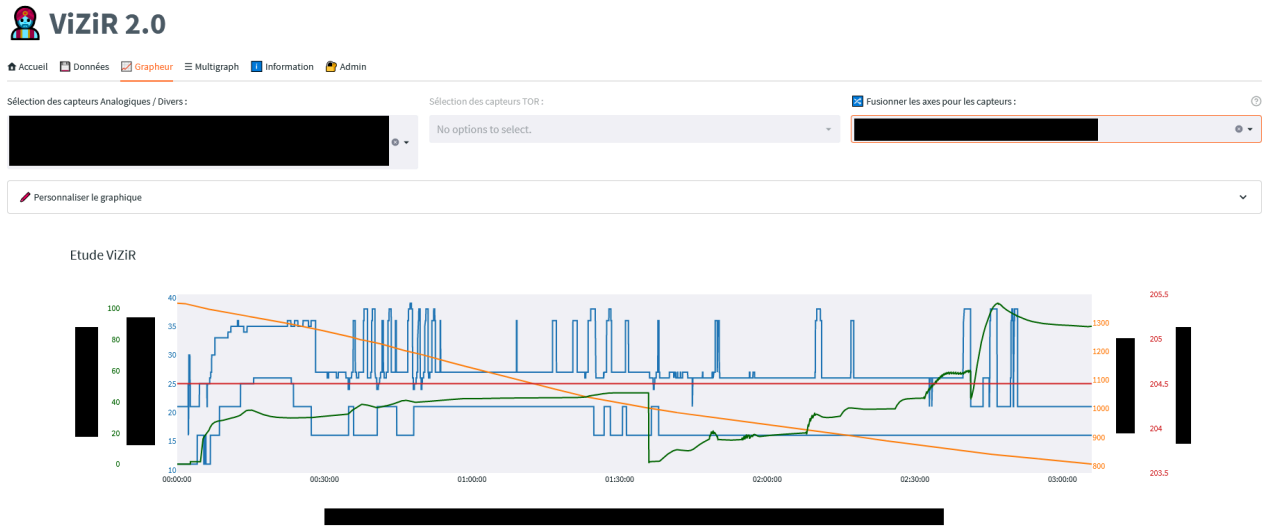

Figure 30 — Onglet Grapheur de *ViZiR 2.0*

Sous le graphique, une légende est disponible afin de notifier, pour chaque capteur, son axe Y_n, sa désignation, son unité et des statistiques descriptives simples.

→ Dans la partie de personnalisation du graphique, l'utilisateur dispose de nombreuses fonctionnalités permettant notamment de changer le titre, de modifier les minimums et maximums, faire des sous-graphes, ajouter des seuils, des annotations, des zones et enfin changer la couleur des courbes.

