

INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

DÉPARTEMENT DE GÉNIE ÉLECTRIQUE ET INFORMATIQUE

Projet de Fin d'Etudes

Spécialité : Informatique-Réseaux

Filière : Systèmes Distribués et Big Data

Développement de la Chaîne Mission du Segment Sol de Satellites

Auteur :

JACQUES Matthieu

Entreprise :

Capgemini Technology Services

Référent INSA :

AIME Martin

Responsable du stage :

RAYNAUD Mathieu

Année 2021-2022

Résumé

Le présent rapport relate de mes activités lors de mon contrat de professionnalisation au sein de Capgemini à Toulouse, du 23 Septembre 2021 au 23 Septembre 2022.

Au cours de cette année, j'ai pu travailler sur un gros projet de la chaîne mission du segment sol d'une constellation de satellite d'observation de la Terre. Mon travail consistait principalement à faire du développement en Java, de la gestion d'environnement Unix, et de la compréhension fonctionnelle d'un projet complexe du domaine spatial.

Après une courte présentation du contexte de mon alternance, j'aborderai les notions clés du spatial et du fonctionnement des satellites d'observation de la Terre. Je présenterai le projet en détail et mes activités au sein de la Mission Chain Run. Je parlerai ensuite des objectifs qui m'ont été fixés et de ma progression dans les différents domaines durant le contrat de professionnalisation. Enfin, je ferais un bilan de mon ressenti et une ouverture sur ce que j'envisage pour la suite de mon projet professionnel.

Remerciements

Je tiens premièrement à remercier Mr Mathieu RAYNAUD, tuteur à Capgemini et responsable technique du projet qui m'a encadré tout au long de mon alternance pour sa disponibilité et pour la confiance qu'il a su m'accorder.

Je remercie ensuite Mr Basile PORET et Mme Hélène GOLSE, chefs de projet, pour leur bienveillance et leur encadrement.

Je remercie aussi Mr Bastien MARRAGOU et Mr Paul HEIDMANN, responsables technique sur le projet, pour leur aide et leurs conseils.

Je remercie également mon référent à l'INSA, Mr Martin AIME, pour son accompagnement et sa bienveillance.

Enfin je tiens à exprimer ma gratitude à l'ensemble des collaborateurs Capgemini avec qui j'ai eu le plaisir de travailler tout au long de cette alternance, qui m'ont rapidement intégrés à une grande équipe et qui ont su m'accompagner dans mes débuts en tant que salarié.

Table des matières

1	Introduction	1
1.1	Contexte général	1
1.2	Préface	2
2	Cadre et objectifs du stage	3
2.1	Environnement de travail	3
2.2	Objectifs de la mission	4
2.3	Présentation de l'équipe	4
2.4	Contexte Fonctionnel	6
2.4.1	Le domaine du spatial	6
2.4.2	Le Segment Sol	6
2.4.3	La Mission Chain Run	7
2.4.4	Définition du satellite	7
2.4.5	Les orbites courantes	8
2.4.6	Les déplacements du satellite	9
2.4.7	La boucle complète	10
2.4.8	Mise en situation : Ordre de prise de vue	11
2.4.9	Enjeux	12
2.5	Contexte Technique	13
3	Réalisations	15
3.1	Fonctionnement du projet	15
3.1.1	Méthodologie	15
3.2	Organisation du développement	17
3.2.1	Le Client lourd	18
3.2.2	Le serveur	20
3.2.3	La base de donnée	20
3.2.4	Git	22
3.3	Mes objectifs	23
3.4	Mes réalisations	25
3.4.1	La mise en place	25

3.4.2	Les formations	26
3.4.3	La maintenance	26
3.4.3.1	Méthodologie	26
3.4.3.2	Anomalies corrigées	29
3.4.4	Les évolutions	33
3.4.5	Confronter les exigences	35
3.4.6	La transmission des connaissances	36
4	Conclusions et perspectives	37
Annexes		40
Annexe 1 : Présentation de la société		40
Annexe 2 : Table des figures		41

Glossaire

ADS (acronyme) : Airbus Defense and Space est l'une des trois divisions du groupe Airbus spécialisée dans les avions militaires, les drones, les missiles, les lanceurs spatiaux et satellites artificiels. Elle est créée en 2014 par la fusion de plusieurs entités existantes.

Bug (n.m.) : Défaut de conception ou de réalisation d'un programme informatique, qui se manifeste par des anomalies de fonctionnement de l'ordinateur.

BackEnd : En informatique, le back-end est un terme désignant un étage de sortie d'un logiciel devant produire un résultat, qui n'est pas visible à l'utilisateur.

Cloud (n.m.) : Le Cloud Computing est la fourniture de services informatiques (notamment des serveurs, du stockage, des bases de données, la gestion réseau, des logiciels, des outils d'analyse, l'intelligence artificielle) via Internet.

Docker : Docker est une plateforme permettant de lancer certaines applications dans des conteneurs logiciels.

ESN (acronyme) : Une Entreprise de Service Numérique est une société qui apporte des solutions informatiques, généralement à d'autres entreprises.

FrontEnd : En développement web, la notion de « front end » fait référence à l'ensemble des éléments visibles et accessibles directement sur un site web (voire sur une application web ou une application web mobile).

Framework : En programmation informatique, un framework est un ensemble cohérent de composants logiciels structurels qui sert à créer les fondations ainsi que les grandes lignes de tout ou partie d'un logiciel.

Géostationnaire (adj.) : Se dit d'un satellite géosynchrone qui décrit une orbite équatoriale et circulaire, dans le sens direct.

Géosynchrone (adj.) : Se dit d'un satellite de la Terre dont la période moyenne de révolution est égale à la période de rotation sidérale de la Terre.

GSD (acronyme) : Dans le domaine du Spatial, la Ground sample Distance (en français, résolution au sol) dans une photo numérique est la distance entre les centres de pixels, mesurée sur le terrain.

IDE (acronyme) : En informatique, un Environnement de Développement Intégré est un logiciel de création d'applications, qui rassemble des outils de développement fréquemment utilisés dans une seule interface utilisateur graphique (GUI).

Intersidéral (n.m.) : Décrit un emplacement situé entre un ou plusieurs astres.

Microservice : En informatique, les microservices sont une technique de développement logiciel qui structure une application comme un ensemble de services indépendants ayant ses propres responsabilités.

POO (acronyme) : La **Programmation Orientée Objet** est un modèle de langage de programmation qui s'articule autour d'objets et de données, plutôt que d'actions et de logique.

TAS (acronyme) : Thalès Alenia Space est une co-entreprise franco-italienne du secteur de l'industrie spatiale basée à Toulouse.

XML (acronyme) : L'Extensible Markup Language, langage de balisage extensible en français, est un langage de balisage, généralement reconnaissable par son usage des chevrons "< >".

Chapitre 1

Introduction

1.1 Contexte général

Ce rapport rend compte de mes activités réalisées durant mon année en contrat de professionnalisation chez Capgemini Technology Services. J'ai entrepris cette alternance dans le cadre de ma 5ème année d'études à l'INSA Toulouse, dans l'optique d'avoir un maximum d'expérience possible avant l'obtention de mon diplôme d'ingénieur.

Après mon DUT Réseaux et Télécommunications, j'ai effectué un an d'école d'ingénieur à CPE Lyon avant de m'orienter vers un parcours plus spécifique en programmation à l'INSA Toulouse, où j'ai été formé en informatique pendant trois ans. Au cours de ma dernière année d'études, j'ai suivi les cours de la spécialité SDBD (Systèmes Distribués Big Data) option AP (Analyse Prescriptive). Ces cours m'ont permis d'acquérir des connaissances en développement, en algorithmie de résolution de problèmes, en gestion de projet et en gestion et traitement des données.

Après mon année de 4A à l'INSA en spécialité Informatique et Réseaux (IR), j'ai souhaité réaliser mon projet de fin d'études en tant que développeur logiciel en contrat de professionnalisation plutôt qu'en stage, pour avoir d'avantages de connaissances quant au monde de l'entreprise en tant que salarié. Je me suis donc mis en relation avec la première ESN de France, Capgemini, à l'occasion du forum de l'alternance de l'INSA. Les périodes d'alternance du contrat de professionnalisation m'ont permis de suivre les même cours que les autres élèves en formation continue, ce qui a été très agréable. De plus, j'ai pu passer toute la période de février à septembre chez Capgemini, ce qui d'après moi m'a permis d'être beaucoup plus efficace que si j'avais suivi un rythme d'alternance plus classique, avec des coupures d'entreprise chaque semaine ou chaque mois.

1.2 Préface

Dans ce rapport je tâcherai de vous expliquer dans une première partie le cadre de mon stage, c'est à dire mon environnement de travail à Capgemini, les objectifs de ma mission et l'équipe avec laquelle j'ai pu travailler durant toute cette alternance. J'y aborderai également le sujet de mon stage, notamment le domaine du spatial et la place de ce dernier dans mon travail. J'y décrirai les principaux enjeux techniques et fonctionnels, le rôle du segment sol, le fonctionnement d'un satellite de prise de photo de la Terre, et les moyens de télécommunication qui les lies.

Dans une seconde partie j'aborderai mes principales réalisations ainsi que ce que j'ai pu apprendre durant mon alternance. J'expliquerai les méthodes de travail et d'organisation de l'équipe, ainsi que les différents enjeux internes à Capgemini. Et enfin j'y détaillerai l'aspect technique de mes développements.

Enfin je conclurai sur mes résultats, ce que cette alternance m'a apporté, et sur mes perspectives futures quant à ma carrière professionnelle.

Chapitre 2

Cadre et objectifs du stage

2.1 Environnement de travail

J'ai effectué la totalité de ma période en entreprise au B612, entre Montaudran et Labège, un bâtiment annexe loué par Capgemini pour tous les projets du domaine spatial en partenariat avec Airbus Defense and Space. Le projet étant confidentiel, tout son développement est effectué sur un site protégé avec un réseau local, et donc forcément entièrement en présentiel. Il consiste en la maintenance et le développement d'un gros logiciel de prise d'image satellite. Ce projet fait partie d'un programme regroupant plusieurs autres projets similaires, qui partagent le même espace de travail que nous, et qui sont également en collaboration avec Airbus Defense and Space.



FIGURE 2.1 – Le B612 à Montaudran

2.2 Objectifs de la mission

Mon intérêt personnel principal pour cette alternance était de me donner une première expérience du monde du travail la plus enrichissante possible avant la fin de mon cursus à l'INSA. Ayant déjà effectué deux stages, l'un en laboratoire de recherche puis un autre dans une petite startup, j'avais pour objectif de réaliser mon projet de fin d'étude dans une grosse entreprise pour avoir une vision d'ensemble de mes possibilités de carrière, ainsi que des avantages et inconvénients de chaque type d'entreprise.

Un autre objectif durant cette alternance a été de m'intégrer à une grosse équipe composée de vingt-huit personnes au moment où j'écris ce rapport, et ce malgré mes périodes d'alternance à l'école. J'ai pu interagir avec plus d'une cinquantaine de collaborateurs sur l'entièreté du plateau, en comptant les partenaires d'Airbus notamment.

2.3 Présentation de l'équipe

J'ai donc travaillé en tant qu'Ingénieur Logiciel comme développeur, et ai pu être encadré par Hélène Golse et Basile Poret mes chefs de projets, ainsi que Mathieu Raynaud mon tuteur en entreprise et référent technique sur le projet.

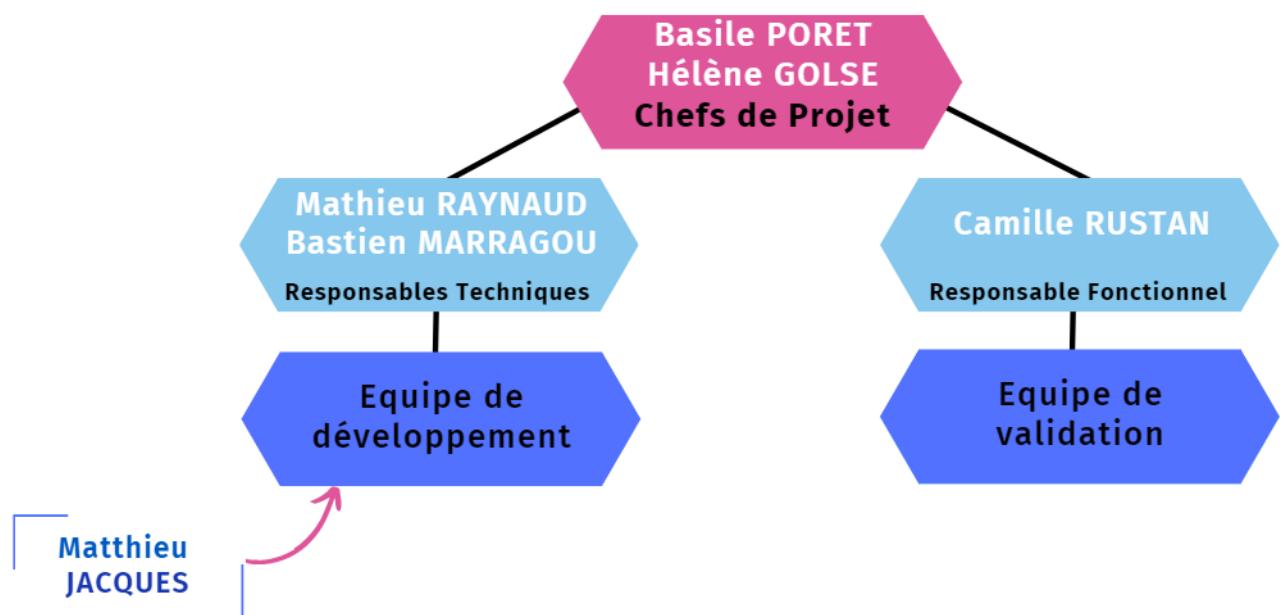


FIGURE 2.2 – Organigramme simplifié

Les deux équipes, valideurs et développeurs, travaillent dans un même espace et échangent couramment quant aux avancées et différentes problématiques.