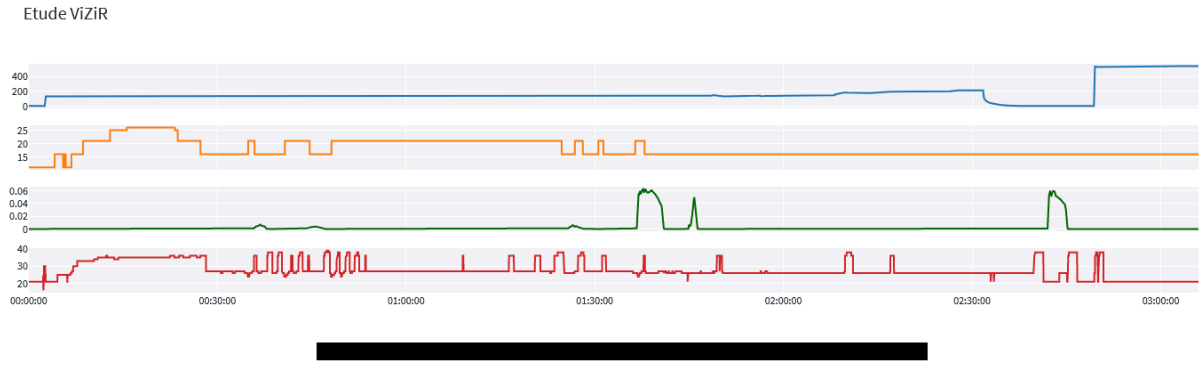

Figure 31 — Subdivision de la fenêtre graphique dans *ViZiR 2.0*

* * *

6.2.2.4 | Gestion des paramètres de session

Une étude réalisée en utilisant le POC de *ViZiR* est par définition éphémère, c'est à dire que chaque déconnexion du serveur ou fermeture de l'onglet du navigateur entraîne une perte de la configuration. L'utilisateur doit ensuite tout re-paramétrer à la main s'il souhaite faire une étude similaire. *ViZiR 2.0* résout ce problème en introduisant une gestion persistante des paramètres, permettant de sauvegarder et de restaurer les configurations automatiquement grâce à un fichier *json*. Cette méthode permet non seulement un gain de temps pour les utilisateurs, mais permet aussi de garantir la reproductibilité des études.

> **Choix d'implémentation**
>
> Dans le contexte du développement web, une session désigne la période pendant laquelle un utilisateur spécifique interagit avec l'application. Les données que l'application enregistre à propos de cette interaction utilisateur sont appelées des **Session States**. `streamlit` propose une puissante Application Programming Interface (API) permettant d'initialiser et de manipuler ces **Session States**.
>
> En utilisant cette API, il est donc possible de créer un système de stockage persistant de données d'interaction utilisateur même lorsque l'application est ré-éxécutée. Si l'idée initiale paraît simple, la mettre en oeuvre de manière optimale pour sauvegarder tous les paramètres clés a été un challenge technique important.

Deux modules de l'application se chargent de la gestion des paramètres de session :

- Le module `app_states.py`,
- Le module `save_session.py`.

Le premier module se charge d'initialiser tous les **Session States** au démarrage de l'application.

app_states.py

```python
def initialize_states() → None:
    """
    Fonction initialisant tous les session_states par onglet

    L'interactivité d'une application Streamlit est liée au fait que chaque
    action utilisateur ré-éxécute entièrement le script de l'application.
    Cependant, ce comportement n'est pas toujours souhaitable et peut
    entraîner de nombreuses difficultés dans le développement.

    ⇒ Pour pallier à ce problème,`st.session_state` permet de stocker et
    partager des variables entre chaque ré-éxécution du script.

    En pratique, cette fonction permet donc d'initialiser la configuration
    de tous les input widgets.
    """

    _states_sidebar_import()
    _states_conversion()
    _states_graph()
    _states_admin()
```

Pour rendre l'architecture plus facile à appréhender, l'initialisation des **Session States** dans la fonction `initialize_states` est déportée dans des sous-fonctions [16] par composant de l'interface utilisateur.

Par exemple, la fonction ci-dessous va permettre d'initialiser des valeurs par défaut pour les widgets de la barre latérale de l'application.

app_states.py

```python
def _states_sidebar_import() -> None:
    """
    Les session_states de la sidebar.

    - simu_linking: Lien avec le simulateur (Oui ou Non)
    - simu_type : Type de simulateur (EPR2, FLA3, etc.) par défaut EPR2
    - csv_profile : Profil CSV (Pivalid, Sultane, etc.) par défaut Pivalid
    """
    if "simu_linking" not in st.session_state:
        st.session_state["simu_linking"] = "Oui"

    if "simu_type" not in st.session_state:
        st.session_state["simu_type"] = "EPR2"

    if "csv_profile" not in st.session_state:
        st.session_state["csv_profile"] = "Pivalid"
```

Ensuite, le second module se charge de créer le fichier *json* de configuration avec la fonction `create_json_from_session` et de le recharger ensuite avec `load_session_parameters`.

save_session.py

```python
def load_session_parameters(dict_params: dict) -> None:
    """
    Charge les paramètres d'une session ViZiR à partir d'un fichier json
    préalablement transformé en dictionnaire (mapping).

    Args:
        dict_params (dict): Le dictionnaire des paramètres de configuration
        de l'étude ViZiR
    """
    for param_session, value_param_session in dict_params.items():
        st.session_state[param_session] = value_param_session
```

En fait, la fonction remplace tous les paramètres des **Session States** avec ceux qui se trouvent dans le fichier de configuration.

[16] L'*underscore* « _ » devant le nom des sous-fonctions est une convention qui indique que cette fonction est destinée à un usage interne au sein du module dans laquelle elle est définie.

Concrètement, l'utilisateur n'a besoin que de deux étapes pour pouvoir visualiser son étude :

1. Importer le fichier de paramètres *json*,
2. Importer le fichier *csv* associé aux paramètres (sans avoir à tenir compte de la barre latérale).

Dans le cas d'une étude customisée en profondeur, cela donne, en quelques secondes :

Figure 32 — Etude rechargée grâce à un fichier de paramètres *json*

Cette fonctionnalité est clairement l'atout majeur de *ViZiR 2.0*.

6.2.2.5 | Elargissement du scope initial de l'application

Initialement, seul un type de fichier *csv* standardisé était disponible à l'importation dans la première version de *ViZiR* — voir Section 6.2.1. Cependant, d'autres entités d'EDF utilisent elles aussi des fichiers *csv* issus de session sur simulateur, mais avec des paramètres différents. C'est par exemple le cas de la Direction Technique (DT) : leurs fichiers *csv* ne disposent pas du même séparateur ni du même pas de temps.

Pour répondre à cette problématique de non-standardisation, j'ai développé le module `imports.py`.

Ce module est dédié au pré-traitement des fichiers *csv* à l'importation et dispose de fonctions permettant de gérer tous les fichiers *csv* possibles, même lorsqu'ils sont faiblement standardisés, à partir du moment où ils disposent d'une colonne de pas de temps.

Les deux paramètres réglables disponibles sont :

- **separator** → Le séparateur du fichier *csv*,
- **timestamp** → Le format de la colonne de pas de temps (millisecondes, secondes, date).

Pour simplifier encore plus l'utilisation de l'application, ces paramètres sont regroupés sous un « profil *csv* » dans un fichier **YAML**. Ce fichier peut comporter un nombre illimité de profils.

```yaml
Sultane:
  separator: ","
  timestamp: "Secondes"
```

⇐ A gauche, le profil *csv* crée spécifiquement pour la DT.

6.2.2.6 | Système de multigraphe

Certains utilisateurs ont exprimé le besoin de comparer des études ou de visualiser différentes séries de capteurs côte à côte. *ViZiR 2.0* intègre cette fonctionnalité : il est possible de recharger un nombre infini d'études.

> **Note**
>
> Il est à noter que cet onglet ne se substitue pas au **Grapheur** et n'a pas vocation à personnaliser ou re-paramétrer l'ensemble des figures chargées.