Le travail d'un validateur est de vérifier que le programme fonctionne conformément aux exigences définies par le client, et ce tout au long de son développement. Il n'interagit pas avec le code et doit donc comprendre l'aspect fonctionnel du projet, réaliser de nombreux tests à chaque nouvelle version, les confronter à la documentation et relever de manière précise les problèmes rencontrés. C'est pourquoi il doit toujours communiquer avec le développeur.

Le développeur lui, utilise un IDE qui lui permet de modifier le code et d'ajouter des points d'arrêt dans l'exécution du programme, par exemple. Son objectif est de réaliser les corrections et améliorations demandées, de les partager avec les autres collaborateurs, et de documenter ses changements pour faciliter le travail des valideurs et des autres développeurs. C'est exclusivement sur ce rôle que je suis intervenu durant mon alternance, bien qu'il soit courant qu'un membre de l'équipe de développement passe dans l'équipe de validation si l'activité de validation devient urgente, par exemple.

Les responsables techniques, en plus d'apporter du support aux autres collaborateurs si nécessaire, peuvent aussi se voir attribuer des tâches qui requièrent une meilleure compréhension du projet, tel que le chiffrage. Le chiffrage consiste en l'estimation du temps de travail requis pour réaliser une évolution du logiciel. Ce chiffrage est ensuite proposé au client, qui le valide ou non.

2.4 Contexte Fonctionnel

2.4.1 Le domaine du spatial

Le spatial est tout ce qui se rapporte à l'espace interplanétaire ou intersidéral. Les stations spatiales, les lanceurs, mais aussi les satellites, sont des exemples de technologies du domaine spatial.

Mon alternance s'est donc orientée vers ce dernier, qui est un domaine apportant une complexité parfois importante aux projets informatiques compte tenu du besoin de comprendre fonctionnellement les méthodes de communication, de calcul de temporalité et de distance qui sont parfois requises pour réaliser certaines corrections ou améliorations du logiciel.

2.4.2 Le Segment Sol

Le Segment Sol est l'ensemble des moyens mis en place pour communiquer avec un satellite depuis le sol terrestre. Sur Terre, le projet s'organise en différentes parties appelées chaînes, chacune gérées par une entité différente, et qui communiquent toutes entre elles.

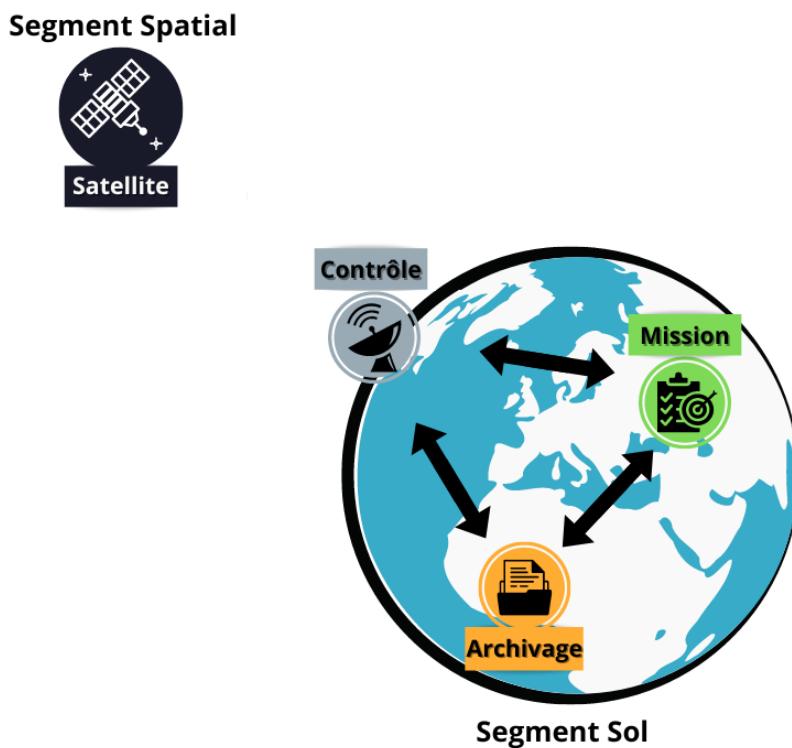


FIGURE 2.3 – Les différentes chaînes du projet

- **Le centre de Contrôle** : Ce centre sert de relais entre le segment spatial et les autres composants du segment sol. Il interagit directement avec le satellite et permet de retranscrire et envoyer les ordres du centre de mission, mais aussi de récupérer les images et données du satellite.
- **Le centre d'Archivage** : Ce centre permet de stocker toutes les informations spatiales récupérées. La **chaîne image** travaille à l'analyse et à la production des images brutes envoyées par le satellite. Elle permet de les rendre exploitable et lisible, et peut leur appliquer des filtres et corrections pour améliorer la qualité de l'image.
- **Le centre de Mission** : Ce centre correspond à l'interface logicielle. Il permet de créer, gérer et planifier les ordres : autrement dit les demandes de prises de photos de la Terre à tel endroit et à tel moment. C'est sur cette **chaîne mission** que j'ai travaillé durant mon alternance.

2.4.3 La Mission Chain Run

Mon alternance s'est déroulée au sein de la chaîne mission, le périmètre "Mission Chain Run". Le mot "run" indique que le projet est en phase de maintenance, c'est à dire qu'il a déjà été livré au client. Sa phase de développement est terminée, mais il continue d'être corrigé et amélioré.

Le projet consiste en la maintenance d'un logiciel de prise de photo de la Terre par satellite, réalisé en collaboration avec Airbus Defence & Space (ADS) et Thalès Alenia Space (TAS).

2.4.4 Définition du satellite

Les satellites existent sous différentes formes et jouent des rôles particuliers. Un satellite peut être naturel, tel que la lune ou une planète ou n'importe quel objet en orbite autour d'un astre céleste. Un satellite artificiel est un objet céleste qui s'oppose à un satellite naturel par son origine humaine. Les satellites artificiels peuvent avoir plusieurs utilités.

Un satellite astronomique a pour mission d'observer l'univers.

Un satellite de télécommunication permet d'utiliser des technologies comme la géolocalisation, la radio ou Internet par exemple.

Dans notre cas, nous utilisons des satellites d'observation de la Terre : leur but étant de photographier la planète.



FIGURE 2.4 – Le satellite Pléiade Neo 3

2.4.5 Les orbites courantes

Les satellites se concentrent principalement sur 3 orbites : LEO, MEO et GEO.

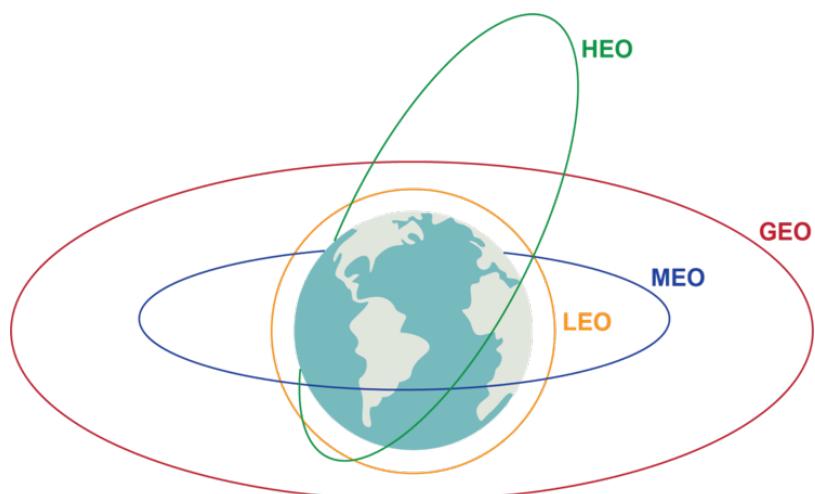


FIGURE 2.5 – Schéma des orbites LEO, MEO, GEO et HEO

— **LEO (Low Earth Orbit)** - 500 - 1'200 km d'altitude

Cette orbite est occupée par des milliers de satellites notamment pour des besoins scientifiques, d'imageries et de télécommunications. C'est sur cette orbite que se trouvent nos satellites de photographie. C'est l'orbite la plus basse pour un satellite artificiel.

— **MEO (Medium Earth Orbit)** - 5'000 - 20'000 km d'altitude

Cette orbite est utilisée pour des applications de navigation notamment le GPS mais peut aussi être pertinent pour d'autres applications nécessitant une faible latence.

— **GEO (Geostationary Earth Orbit)** - 30'000 - 40'000 km

Parmi les centaines de satellites en orbite sur GEO, on trouve des satellites de télécommunications mais aussi des satellites à but météorologique. Cette orbite corres-

pond à la rotation de la Terre. Ces satellites ont la particularité de suivre la rotation de la Terre, ce qui implique qu'ils voient toujours la même face de la planète.

Les autres orbites, comme la **HEO (Highly Elliptical Orbit)**, sont des orbites non-circulaire qui ne sont utilisées par les satellites artificiels que dans de rares cas spécifiques.

2.4.6 Les déplacements du satellite

Notre satellite suit une orbite dite **heliosynchrone**, c'est à dire synchronisée avec le soleil. Il fait toujours face au soleil, indépendamment de la rotation de la Terre. Du point de vue du satellite, il se trouve toujours à la même heure terrestre. Cela crée donc deux phases distinctes : la phase descendante où il "descend" du pôle Nord au pôle Sud, face au soleil et la phase ascendante où il "monte" du pôle Sud au pôle Nord, de l'autre côté de la Terre, et donc dans la nuit.

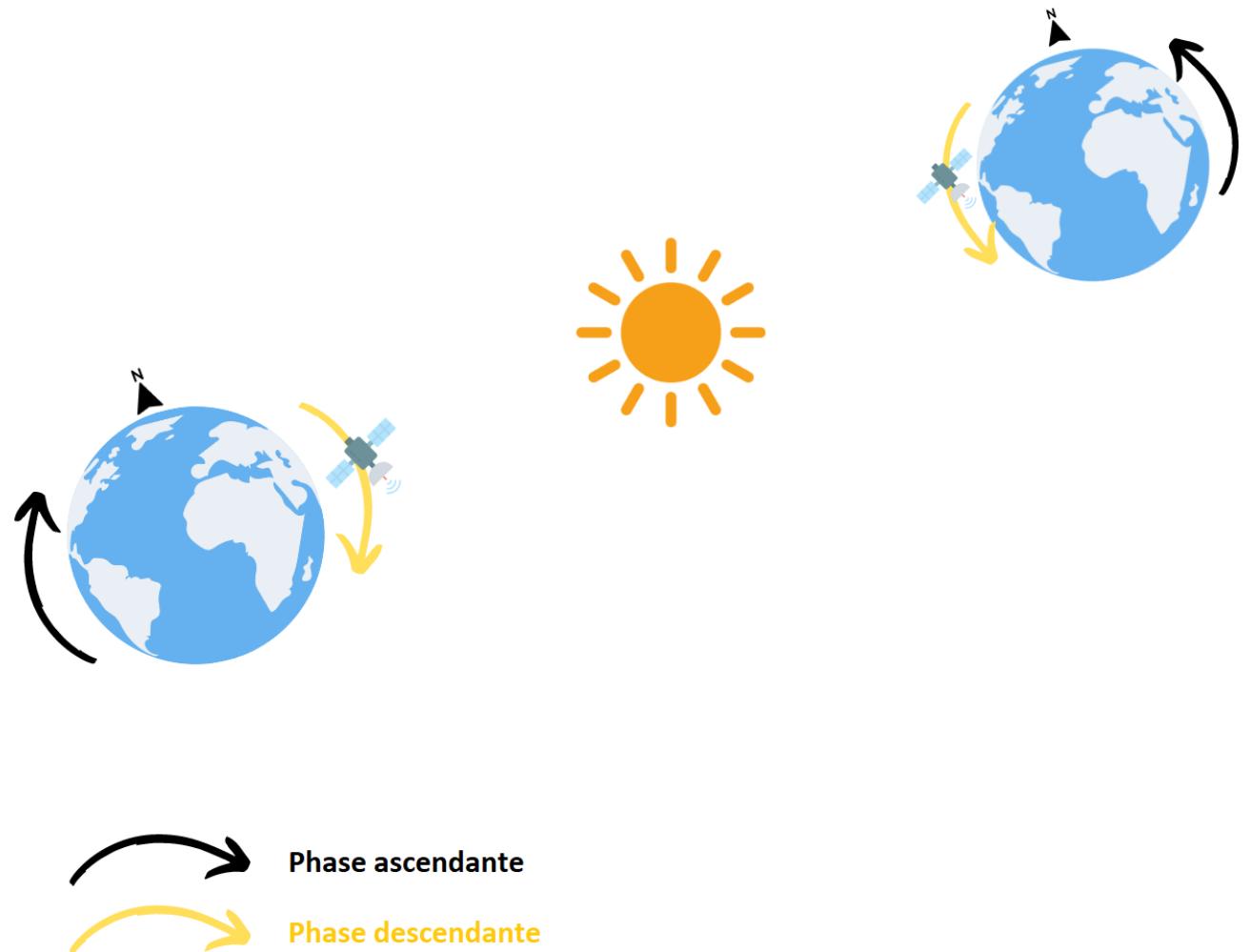


FIGURE 2.6 – Schéma de l'orbite du satellite d'observation

Lorsque le satellite est en phase descendante, la lumière du soleil lui permet de prendre des photos éclairées de la Terre. Quand il est en phase ascendante, il peut prendre des photos mais étant donné l'utilité moindre d'une photographie sans lumière, cette phase est surtout utilisée pour calibrer les différents capteurs, l'inclinaison, et communiquer avec la Terre si le satellite passe proche d'une station.

Le satellite effectue seize tours de la Terre par jour, ce qui lui permet de couvrir une bonne partie de la surface de la planète en une journée. Cependant, étant donné qu'il effectue la moitié de ses déplacements en phase ascendante de nuit, il faut parfois attendre deux voir trois jours avant d'avoir une visibilité sur la cible. L'endroit où l'utilisateur veut prendre une photo est appelé "AoI" (Area of Interest) ou zone d'intérêt.

Le satellite peut également se tourner sur lui-même. Il utilise des propulseurs pour s'orienter. Cela sert à se placer dans le bon angle pour prendre la photo, mais aussi à s'orienter face au soleil quand il a besoin de recharger ses batteries, ou au contraire à protéger ses composants du soleil.

2.4.7 La boucle complète

La boucle complète est une appellation qui désigne un cycle d'utilisation du logiciel. Le cycle débute par la création d'un ordre par l'utilisateur, et se termine par la réception de la photo prise par le satellite.

Le satellite reçoit la demande de prises de photo, qu'on appelle "Programming Order". Ces ordres sont créés par l'utilisateur, et contiennent les différents paramètres souhaités pour la prise de vue : la zone à photographier, le mode de prise de photo, le nombre de photos, la forme de la zone, et bien d'autres.

Les images reçues sont brutes et donc ne sont pas lisibles. Elles doivent d'abord être envoyées à la chaîne image qui se charge de leur appliquer des transformations de différents niveaux. Un premier traitement d'image, appelé Perfect Sensor, est appliqué par défaut pour rendre la photographie exploitable. Ensuite, selon la demande de production faite par l'utilisateur, appelée "Production Order", d'autres filtres peuvent être appliqués, tel que l'orthorectification qui corrige l'angle de prise de vue (un satellite n'est pas forcément pile au dessus de sa cible quand il prend une photo).

Enfin, par la "Delivery Order", c'est à dire la demande de livraison, la chaîne image enverra le résultat à l'utilisateur. A noter qu'il n'est pas obligatoire de faire une demande

de programmation pour avoir une livraison, si on souhaite se faire livrer une image déjà prise par le satellite mais que l'on souhaite lui appliquer un nouveau filtre, par exemple.

2.4.8 Mise en situation : Ordre de prise de vue

Prenons le cas d'un ordre de programmation que nous pourrions envoyer au satellite, pour prendre une photo de Toulouse. Si un seul satellite nous est proposé, nous devrons donc attendre qu'il ait la visibilité sur la ville.

La visibilité de l'AoI est établie quand le satellite passe au dessus de cette dernière pendant une phase descendante, et avec un angle relativement faible par rapport à elle. Plus le satellite est directement au dessus de la zone, avec un angle d'incidence faible, plus la qualité de l'image est élevée.

Mais d'autres paramètres contraignent la prise de notre image. Premièrement, l'envoi de notre ordre devra attendre que le satellite passe en visibilité du centre de contrôle pour lui envoyer l'ordre de prise de vue. De plus, le satellite a déjà un autre ordre de programmation pour une AoI proche de la notre, nous sommes donc en concurrence avec cette autre demande. Ici, cette dernière est classée comme urgente contrairement à la notre. Dans ce cas, on calcule si le satellite est capable de prendre l'image urgente, puis de faire sa manœuvre pour avoir notre zone pour cible, et ensuite de prendre notre photo. Si ce n'est pas possible, alors la prise de notre image sera encore repoussée à la prochaine visibilité, comme dans ce cas hypothétique. Nous sommes alors informés, avant même d'envoyer l'ordre, qu'il faudra attendre encore deux jours supplémentaires pour que la photo soit prise. Le délais total est alors de 5 jours. Nous validons la date proposée par le logiciel, et envoyons l'ordre à la station, qui l'enverra au satellite.

Cinq jours plus tard, le satellite est enfin libre de faire la photographie que nous lui avions demandée. Mais il reste encore à savoir si la photo est exploitable. Le satellite étant à une altitude bien supérieure à celle des nuages, il est possible que la zone soit entièrement cachée, ce qu'on appelle le "CCN" ou couverture nuageuse. Le satellite aura alors pris la photo, et l'aura envoyé à la prochaine station de réception en visibilité. C'est la chaîne image, responsable du traitement de la photo, qui fera la vérification et informera que la couverture nuageuse est trop importante. Dans ce cas, le logiciel refera automatiquement un ordre de programmation. Nous en sommes également informés sur l'interface.

Il aura fallu attendre 7 jours pour recevoir notre photo.



FIGURE 2.7 – Photographie de Toulouse par un satellite Pléiades Neo (Airbus DS)

2.4.9 Enjeux

Il ne m'est pas possible de décrire les enjeux du client final étant donné le caractère confidentiel du projet.

Plus globalement, les enjeux du lancement d'un satellite de prise d'images sont multiples. Le type de satellite sur lequel nous travaillons est prévu pour être maintenu pendant une durée supérieure à 10 ans dans l'espace. C'est en fait parmi les longévités les plus longues pour les satellites en orbite basse, qui sont en moyenne maintenus pendant 5 ans maximum.

Aussi, de nombreux acteurs doivent travailler main dans la main pour parvenir au résultat final, en établissant une norme des appellations, des formats et des moyens de télécommunication par exemple. De nombreuses informations sur le fonctionnement global du système doivent être comprises par toutes les chaînes de production, ce qui n'est pas forcément habituel. Les gros projets industriels, qu'ils soient dans la manufacture ou l'informatique, ont rarement besoin d'une telle transversalité de l'information au travers

de tous les différents acteurs du projet.

Par exemple dans notre cas, les personnes en charge de développer l'IHM, malgré qu'elles ne soient jamais amenées à interagir directement avec le satellite, doivent pouvoir connaître et comprendre sa manière de se déplacer autour de la terre, ses positions précises en temps réel, et les moments où il sera capable de communiquer avec la terre. De même, le logiciel de prise de photo ne peut pas être utilisé par le client final sans formation. De nombreux paramètres et contraintes qui sont propres au domaine du spatial doivent être compris pour utiliser le logiciel de manière optimale.

2.5 Contexte Technique

Pendant ma recherche de projet de fin d'études, j'ai souhaité pouvoir gagner en compétences dans un langage de programmation orienté objet. Je cherchais aussi à éviter la programmation web, et ne pas me limiter au frontend, puisque j'avais déjà passé la majorité de mon stage de 4ème année sur ce type de sujets.

Capgemini m'a donc proposé un projet entièrement en Java, sur lequel j'ai développé le frontend et le backend.

Le logiciel fonctionne avec un framework propriétaire pour l'IHM, et un serveur JBoss pour la partie backend . La base de donnée est en PostGreSQL, avec laquelle nous communiquons grâce au framework Hibernate. Plusieurs outils et microservices sont dockersisés (tournent sur un docker) qui interagissent avec le serveur JBoss. Parmi ces outils, on peut noter le serveur OpenLDAP qui permet de gérer les permissions de l'utilisateur. Nous développons également une unité permettant de calculer et de gérer le temps pendant lequel un satellite est occupé.

Pour développer, étant donné que nous ne pouvons pas faire de tests réels en demandant des prises de vues au satellite, nous simulons toutes ces communications sur des outils développés soit en interne, soit par ADS.

Chapitre 3

Réalisations

Ce chapitre détaille l'ensemble de mes réalisations ainsi que ce que j'ai pu apprendre durant mon alternance. Il fait mention des méthodes de travail et de l'organisation de l'équipe, ainsi que des différents enjeux internes à Capgemini. J'y décrirai l'aspect technique de mes développements, mais aussi mes prises de décisions et les échanges que j'ai pu avoir avec des collaborateurs et des clients.

3.1 Fonctionnement du projet

3.1.1 Méthodologie

Le projet est réalisé entièrement en présentiel dans un réseau interne sécurisé, sur des machines qui ne sont pas reliées à Internet. Nous utilisons néanmoins des logiciel de communication en ligne sur nos ordinateurs personnels, tel que Microsoft Teams.

De nos jours, une majorité de projets industriels utilisent des méthodes dites Agiles qui facilitent l'organisation des équipes de façon flexible, permettant à une équipe de réagir rapidement aux changements des demandes client et aux imprévus. La méthode Scrum, ou SaFe pour les plus grandes équipes, sont des méthodes courantes et très utilisées au sein de Capgemini ou même au sein du Segment Sol.

Mais notre projet utilise une méthode différente qui n'est pas une méthode agile, appelée le **Cycle en V**. Celle-ci organise le projet en deux phases :

1. PHASE DESCENDANTE - Cette phase débute par la définition du périmètre qui va permettre d'établir l'ensemble des évolutions et corrections d'anomalies à mettre en place durant le cycle. Une fois la version bien définie, les différentes activités vont

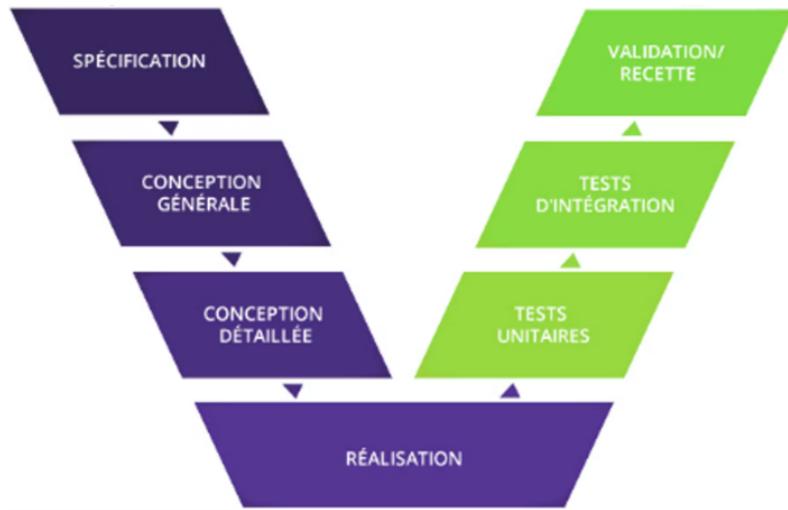


FIGURE 3.1 – Schéma de la méthodologie du Cycle en V

être chiffrées, puis développées et implémentées. Chaque composant est ensuite déployé pour effectuer l’assemblage de la nouvelle version.