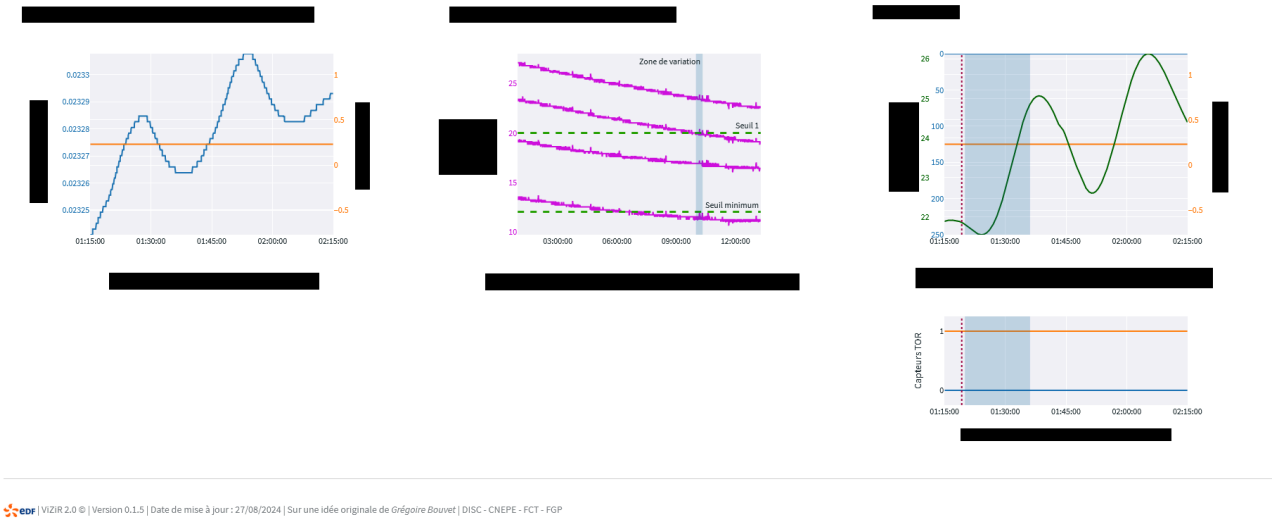

Figure 33 — Onglet Multigraph de *ViZiR 2.0*

Ce système de multigraphe n'est pas forcément le plus optimal d'un point de vue utilisateur, mais tient compte de deux limites identifiées :

- Le temps d'éxécution des fonctions de création d'une figure sur `plotly`, qui augmente linéairement avec la volumétrie des données,
- La puissance limitée de la machine virtuelle mise à disposition sur **Datatools**.

* * *

6.2.2.7 | Monitoring de l'application

ViZiR 2.0 dispose d'un onglet d'administration restreint qui requiert un nom d'utilisateur et un mot de passe pour accéder au panneau suivant :

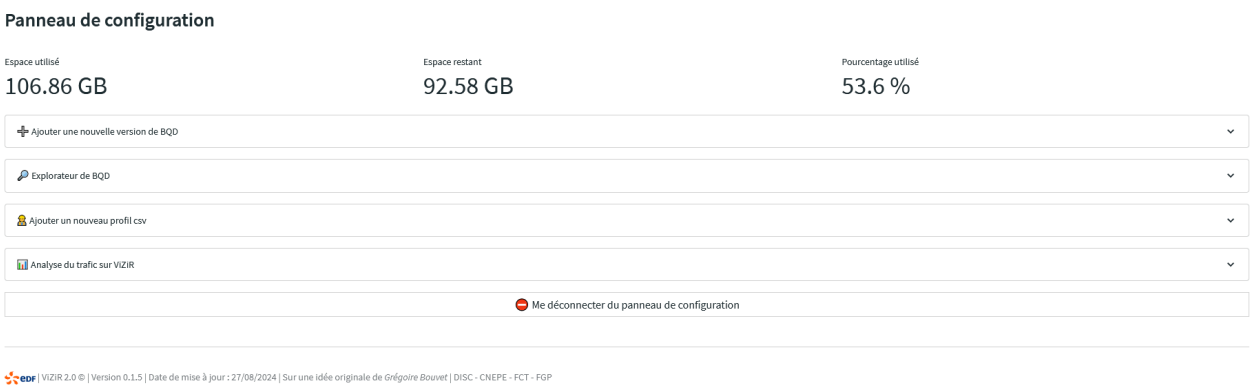

Figure 34 — Onglet Administrateur de *ViZiR 2.0*

Concentrons-nous ici sur la fonctionnalité d'analyse du trafic.

Le trafic est suivi grâce à une fonction encapsulée dans le système d'import : celle-ci gère l'incrémentation des lignes du fichier **monitoring.parquet** [17] à chaque fois qu'un nouveau fichier *csv* est importé. Enfin, pas tout à fait... En réalité, un fichier peut rester dans le cache de l'application pendant 30 minutes grâce au décorateur `@st.experimental_memo(ttl=1800)`, et ne va pas générer de nouvelle ligne — cette précision est importante, car, de part la nature de `streamlit`, sans ce décorateur, chaque interaction utilisateur signifierait l'écriture d'une nouvelle ligne dans le fichier de monitoring, ce qui entraînerait un nombre de lignes colossal par fichier importé, et fausserait le suivi.