2. PHASE ASCENDANTE - Une fois assemblée, c'est l'équipe de validation qui va prendre le relais en rédigeant des tests à jouer pour chaque développement effectué. Les tests spécifiques mais aussi les tests de non-régression sont ensuite joués sur la version. Une documentation est aussi produite. Si les tests passent, on peut alors livrer la version. En cas de persistance d'anomalies ou de régression, on revient à l'étape de réalisation, puis on ré-effectue la phase ascendante.

Cette méthode n'a pas été choisie pour faire un parallèle avec le satellite, mais plutôt pour des raisons de taille d'équipe, d'apparition d'anomalies souvent détectées tardivement, et la difficulté de prévoir de manière précise le temps que prendra une correction. Ces facteurs rendent l'application stricte des méthodes agiles peu envisageable. Il arrive souvent qu'un projet décide de n'appliquer qu'un seul cycle en V pendant son développement, puis qu'il passe en méthode agile une fois livré, pour la phase de maintenance. Dans notre cas, nous effectuons un cycle en V toutes les 8 à 10 semaines, et donc une nouvelle version livrée à chaque fois.

En plus de ces étapes, nous incorporons certaines cérémonies agiles empruntées à la méthode Scrum. Le **DSTUM** pour Daily Stand Up Meeting, est une réunion journalière à laquelle tous les membres de l'équipe participent dans le but de mentionner leur mission en cours, leur avancée et les éventuels blocages rencontrés. Elle n'a pas pour but d'entrer dans les détails des réalisations de chacun et doit durer entre 5 à 10 minutes, c'est à dire une trentaine de seconde par personne. J'ai surtout apprécié ce format de cérémonie car il me permettait de rapidement informer toute l'équipe lorsque j'avais besoin d'aide, et

également de savoir quel collaborateur était le plus disponible pour m'aider en fonction de leur charge de travail. La nécessité du DSTUM était parfois remise en question, même si je pense qu'il reste relativement utile tant qu'il n'est pas trop long. Mon expérience antérieure avec les réunions journalières auxquelles j'avais participé pendant mon stage était bien plus négative, car ces réunions étaient alors moins encadrées et pouvaient durer jusqu'à une heure si des collaborateurs entraient dans les détails de leurs développements.

Notre logiciel de gestion de ticket, Redmine, recense tous les membres de l'équipe et les tâches qui leur sont assignées. Ce suivi est affiché durant le DSTUM. Il peut également permettre de connaître le temps passé sur une tâche, le temps restant estimé, les différentes avancées et informations sur chaque ticket. C'est un logiciel de suivi très semblable à Jira, sur lequel nous avons eu des cours en 5ème année de l'INSA. Le client dépose des tickets sur un logiciel de suivi propriétaire d'Airbus DS, puis nous les retransmettons sur Redmine. Les développeurs et valideurs choisissent et s'assignent eux-même leurs tickets. Au début de l'alternance, il m'a fallu quelques semaines avant de pouvoir choisir moi-même mes tickets étant donné que je ne comprenais pas encore les termes techniques qu'ils contenaient, et que donc je n'avais aucune idée de la difficulté de la tâche.

3.2 Organisation du développement

Le logiciel de la chaîne mission est une application qui est basé sur une architecture client-serveur. Ce produit est le résultat de nombreux composants qui contribuent à sa construction à la fois côté client mais aussi serveur.

Une complexité supplémentaire du projet vient du fait que plusieurs projets différents sont construits l'un sur l'autre. Un projet socle, commun à de nombreux logiciels de prise de vue par satellite, fait la base de cette architecture. Un second projet d'intégration de services extérieurs a ensuite été rajouté, pour permettre au logiciel d'ajouter de nouvelles fonctionnalités. Et enfin, un troisième projet spécifique à la mission satellite a été ajouté. Chacun de ses composants dépendent des deux projets précédents, pour un total de plus de 12'000 classes Java, et 35'000 fichiers. Plusieurs dizaines de composants sont divisés en 51 projets et se réfèrent à une partie bien définie de l'application, tels que la visualisation de l'avancement d'un ordre de prise de vue dans la boucle complète, ou le catalogue des images par exemple.

Pour rappel, le client et les serveurs sont tous deux en Java, et communiquent avec une base de donnée PostGreSQL, ainsi que différents dockers. Durant mon alternance, j'ai pu interagir avec chaque partie du projet, et ai directement modifié le client, le serveur

ainsi que la base de donnée.

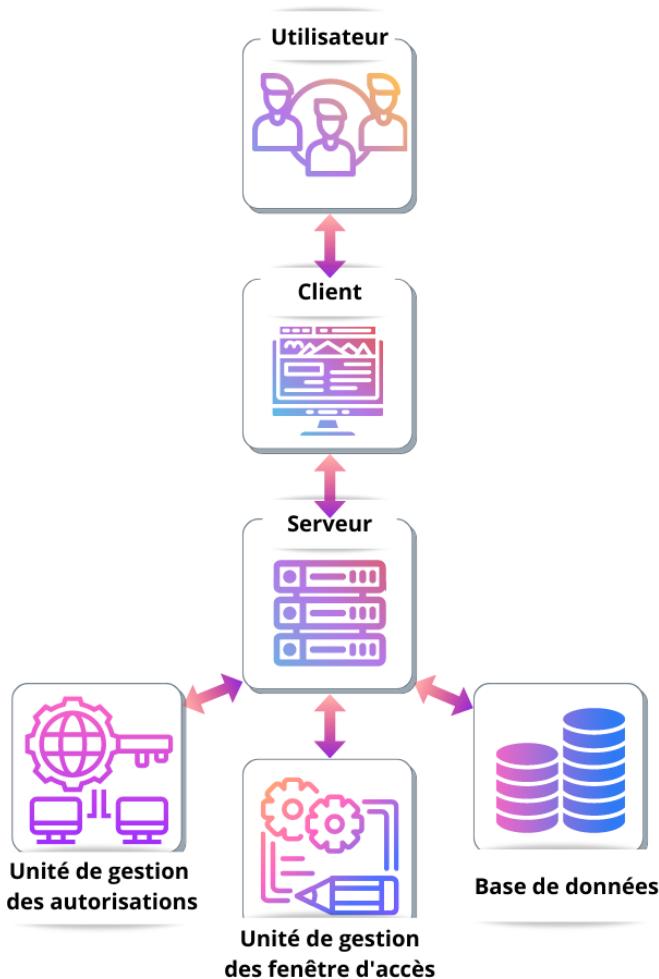


FIGURE 3.2 – Architecture du logiciel

3.2.1 Le Client lourd

Il existe deux types d'architecture client :

1. Le client léger (ou thin client) :

C'est un client dont la majorité des ressources se trouvent sur le réseau. Les clients web en sont un bon exemple. Ils n'ont pas besoin d'être entièrement installés sur la machine de l'utilisateur.

2. Le client lourd (ou fat client) :

C'est un client installé et exécuté sur la machine de l'utilisateur avec la majorité des services dont il a besoin pour fonctionner. C'est ce type de client qui est utilisé

pour notre logiciel, pour des problématiques de sécurité et de frameworks utilisés.

Le client est basé sur un framework propriétaire dont le nom est confidentiel, et s'organise sous la forme de trois couches :

1. Couche Domaine :

Basée sur **EMF** (Eclipse Model Framework) permettant de représenter et manipuler les éléments du modèle de données.

2. Couche Service :

Basée sur le framework **OSGi** qui implémente un modèle de composants dynamique.

3. Couche Graphique :

Basée sur **Eclipse RCP** (Rich Client Platform), un framework qui permet de faciliter la création d'interfaces riches.

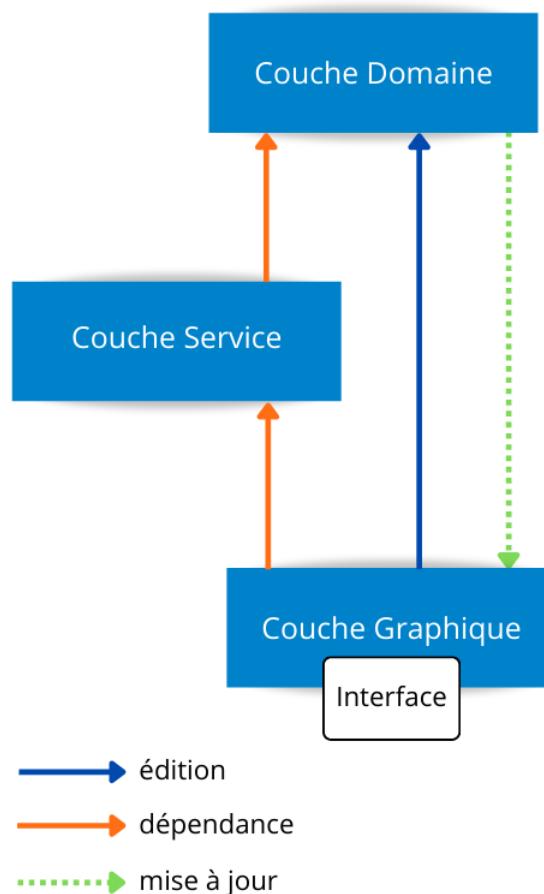


FIGURE 3.3 – Architecture du client

3.2.2 Le serveur

Le serveur est une application Java EE, déployé au format "EAR". Ce format de fichier permet d'empaqueter plusieurs modules en un seul fichier. On exécute ensuite le fichier EAR sur un serveur d'application JBoss. Celui-ci va faire interagir le client et les différents services comme la base de donnée ou les serveurs dockerisés, tel que le serveur OpenLDAP qui permet de gérer les autorisations des utilisateurs du logiciel.

3.2.3 La base de donnée

La base de donnée fonctionne sous PostGreSQL, un système de gestion de données relationnelle. Il est comparable à d'autres outils de gestion de base de donnée notamment enseignés à l'INSA tels que Oracle ou MySQL. Une base de données relationnelle utilise un schéma pour dicter la structure des données stockées. PostGreSQL est typiquement utilisé pour trois raisons principales :

1. Il est fiable et a une intégrité des données performantes, c'est à dire qu'il garantit que les transactions de données vont réussir, selon le principe des propriétés A.C.I.D (atomicité, cohérence, isolation, durabilité).

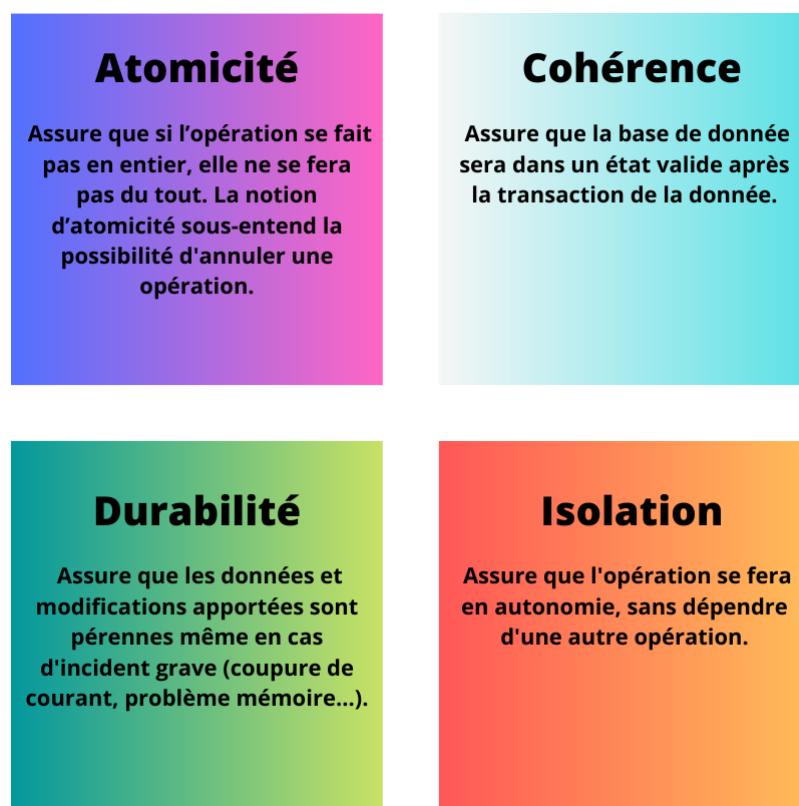


FIGURE 3.4 – Propriétés ACID

Pour notre utilisation du stockage de donnée, c'est sans aucun doute le critère le plus important.

2. Il est très bien maintenu et a une documentation complète. C'est également l'outil le plus conforme aux standards SQL.
3. C'est un logiciel libre qui ne dépend pas d'un contributeur privé.

Il est aussi à noter que PostGreSQL est très utilisé pour sa capacité à gérer les bases de données spatiales car il intègre des bibliothèques de fonctions géographiques avancées.

Plusieurs types de base de données existent et s'appuient sur différents principes de la transaction des données. Les bases de données relationnelles comme la notre se basent sur le principe des propriétés A.C.I.D, ou sur des principes similaires. Parmi les autres types de modèles de bases de données on retrouve par exemple les bases orientées document, ou encore les bases orientées graphe. Nous avons pu voir en cours différents modèles tels que MongoDB pour la base orientées document, ou Cassandra pour la base orientée graphe.

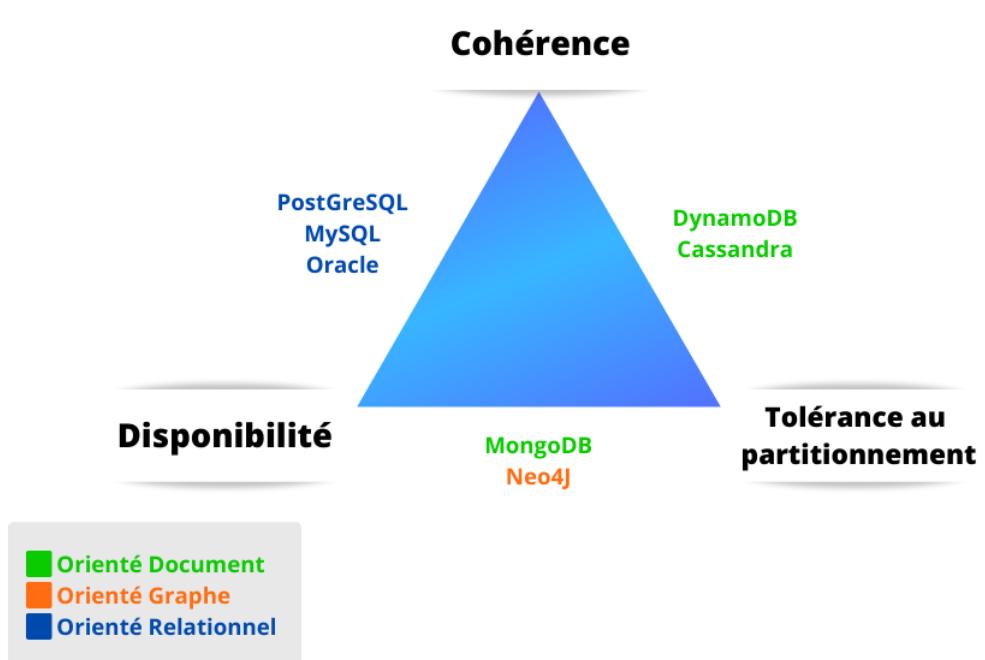


FIGURE 3.5 – Schéma du théorème de CAP

Le **théorème CAP** ou théorème de Brewer [4] est un théorème qui explique que la transaction des données d'un système de stockage d'information ne peut respecter que deux des trois grands principes d'une transaction de donnée, à savoir :

- Consistency** : La cohérence, qui implique de tous les clients voient les même données en même temps.
- Availability** : La disponibilité, qui signifie que tout client qui fait une requête obtient une réponse même si des noeuds sont en panne.
- Partition Tolerance** : La tolérance au partitionnement, qui implique que la base de donnée doit continuer à fonctionner même si un ou plusieurs noeuds sont en panne, ou si il arrive une panne de communication entre les noeuds.

Dans notre cas, l'aspect cohérence et disponibilité est bien plus important que la tolérance au partitionnement. La cohérence est le point le plus important, car il est primordial que chaque client de la base de donnée reçoive les mêmes données même s'ils modifient ces données en temps concurrent. C'est donc bien une base de données orientée relationnel qui semble le plus approprié.

3.2.4 Git

Pour organiser les développements de chaque développeur, nous utilisons Git et le framework GitFlow. Git est un logiciel de gestion de versions décentralisé qui permet de mutualiser les développements, de comparer les modifications et de suivre les versions de chaque composant d'un programme. C'est le logiciel de gestion de version le plus populaire au monde, et qui est également à la base des plus importants hébergeurs de code informatique : GitHub et GitLab.

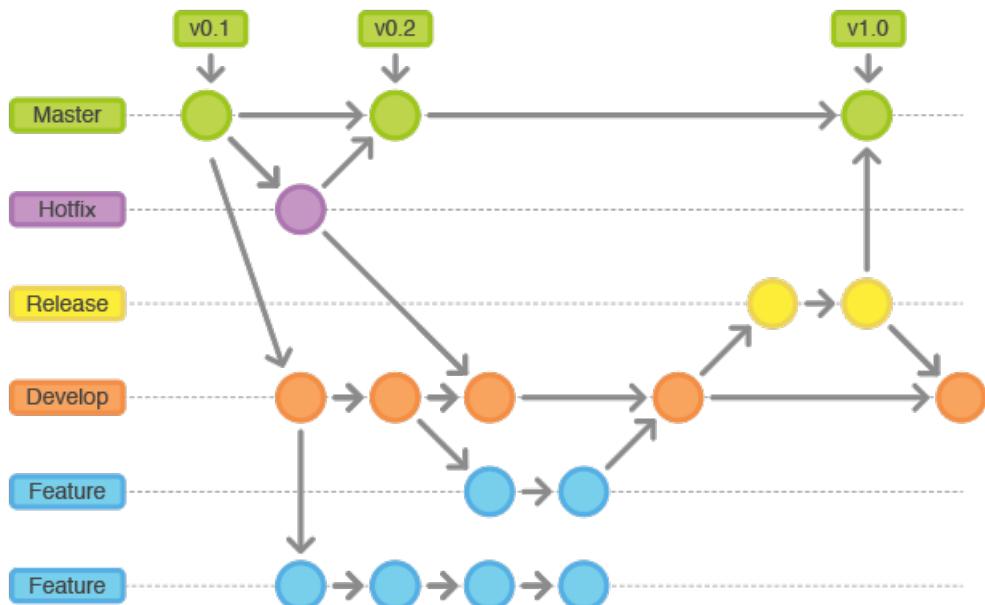


FIGURE 3.6 – Fonctionnement de Git, ou "Git-Flow"

Notre projet utilise GitLab, qui est un concurrent de GitHub. Les deux outils ont beaucoup de points communs, mais se démarquent sur leur "Flow", c'est à dire la manière dont il est conseillé de s'en servir pour mutualiser le code et créer des versions. GitLab est orienté sur la fiabilité, tandis que GitHub favorise la rapidité de développement. Il est donc plutôt conseillé d'utiliser GitLab pour de gros projets comme le nôtre, ce qui est probablement l'une des raisons pour laquelle GitLab a été choisi. Mais surtout, GitLab permet de créer des dépôts privés auto-hébergés sans avoir de licence, ce qui signifie que l'on peut stocker le code sur une machine appartenant à Capgemini, en créant un dépôt GitLab sur notre serveur, et ce sans acheter GitLab contrairement à GitHub. Étant donné que le projet est confidentiel et entièrement sur un réseau local dont les serveurs se trouvent dans nos locaux, GitLab était un choix plus logique et moins onéreux.

Chacun de nos composants possède une branche appelée **develop**. Le développeur se servira de ces branches comme base, et en créera une nouvelle à chaque fois qu'il aura besoin de modifier un composant pour une correction d'anomalie ou une évolution. Chaque développement, chaque correction est positionnée sur une branche particulière **feature**. Une fois finie, elle est "mergée" dans **develop**, ce qui signifie que ses modifications sont ajoutées à la branche de base.

Lorsque tous les développements d'une version sont réalisés, on crée une branche **release**, sur laquelle toutes les modifications sont réunies. Il peut arriver que plusieurs personnes aient modifiés le même fichier pendant la même version. Dans ce cas, on vérifie manuellement quelles sont les modifications et si elles sont compatibles au préalable.

La dernière étape est l'assemblage, qui crée une nouvelle version avec toutes les modifications.

3.3 Mes objectifs

Pendant cette alternance, j'ai pu définir des objectifs communs avec Capgemini en plus de mes objectifs personnels. Mes objectifs professionnels étaient :

1. Contribuer par sa posture et sa communication à véhiculer une image positive de Capgemini chez son client.
2. Comprendre les processus Capgemini encadrant son activité.
3. Être fiable dans l'estimation de sa charge et de son reste à faire. Maîtrise parfaitement son reporting et sait en cas de difficulté alerter rapidement.
4. Être un ambassadeur de Capgemini.
5. Adopter une posture collaborative.

6. Démontrer de l'intérêt, de l'engagement, de la pro-activité dans son quotidien.
7. Faire preuve de flexibilité et de réactivité.
8. Être capable d'interagir avec son écosystème.

Ces objectifs sont communs à chaque employés de Capgemini de Grade A (débutants et alternants). Ils ne sont pas personnalisés. Ils permettent d'encourager l'intégration et l'implication dans le projet et dans l'entreprise.

Il existe également des objectifs attribués en fonction du rôle cible. Dans mon cas, j'ai choisi l'Ingénieur Logiciel qui rajoutait une dizaine d'objectifs qui peuvent être résumés en ces points :

1. Développer son autonomie sur une activité de conception, développement et validation
2. Respecter des normes de codage ainsi que de documentation pour préserver une certaine qualité dans le code et le projet en général
3. Gagner en compétences fonctionnelles côté métier pour comprendre les spécifications clients et établir par la suite de meilleure solution technique
4. Gagner en compétences techniques sur le projet
5. Appliquer des corrections sans régressions et effectuer des chiffrages cohérents au niveau de la technique et de l'estimation
6. Pouvoir alerter en cas de difficultés ou de problèmes particuliers

Tous ces objectifs ont vocation à être validés au travers de "feedbacks" : des retours écrits sur notre travail, qu'il est possible de demander à n'importe quel collaborateurs. Une fois remplis ils permettent d'augmenter son grade, qui représente l'expérience et détermine en grande partie le salaire.

Enfin, il existe les objectifs projet qui sont déterminés au cours d'une réunion avec le chef de projet, dans mon cas Mme Hélène GOLSE. Ils sont au nombre de cinq et se focalisent sur mon amélioration personnelle, ma participation à l'esprit d'équipe et à l'ambiance du projet, ainsi que la capitalisation et le partage de connaissances.

Voici un exemple de feedback qui m'a permis de valider quelques-uns de ces objectifs :



Camille RUSTAN

May 30, 2022

Bien que le projet soit techniquement complexe, tu t'es rapidement adapté et tu avances dans tes résolutions avec une grande rapidité et une grande qualité. De plus, tu prends du recul et tu alertes quand tu vois quelque chose qui sort du cadre (tu m'as alerté sur plusieurs "anomalies" à requalifier en évolutifs, ce qui n'est généralement pas aisés pour un jeune développeur !). Tu es également capable de challenger la partie fonctionnelle en te confrontant aux exigences. En bonus, tu es toujours motivé et souriant, ce qui fait de toi un collaborateur très agréable. Merci, et bravo Matthieu !

FIGURE 3.7 – Exemple de feedback reçu

A chaque trimestre, une réunion appelée "baromètre" se tient entre les supérieurs hiérarchiques et les tuteurs d'entreprise de chaque salarié pour vérifier la validation des objectifs et faire un retour à l'employé, sous forme de note de potentiel, et d'un commentaire. La note peut être Low, Medium ou High, et elle représente la projection vers le grade supérieur. Cette réunion peut permettre d'augmenter le grade si tous les objectifs sont remplis.

Voici un exemple de restitution de baromètre que j'ai obtenu en Juin dernier :

Trimestre 2



Matthieu confirme ses très bons débuts sur le périmètre FE. Sa montée en compétences techniques et fonctionnelles se poursuit sur d'excellentes bases. Il a su gagner la confiance du projet, si bien que le développement de gros évolutifs lui ont été donnés. Ainsi, il entre également en interaction directe avec le client. Ce sera là une bonne occasion de gagner en visibilité hors du périmètre projet interne.

FIGURE 3.8 – Exemple de restitution reçue

3.4 Mes réalisations

3.4.1 La mise en place

La première période de mon alternance fût en partie consacrée à l'installation de mon poste de travail. Le projet est doté d'une documentation très détaillée, ce qui n'est pas le cas de tous les projets d'entreprise. En suivant cette documentation, il faut compter environ trois jours en partant d'un environnement quasiment vierge, avec simplement l'environnement Linux et quelques scripts d'installés. On installe pendant cette étape l'IDE Eclipse ainsi que tous ses frameworks, on crée les workspaces, on copie les dockers, on importe les projets Git et on les ajoute à Eclipse. Une bonne partie du temps est également

consacrée à corriger les problèmes récurrents lors de l'installation de la machine.

Après lecture de la documentation, on réalise une boucle complète avec un ordre de prise de vue. Dans l'environnement de développement, le satellite, la chaîne image et le centre de contrôle ne sont pas interrogés. Nous utilisons à la place des services tels que le TWMU (Tasking Window Management Unit) qui permettent de simuler les réponses des différents services ou infrastructures que le client lui, pourra réellement utiliser.