Côté interface utilisateur, il suffit de sélectionner un jour dans le calendrier pour obtenir un graphique des imports par heure, et un tableau qu'il est possible de télécharger.

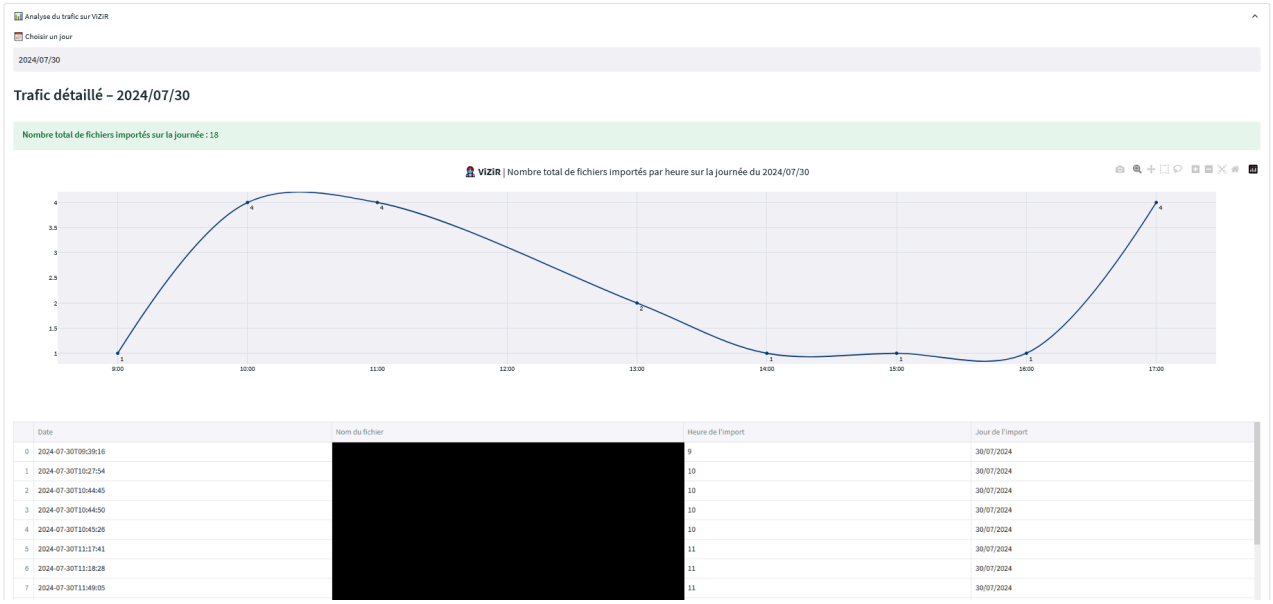

Figure 35 — Analyse du trafic sur *ViZiR 2.0*

[17] Le format parquet dispose d'une compression efficace, ce qui réduit l'espace disque nécessaire et accélère les temps d'écriture et de lecture.

> **Important**
>
> En prenant un certain recul vis-à-vis de mes développements, la fonctionnalité de monitoring est en réalité très importante stratégiquement pour le groupe FGP, puisqu'elle permet notamment d'identifier des pics d'activité sur l'application, de mesurer la charge de travail pour le conteneur, et surtout de justifier des prises de décision et défendre l'intérêt de *ViZiR*. De plus, le chiffrage du gain engendré par l'outil sera facilité.

7 | Conclusion

Ce stage de fin d'études chez EDF au CNEPE m'a offert une expérience professionnelle très enrichissante à plusieurs égards. J'ai eu l'opportunité de travailler sur deux sujets différents mêlant des compétences techniques variées, notamment en **python**, **R**, **Git**, et bien d'autres outils, tout en m'imposant un cadre rigoureux en matière d'ingénierie logicielle. Ces six mois m'ont permis non seulement de monter en compétence dans le domaine de la Data, mais aussi de découvrir le domaine complexe du Nucléaire.