3.4.2 Les formations

Capgemini m'a donné au début de mon alternance une liste de formation en ligne à réaliser. Ces formations durent environ 3 jours au total, et abordent des sujets comme les conflits d'intérêt, le harcèlement, ou les consignes de sécurité en cas d'incendie. Un court test est à réaliser en fin de formation pour valider les notions vues.

3.4.3 La maintenance

Ma principale activité au cours de ce stage fut la résolution d'anomalie. Je vous présenterai dans cette partie la méthodologie de la résolution d'anomalie, puis j'expliquerai plus en détails quelques-unes des corrections que j'ai pu effectuer parmi les plus notables et qui ne nécessitent pas trop de contexte pour être comprises.

3.4.3.1 Méthodologie

Après avoir choisi un ticket sur notre outil de suivi Redmine, nous devons suivre un procédé pour renseigner nos différents avancements.

La première étape est de tenter de reproduire l'anomalie en fonction des informations dans le ticket, c'est pourquoi il est important lors de la création du ticket d'y renseigner le plus de détails possible. Si le client crée un ticket sans y ajouter suffisamment de détails, nous cessons l'analyse et envoyons directement une demande de complément d'information. S'en suis alors les étapes de la correction du problème, où un développeur va résoudre l'anomalie, puis un validateur testera la correction.

Le processus d'une correction d'anomalie est le suivant :

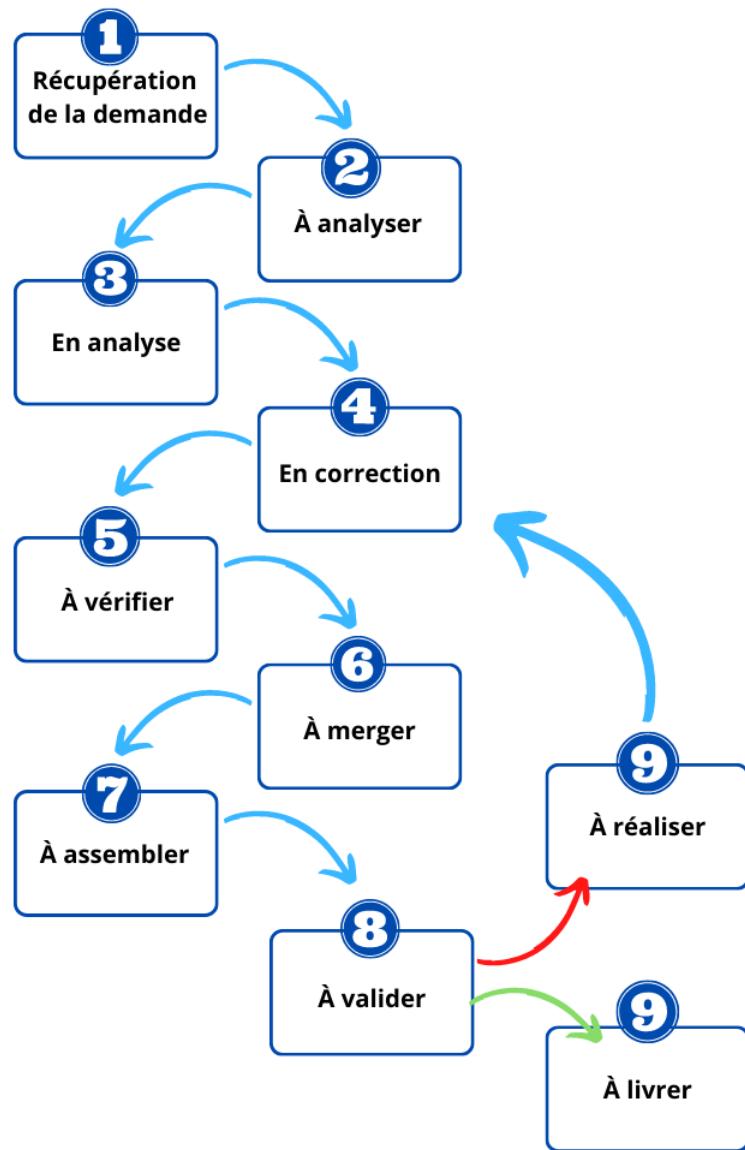


FIGURE 3.9 – Processus de résolution d'anomalie

1. *Récupération de la demande*. L'anomalie est levée par le client et importée sur Redmine. Il est possible qu'un membre de l'équipe lève un ticket en interne, dans lequel cas il sera d'abord mis en visibilité du client pour qu'il valide la réalisation de cette correction.
2. *A analyser*. Le ticket de l'anomalie est visible par les membres de l'équipe qui peuvent se l'assigner.

3. *En analyse.* Un développeur est en train d'analyser le ticket. Il tente de le reproduire, puis de voir d'où vient le problème.
4. *En correction.* Le développeur corrige l'anomalie.
5. *Code à vérifier.* Le code a été partagé sur le Git, et doit être revu par un responsable technique afin d'assurer la qualité du code.
6. *A merger.* La correction est effectuée sur une branche particulière d'un ou plusieurs composants, il faut donc ensuite regrouper ce code avec la branche principale du projet.
7. *A assembler.* On assemble les différentes corrections dans une nouvelle "baseline", une version pour les développeurs.
8. *A valider.* Un membre de l'équipe de validation prend le relais, récupère le dernier assemblage et vérifie que la correction demandée par le ticket a bien été implémentée.
9. *A livrer.* Si le développement est validé il peut alors être inclus dans la version livrée à ADS
ou
A réaliser. Si le développement n'est pas validé, il repasse par un développeur avec de nouveaux éléments d'informations relevés par le validateur. Le développeur repassera alors le ticket à l'étape "En correction".

Il existe aussi l'état *En attente* dans le cas où durant l'analyse, le développeur ne serait pas capable de reproduire l'anomalie par manque d'information, ou si il lui manque des informations pour savoir comment corriger le problème. Dans ce cas, il faut contacter le client qui devra nous préciser sa demande.

Sur toutes les étapes de la réalisation de cette mission, chaque collaborateur va renseigner son temps passé et son reste à faire, afin d'effectuer un suivi du temps et des ressources.

Résoudre les anomalie est une tâche très formatrice, étant donné que certains bugs peuvent parfois nous demander d'utiliser beaucoup d'aspects du projet pour les reproduire. Une fois en analyse de code, il faut souvent explorer plusieurs dizaines de fichier avant de trouver une ligne ayant un rapport avec notre problème, de laquelle on suit l'exécution du programme pas à pas pour comprendre quand l'erreur arrive. Ensuite, on

modifie le code et teste nos modifications, en essayant de faire en sorte qu'il n'y ait pas d'impact sur d'autres fonctionnalités du projet.

La prise en main de l'application est longue et cela peut être frustrant en tant que nouvel arrivant. Heureusement, chacun des développeurs connaît et comprend ce processus. L'entre-aide est très importante entre les plus anciens et les juniors de l'équipe.

Le projet existe depuis déjà plusieurs années, et part d'une base déjà existante. Les anomalies à corriger sont donc souvent très complexes et peuvent nécessiter les connaissances de plusieurs personnes pour être comprises.

De plus, il peut être important de demander l'avis d'un autre membre de l'équipe sur les impacts de ses modifications pour éviter les effets de bord. Ce que l'on appelle la non-régression et l'effet de bord sont des problématiques importantes. Une modification anodine d'une fonction peut parfois changer complètement le fonctionnement du programme sur des aspects qu'il est difficile de prédire. En tant que débutant, il est difficile de se rendre compte de la portée des actions de son code.

3.4.3.2 Anomalies corrigées

Ceci n'est pas une liste exhaustive de tous les tickets sur lesquels j'ai travaillé mais plutôt des exemples parmi les anomalies les plus intéressantes qui m'ont été assignées.

Mon premier ticket en tant que développeur sur le projet était un travail de correction d'une évolution récemment assemblée qui ne fonctionnait pas. Précédemment, plusieurs paramètres de distance devaient être renseignés car les différents satellites utilisés par le logiciel n'utilisaient pas les mêmes unités de mesure. Certains exprimaient cette notion de taille de l'image en mètres, et d'autres en "GSD" pour Ground Sample Distance. L'évolution permettait d'utiliser un seul et même paramètre pour décrire la résolution au sol de l'image à prendre, et convertissait automatiquement ce paramètre en l'unité demandée par chaque satellite. Les valeurs renvoyées n'étaient apparemment pas les bonnes, ce qui impliquait une modification du serveur.

Après une analyse du code grâce aux outils d'Eclipse que j'ai pu commencer à prendre en main lors de cette tâche, il ne me semblait pas que le code était incorrect mais la valeur renvoyée pouvait parfois être très éloignée des valeurs habituelles pour ce genre de paramètres (plusieurs dizaines de kilomètres, alors que la valeur est généralement de l'ordre du mètre). L'erreur venait en fait simplement d'un manque d'information quant à la formule de conversion proposée par Airbus DS qui devait prendre des angles en radians,

alors que les angles dans l'application sont communiqués en degrés.

J'ai donc renseigné toutes ces informations sur Redmine, ainsi que le temps nécessaire à la correction, à savoir quelques heures pour la reproduction de l'erreur, une demi-journée pour l'analyse du code, une demi-journée pour la correction de l'erreur et la vérification de son fonctionnement. Reste à ajouter les modifications sur Git pour mettre en commun les développements. Cette correction s'était concentrée sur un seul composant, ce qui évite une étape de mise à jour des dépendances entre les composants impactés, et facilite la mutualisation du code.

Pour ajouter les modifications sur Git, on commence par créer une branche avec un nom adéquat, qui mentionne le numéro du ticket et son nom, en suivant les consignes de dénomination. Ensuite, on peut alors pousser nos modifications sur cette branche par les étapes de "add", "commit" et "push".

Avant de valider complètement notre développement, on utilise un script permettant de vérifier que les règles de codage en rigueur soient respectées, appelé "Checkstyle". Ce type de script est selon moi très important, surtout pour un projet d'envergure, car il encourage les développeurs à avoir une syntaxe propre et plus facilement lisible pour les prochains membres de l'équipe qui liront leur code. Le checkstyle vérifie la syntaxe des nouvelles méthodes ou classes, la dénomination des différentes variables, et si différentes réglementations mises en place par l'équipe de développement sont respectées. Je pense qu'il est primordial d'instaurer ce type de vérification sur tous les projets informatique sur lesquels plus de cinq personnes travaillent, et qu'il aurait même pu être intéressant de le rendre plus strict sur certains aspects tels que la dénomination, ou le rendre automatique à la sauvegarde de fichier java par exemple.

L'anomalie précédente était parmi les plus simples au niveau du code, mais était unique dans sa solution. Les corrections d'anomalies suivantes nécessitaient plus d'analyse du programme, mais celle-ci restait un bon début pour s'initier à Eclipse, à l'interface de création d'ordre de prise de vue du logiciel, ainsi qu'aux nombreuses commandes Linux permettant d'accélérer mes prochains développements.

J'ai ensuite effectué une modification du client dans un ticket intitulé "Comments should be editable for Activated Orders" ce qui signifie que les commentaires attribués à une demande d'acquisition de photo devraient pouvoir continuer à être modifiables même si la demande a été envoyée au satellite. La modification de l'interface peut être relativement laborieuse étant donné le nombre de frameworks utilisés, mais elle a l'avantage de ne pas nécessiter une relance entière du serveur à chaque modification du code, ce qui

peut prendre jusqu'à cinq minutes.

Un sujet sur lequel j'ai beaucoup travaillé durant mon alternance était un système permettant de donner des fichiers au format XML au logiciel pour qu'il enregistre automatiquement un ordre de prise de photographie. J'appellerai ces fichiers les **FCO** pour Fichier de Création d'Ordre, le nom réel étant confidentiel. Un FCO permet d'éviter de passer par l'interface pour créer un ordre de prise d'image, et de renseigner tous les paramètres voulus. La zone d'intérêt, la date de validité, le niveau de production et tous les autres paramètres peuvent être décris sous la forme d'un fichier XML que l'on dépose dans un dossier connu par l'application. Le système ingère tous les fichiers présents dans le dossier ayant un nom et une forme approprié, et ce toutes les minutes. C'est ce qu'on appelle un **polling**, une technique d'interrogation continue et séquentielle qui vérifie qu'il n'y ait pas de données à transférer. Le temps entre deux polling est paramétré grâce à des expressions **CRON** qui se rédige sous le format suivant :

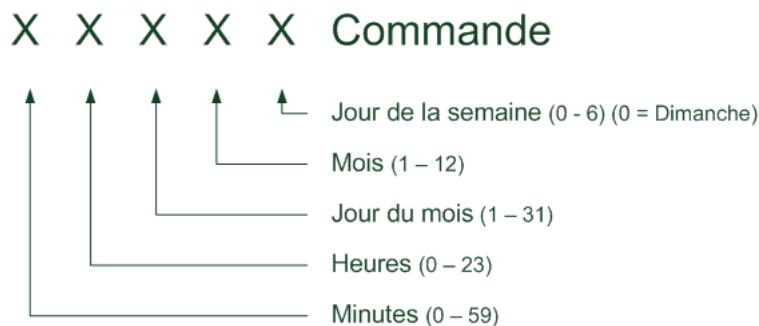


FIGURE 3.10 – Syntaxe du format CRON

Caractère	Utilisation
*	Valeur quelconque
,	Séparateur
-	Intervalle
/	Step

TABLE 3.1 – Caractères du format CRON

Par exemple, pour avoir un polling qui s'effectue toutes les minutes, on utilisera l'expression basique : * * * * *. Tandis que pour un polling qui s'effectue à 12h30 tous les 9 Janvier on écrira l'expression suivante : 30 12 9 1 *[3]

Mon premier ticket concernant les FCO impliquait une modification de ces derniers de manière à autoriser l'utilisateur à s'en servir d'une nouvelle manière :

En y décrivant un ordre de production et de livraison, l'application devrait pouvoir être capable d'ingérer plusieurs images et toute les traiter. Précédemment, une seule image pouvait être utilisée par un FCO. Ce ticket nécessite donc une modification du serveur de manière à lui faire accepter un format d'FCO différent.

Pour cette correction, il m'a donc fallu tout d'abord créer des fichiers FCO habituels, et d'autres comportant deux images, pour tester les différents comportements de l'application. J'ai ensuite passé un temps important à vérifier qu'autoriser plusieurs produits n'entraînait pas de régression ou d'effet de bord, et que l'application pouvait réaliser l'entièreté de ses fonctions après avoir ingéré un FCO avec deux produits. Bien que les validateurs aient pour but de réaliser ce genre de tests, laisser une correction être assemblée et validée par un autre collaborateur est une grosse perte de temps si elle ne fonctionne pas.

J'ai également eu des ticket que j'ai dû passer en état "En attente d'information" étant donné que je n'arrivais pas à reproduire l'anomalie. L'anomalie suivante est un exemple qui met en perspective la pénibilité que peut représenter ce genre de situation.

Un ticket concernant l'interface décrivait un problème qui empêchait l'utilisateur de supprimer un certain capteur d'une liste de satellites pour le calcul du planning de la prise de photo. Il existe trois parties différentes de l'application qui correspondent à cette description. Après une demi-journée à tester chacun de ces trois éléments de l'interface dans un maximum de comportements possible, j'ai décidé de contacter le client pour obtenir plus d'informations sur ce problème. Il faut savoir que lorsqu'une anomalie n'est pas reproduite et annulée, elle est facturée pour seulement 30% du prix d'une anomalie corrigée. Il faut donc réagir rapidement si on ne réussit pas à reproduire, il n'est pas possible de passer plusieurs jours sur des tentatives même si il reste des cas qui n'ont pas été testés. Ici, la réponse du client indiquait un numéro de version sur laquelle l'anomalie avait été relevée, et un élément de l'interface à vérifier, qui s'est avéré par la suite ne pas être le bon. Le changement de version prend environ une heure, ce qui retarde encore la reproduction de l'erreur, et ces échanges avec le clients se sont répété trois fois sans plus d'avancement. Dans ce genre de cas, nous pouvons être amenés à se rendre dans les locaux de ADS pour qu'ils nous montrent en direct les anomalies que nous ne parvenons pas à reproduire. Les paramètres et leur environnement d'exécution peuvent varier du nôtre et causer des dysfonctionnements imprévus et difficilement reproductibles. Au 30 Août, cette anomalie n'a toujours pas pu être reproduite, alors que mes premières tentatives datent de Mars.

3.4.4 Les évolutions

A partir de Mai, j'ai pu entreprendre des travaux différents de la correction d'anomalie : les évolutions. Une évolution a pour but d'ajouter une ou plusieurs nouvelles fonctionnalités au projet. Je remercie Capgemini de m'avoir fait confiance pour la réalisation d'évolutions qui sont souvent réservées aux membres de l'équipe avec assez d'expérience. Pour organiser une évolution, ADS et Capgemini suivent les étapes décrites ci-dessous :

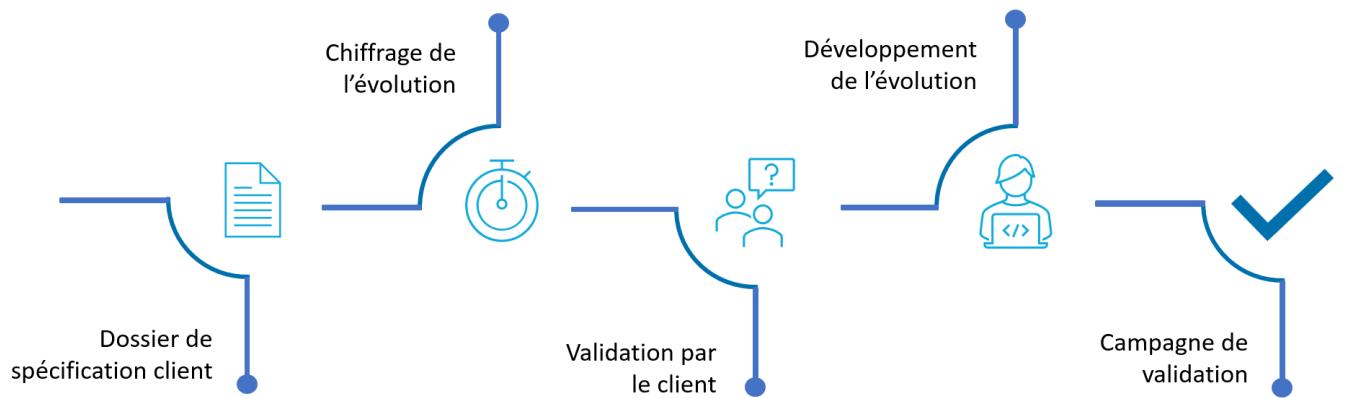


FIGURE 3.11 – Réalisation d'une évolution

1. *Nouvelles spécifications client.* ADS rédige les nouvelles exigences pour le projet, des dossiers dans lesquels il décrit précisément toutes les nouvelles fonctionnalités à implémenter.
2. *Chiffrage de l'évolution.* Capgemini accepte l'évolution, et un développeur expérimenté va alors la chiffrer. Cette étape consiste en l'analyse des modifications du code qu'entraînera l'évolution, ainsi que de tous les impacts que pourraient avoir ces modifications. Le développeur estime ensuite le temps que prendra chaque étape, et nous proposons ensuite le chiffrage à ADS.
3. *Validation ou refus du client.* Le client analyse à son tour le chiffrage, et décide s'il accepte ou non de payer pour ces modifications.
4. *Développement de l'évolution* Un développeur programme l'évolution en suivant les directives du chiffrage. Contrairement à une correction d'anomalie, il est courant qu'il faille rajouter de nouveaux fichiers et impacter de nombreux composants lors de ce processus.
5. *Campagne de validation.* Tout comme pour les anomalies, les validateurs passent derrière les développeurs pour vérifier le bon fonctionnement de l'application avec les

modifications apportées, et réalisent tous les tests de non-régression habituels.

Ma première évolution était un système similaire aux FCO vus précédemment, qui permettent de créer des ordres de prise de vue dans l'application sans passer par l'interface. Ces nouveaux fichiers, que j'appellerai FCOW pour Fichier de Création d'Ordre Web, permettent également de créer des ordres qui seront envoyés depuis un client web au lieu d'être directement placés dans un dossier par l'utilisateur, pour rendre le système plus accessible.

Il a donc fallu développer un nouveau système de polling comme vu plus tôt, des fonctions permettant de récupérer le contenu d'un fichier, de vérifier que ce dernier suit bien le format XML et qu'il est correctement nommé. Il a ensuite fallu faire en sorte que l'application ingère le contenu du fichier comme étant un nouvel ordre, en rajoutant les nouvelles informations à la base de donnée grâce au framework Hibernate.

J'ai également pu réaliser un chiffrage avec un autre développeur sur une petite évolution. Celle-ci abordait un sujet relativement simple, mais nécessitait des changements de l'interface qui peuvent parfois être plus complexes que prévus. C'est pourquoi notre chiffrage a nécessité des modifications du code, pour tester les répercussions des modifications et avoir une meilleure idée du temps que prendrait la personne en charge de l'évolution pour tout développer. En principe, un chiffrage est simplement une analyse du code sans développement. Mais il est parfois important de savoir si les modifications visibles au premier regard ne sont pas que la partie émergée, pour ne pas sous-estimer le temps nécessaire.

Enfin, la tâche qui m'a pris le plus de temps et que j'espère pouvoir terminer avant la fin de mon alternance est également une évolution très intéressante. Elle doit permettre à l'utilisateur de modifier les satellites reconnus par l'application en lui transmettant un fichier XML qui décrit les caractéristiques de nouveaux satellites. Actuellement, le logiciel reconnaît uniquement une liste fixe de satellite : ceux du client, et quelques autres satellites loués à différentes entités.

Pour cette évolution, j'ai pu obtenir de nombreux documents extrêmement détaillés de la part de Thalès Alenia Space ainsi que d'Airbus Defense and Space. Ces dossiers regroupaient de multiples diagrammes, et une liste quasiment exhaustive de toutes les dénominations et normes que devraient suivre le développement de mes nouvelles fonctionnalités. Plusieurs réunions avec ADS ont été réalisées pendant le développement, d'abord pour décider des derniers points et spécifications qui n'avaient pas été mis au clair dans les documents, et également pour répondre à mes interrogations quant à la façon de faire.

Comme j'avais déjà pu développer pour ma précédente évolution, celle-ci nécessitait un système de polling et de parsing pour récupérer les fichiers mentionnés, ainsi qu'un échange avec la base de donnée. Cette étape fût plus longue que prévue dû à la différence entre les fichiers FCO et ce nouveau type de fichier. De plus, une grosse évolution réalisée en parallèle par un autre développeur entrait en jeu. Cette autre évolution devait permettre à l'application de pouvoir se servir de satellite RADAR et non plus seulement de satellite de optiques. Le code de cette évolution n'étant pas encore assemblé, j'ai dû travailler en admettant que j'aurais ensuite le programme permettant de faire fonctionner les satellites RADAR, qui doivent également être inclus dans mon évolution.

C'est donc a priori la dernière tâche que j'accomplirai sur le projet, et également de très loin la plus impactante :

Elle touche pour l'instant cinq composants, rajoute un package Java ainsi qu'une quinzaine de classes Java. Elle rajoute aussi plusieurs fichiers de configuration et modifie de nombreuses classes déjà existantes. Elle nécessite de comprendre beaucoup de modules et d'aspects du projets, certains même dont je n'avais jamais entendu parler. Pour mettre ces chiffres en perspective, les corrections d'anomalies que j'avais pu effectuer jusque là impactaient au maximum deux composants, nécessitaient en général la modification d'une ou deux classes, et je n'avais encore jamais créé de classe ou de fichier dans l'application.

3.4.5 Confronter les exigences

Un travail important concerne la compréhension et l'analyse des exigences. Celles-ci sont inscrites dans des documents fournis par le client, et doivent entièrement décrire le comportement attendu du logiciel. Une fois validées par tous les partis, elles doivent être suivies à la lettre et ne peuvent être modifiées qu'au travers d'une demande d'évolution qui sera chiffrée.

Durant mon alternance, il m'est arrivé couramment de m'attribuer un ticket de correction d'anomalie qui décrivait un comportement supposément inattendu, mais qui me semblait être correct ou du moins non-précisé par les spécifications.

Ce genre de ticket pouvait par exemple être une simple modification d'un nom dans l'interface, auquel cas je choisissait de simplement corriger selon la demande client sans passer par le chiffrage étant donné que la correction me prenait moins d'un jour. Mais j'ai aussi eu des tickets nécessitant un chiffrage complet. Par exemple, un ticket abordant le sujet de la couverture nuageuse indiquait un comportement inattendu : lorsqu'une photographie de la Terre était prise avec une couverture nuageuse supérieure au critère d'acceptation, l'image n'était pas livrée au client et l'acquisition de l'image était automa-

tiquement reprogrammée. Le client indiquait que ce n'était pas normal et qu'il devrait recevoir toutes les images, même celles qui ne validaient pas ce critère. Après une vérification dans les dossiers de spécifications, j'ai pu me rendre compte que le comportement observé était bien le comportement décrit dans les exigences. J'ai donc informé le responsable fonctionnel que le ticket de maintenance devrait être transformé en évolution, et donc qu'il devrait être discuté avec le client puis chiffré avant d'être implémenté.