Au-delà des compétences purement techniques, j'ai amélioré mes compétences personnelles telles que la gestion et l'organisation de mon temps de travail, ce qui m'a été indispensable pour mener à bien mes deux projets tout en respectant les délais. En échangeant régulièrement avec des collègues d'autres unités dans le cadre de ces missions, j'ai également renforcé mes capacités de communication. Les présentations lors d'ateliers dédiés à la Direction Technique ainsi qu'au « Club Simu » (une communauté d'ingénieurs utilisant les simulateurs) m'ont aussi permis de gagner en crédibilité dans mon travail, via le renforcement de mes facultés de synthèse.

L'application *ViZiR* constitue une base solide susceptible d'accueillir d'autres fonctionnalités dans les prochains mois. Le cœur de l'outil est à la fois robuste et modulaire, ce qui permettra à un futur repreneur de déployer l'application pour d'autres usages opérationnels, tout en garantissant une maintenance facilitée.

Finalement, un des meilleurs indicateurs de la réussite de ce stage est le fait que les applications que j'ai développées sont désormais utilisées et valorisées. En effet, le spectre d'utilisation d'un outil comme *ViZiR* est assez large (nouveau nucléaire, ingénierie du parc en exploitation, etc.). Pour *HPConnect*, il serait tout à fait envisageable qu'EDF transpose la logique au projet EPR2, avec l'avantage non négligeable de disposer de toute la documentation et de la logique métier.

Enfin, je tiens à souligner que ces six mois ont été marqués par une ambiance de travail idéale, qui a largement contribué à l'accomplissement de mes objectifs et à la réussite de mes projets.

* * *

Acronymes

AAR :	**Arrêt Automatique Réacteur**
AP :	**Automation Processor**
API :	**Application Programming Interface**
ASN :	**Autorité de Sûreté Nucléaire**
BK :	**Bâtiment Combustible**
BP :	**Basse Pression**
BR :	**Bâtiment Réacteur**
CEA :	**Commissariat à l'Energie Atomique**
CNEPE :	**Centre National d'Equipements de Production d'Electricité**
CNPE :	**Centre Nucléaire de Production d'Electricité**
CRF :	**Circuit de Refroidissement**
CSS :	**Cascading Style Sheets**
DETU :	**Département Etudes**
DFD :	**Detailed Functional Diagrams**
DIPE :	**Département Ingénierie du parc en exploitation**
DISC :	**Direction Ingénierie et Supply Chain**
DPNN :	**Département Projets Nouveau Nucléaire**
DT :	**Direction Technique**
EDF :	**Electricité De France**
EPIC :	**Etablissement Public à caractère Industriel ou Commercial**
EPR :	**Evolutionary Power Reactor**
ETP :	**Equivalent Temps Plein**
FGP :	**Fonctionnement Général et Performance**
FUM :	**Function Module**
HP :	**Haute Pression**
HSL :	**Hardware Signals List**
I&C :	**Instrumentation et Contrôle-Commande**
IDE :	**Integrated Development Environment**
IHM :	**Interface Homme Machine**
IPE :	**Ingénierie du Parc en Exploitation**
MCS :	**Moyen de Conduite de Secours**

NNB : **Nuclear New Build**

OS : **Operating System**

PAS : **Plant Automation System**

POC : **Proof Of Concept**

REP : **Réacteur à Eau Pressurisée**

SA : **Société Anonyme**

SAS : **Safety Automation System**

SDC : **Salle de Commande**

SDM : **Salle Des Machines**

SID : **Service d'Information et de Documentation**

SSL : **Software Signals List**

TBP : **Turbine Basse Pression**

TOR : **Tout Ou Rien**

UNGG : **Uranium Naturel Graphite Gaz**

Liste des figures

Liste des tables