3.4.6 La transmission des connaissances

Pendant cette année d'alternance j'ai pu me former sur beaucoup de sujets différents du projet, j'ai pu découvrir et corriger certains problèmes rarement rencontrés par l'équipe. Les problèmes d'environnement sont nombreux, et peuvent ne pas avoir de solution encore connue. Quand ce genre de problème survient, il est important capitaliser l'information, notamment en complétant le wiki du projet. On y renseigne la manière de corriger l'erreur, accompagné de mots clés et des messages d'erreurs renvoyés par l'application, de façon à ce que les prochains collaborateurs puissent trouver facilement une solution à des problèmes déjà résolus auparavant.

Un des problème que j'ai rencontré était une erreur inconnue qui nécessitait de modifier une version de python pour l'exécution de certains scripts. La résolution de l'erreur s'est faite par des commandes trouvées au bout de plusieurs jours de blocage, et qui n'étaient pas mentionnées sur le wiki. Il était donc important de les noter sur un espace accessible à tous les membres de l'équipe.

Chapitre 4

Conclusions et perspectives

Capgemini m'a permis de réaliser une expérience extrêmement enrichissante, aussi bien d'un point de vue technique, fonctionnel et social. J'ai eu l'opportunité de participer à un grand projet très complexe d'un domaine qui me passionne, avec une équipe soudée et qualifiée.

Le spatial est un secteur vaste et intéressant dans lequel j'ai beaucoup appris, et qui a de grandes chances de m'être utile dans mes prochaines années que je compte passer à Toulouse.

J'ai découvert les problématiques d'un projet impliquant de nombreux acteurs. Entre développeurs et valideurs, mais aussi avec les clients. J'ai aussi pu comprendre les enjeux que recèlent la gestion de projet, le partage de connaissance et de code, la confidentialité, la maintenance d'une application et la cohésion dans une équipe d'une vingtaine de personnes. J'ai pu y appliquer de nombreuses notions vues à l'INSA, ce qui m'a permis d'améliorer ma compréhension globale et parfois d'y trouver des intérêts que je n'avais pas saisi en cours.

J'ai pu améliorer mon sens de l'analyse de part mes différentes corrections d'anomalies, et surtout ma capacité à comprendre un code d'envergure déjà existant, chose qui n'est pas souvent abordée à l'INSA. J'ai pu faire avancer ce projet de nombreuses manières : en réalisant de multiples corrections de bugs, en ajoutant plusieurs fonctionnalités au projet, en communiquant avec le client, et en faisant partie de la chaîne d'entraide très importante des collaborateurs.

J'ai pu renforcer ma rigueur de code en me rendant compte de la difficulté de comprendre un programme auxquels des centaines de personnes ont contribué. J'ai pu comprendre la nécessité des tests rigoureux et de l'application des consigne de développement. J'ai pu mieux saisir les enjeux des méthodes de gestion de projet et de l'importance de

participer à leur bon déroulement ainsi qu'à l'évolution de ces processus.

Enfin j'ai beaucoup appris des entreprises de conseils, de leur façon de procéder et d'organiser leurs équipes, mais aussi le côté administratif. En tant qu'alternant, j'ai pu être concerné par des problématiques administratives, des avantages et inconvénients d'être salarié plutôt que stagiaire, ce qui m'a aidé dans ma recherche d'emploi à comprendre certaines notions et spécificités de chaque contrat.

Côté perspectives de carrière, Capgemini m'a proposé un contrat à durée indéterminée, en me laissant choisir de continuer sur ce projet ou sur un autre sujet du spatial et de l'aéronautique. Mais après de longues réflexions, j'ai finalement opté pour une autre opportunité dans une entreprise de conseil similaire à Capgemini, toujours à Toulouse. J'espère pouvoir y apprendre autant que pendant cette année au sein de la Mission Chain Run, dans des domaines toujours aussi vastes et intéressants mais sur des projets à taille plus humaine, et avec moins de restrictions quant à la présence sur le site. J'espère aussi pouvoir réaliser une mission d'un an à l'étranger prochainement, pour remplacer l'Erasmus que je n'ai pas pu effectuer durant mes superbes années passées à l'**INSA**.

Annexes

Annexe 1 : Présentation de la société



FIGURE 4.1 – Logo de Capgemini

Capgemini est la plus grande entreprise de conseil informatique de France, avec plus de 325'000 salariés dans le monde et un chiffre d'affaire de 18 160 000€ en 2021. Elle fut créée en 1967 par Serge Kampf. Son siège social est basé à Paris. A Toulouse, son bâtiment principal se trouve au 109 Avenue du Général Eisenhower, et elle y embauche plus de 2000 salariés.[2]



FIGURE 4.2 – Capgemini à Toulouse

Capgemini est une "ESN" pour Entreprise de Service Numérique, aussi appellé SSII pour "Société de Services et d'Ingénierie en Informatique". Capgemini propose de réaliser des projets informatiques pour différents clients dans de nombreux domaines industriels différents. A Toulouse, les principaux secteurs d'activités sont évidemment l'aéronautique et le spatial, mais aussi la finance et la santé.

Capgemini se divise en 5 sous-branches spécialisées dans la réalisation de différentes solutions techniques. Ma branche, anciennement "CSD" pour "Custom Software Développement", a été renommée pendant mon alternance en "C&CA" pour "Cloud and Custom Applications". Elle se spécialise dans la réalisation de projets très spécifiques au client tel que le mien, et dans le développement des services Cloud qui sont en très forte expansion sur le marché du numérique ces dernières années. [1]

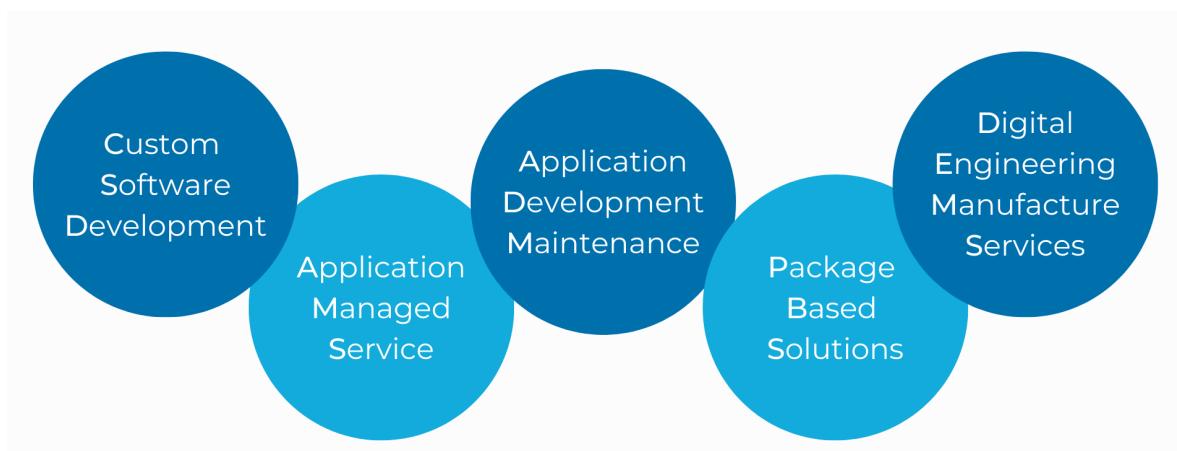


FIGURE 4.3 – Les cinq pratiques de Capgemini

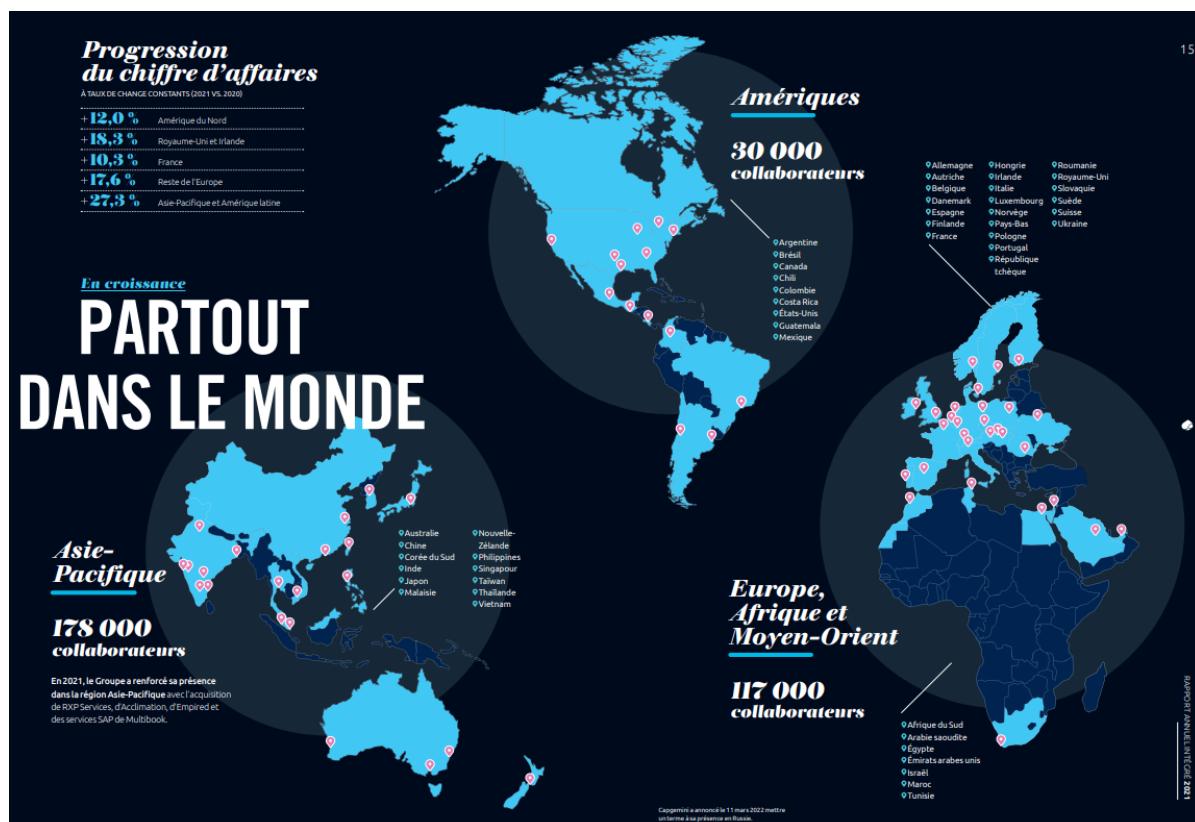


FIGURE 4.4 – Statistiques d'embauche de Capgemini dans le monde

Table des figures

2.1	Le B612 à Montaudran	3
2.2	Organigramme simplifié	4
2.3	Les différentes chaînes du projet	6
2.4	Le satellite Pléiade Neo 3	8
2.5	Schéma des orbites LEO, MEO, GEO et HEO	8
2.6	Schéma de l'orbite du satellite d'observation	9
2.7	Photographie de Toulouse par un satellite Pléiades Neo (Airbus DS)	12
3.1	Schéma de la méthodologie du Cycle en V	16
3.2	Architecture du logiciel	18
3.3	Architecture du client	19
3.4	Propriétés ACID	20
3.5	Schéma du théorème de CAP	21
3.6	Fonctionnement de Git, ou "Git-Flow"	22
3.7	Exemple de feedback reçu	25
3.8	Exemple de restitution reçue	25
3.9	Processus de résolution d'anomalie	27
3.10	Syntaxe du format CRON	31
3.11	Réalisation d'une évolution	33
4.1	Logo de Capgemini	40
4.2	Capgemini à Toulouse	40
4.3	Les cinq practices de Capgemini	41
4.4	Statistiques d'embauche de Capgemini dans le monde	42

Bibliographie

- [1] Capgemini, profil entreprise et chiffres clés. <https://www.capgemini.com/fr-fr/notre-groupe/profil-entreprise-chiffres-cles/>, 2022. [Online ; accessed 28-August-2022].
- [2] Capgemini, rapports annuels 2021. <https://investors.capgemini.com/fr/rapports-annuels/?fiscal-year=2021/>, 2022. [Online ; accessed 28-August-2022].
- [3] Cronitor. Cron editor expression. <https://crontab.guru/>, 2021. [Online ; accessed 31-July-2022].
- [4] Nancy Lynch Seth Gilbert. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.20.1495&rep=rep1&type=pdf>, 2021. [Online ; accessed 31-August-2022].
- [5] Geert Verhoeven. Resolving some spatial resolution issues. https://www.researchgate.net/publication/328353906_Resolving_some_spatial_resolution_issues_-_Part_1_Between_line_pairs_and_sampling_distance, 2018. [Online ; accessed 30-June-2022].