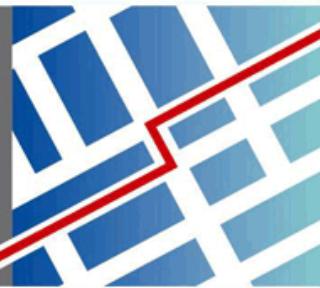




Business Geografic



RAPPORT DE STAGE HUGO JACQUEL CIRIL GROUP JUIN & JUILLET 2023

STAGE

RÉALISÉ PAR

HUGO JACQUEL

hugo.jacquel.pro@gmail.com

*Ciril Group
Villeurbanne*

2023
BTS SIO 1 | SLAM

VERSION CONTEXTE DATE	AUTEUR
V1.0 RENDU DE MI-STAGE 15/06/2023	HUGO JACQUEL
V2.0 FIN DE STAGE 05/07/2023	HUGO JACQUEL

Table des matières

1 Préface	3
1.1 Aperçu	3
1.2 Remerciements	3
2 Qui suis-je ?	4
3 Présentation de l'entreprise	5
4 Lexique	7
5 Documentation Technique : Vue 3	10
5.1 Qu'est ce qu'un component?	10
5.2 Cycle de vie	10
5.3 Différence Vue 3 vs Vue 2	12
5.4 Composition API	12
6 Objectif du stage	13
6.1 Présentation de la plateforme GEO	13
6.2 Contexte Mission principale : Migration de stack Vue 2 -> Vue 3	15
6.3 Missions secondaires	15
7 Déroulé du stage	16
7.1 Introduction et début dans l'entreprise	16
7.2 Mission principal : Migration de stack Vue 2 -> Vue 3	17
7.2.1 Installation de la stack GEO	17
7.2.2 Découverte du framework Vue 3 "Histoire"	18
7.2.3 Séparation logique UI & logique métier	18
7.2.4 Entraînement de reproduction	20
7.2.5 Packeting Node JS NPM	22
7.2.6 Ré-implémentations en Composition API	24
7.2.7 Problème d'héritage objet & d'alias	27
7.2.8 Isolation et TDD des applications Vue dans Histoire	28
7.3 TDD : Debug Test Unitaire sur Elastic Search & Lucène Java Maven	28
7.3.1 Qu'est-ce que ElasticSearch & Lucène	28
7.3.2 TDD & Test Unitaire : Procédure	29
7.3.3 Merge Request : Review et Sonarqube	32
8 Méthodologie Agile et organisation de l'entreprise	36
8.1 Définition et présentation de la méthodologie	36
8.2 Outils utilisé	38
8.2.1 Gestion et stockage du code avec Gitlab	38
8.2.2 Organisation Agile & Kanban avec Jira	40
8.2.3 CI/CD : Organisation automatisé des tests de déploiement et unitaire avec Jenkins .	43
8.3 Système de version avec GEO	43
8.4 Implémentation dans l'entreprise	44
8.4.1 Point Synchro & Stand up	44
8.4.2 Sprint Review	47
8.4.3 Rétrospective	48
8.4.4 Entretien perso	48
8.5 Documentation ADR	48
9 Conclusion général	49
9.1 Conclusion du stage	49
9.2 Compétence validé en Bloc 1	50

10 Documentation utilisateur	51
10.1 Présentation de l'application	51
10.2 Utilisation des ressources GEO	53
10.2.1 Données	53
10.2.2 Fonctionnalités	54
10.2.3 Cartothèque	54
10.2.4 Applications	55
10.3 Installation	55
11 Documentation technique	56
11.1 Vocabulaire Technique	56
11.2 Cycle de vie : Architecture logicielle	58
11.3 Logique de construction de requêtes	61
11.3.1 Typologie de requête	61
11.3.2 Structure de requête	61
11.4 Installation	62

1 Préface

1.1 Aperçu

Ce document représente un aperçu de l'évolution de mon expérience professionnelle pour mon stage réalisé chez Business Geographic (filiale de **Ciril Group**) à propos de ma mission principal et des missions secondaires que j'ai pu réaliser.

Ce document est vérifié par **Cyril RICHARD** et **Olivier SOTIN** sur l'utilisation de captures d'écran dans le but d'illustration de mes travaux sur le projet. Certains documents sont censurés pour des raisons évidentes dans le cadre d'un accord de confidentialité des données.

La lecture du Lexique est recommandé pour comprendre l'essentiel des termes techniques génériques et lié à l'entreprise utilisé dans ce rapport.

1.2 Remerciements

Je souhaite remercier l'intégralité des personnes, qui m'ont aidé, accompagné et encouragé tout au long de mon stage et pour son organisation :

Institution des Chartreux

- **Mickaël CHAVAND** : Directeur des Formations diplômantes à l'Institution des Chartreux
- **Fanny COUDER** : Professeure d'informatique et adjointe de direction à l'Institution Des Chartreux
- Mes professeur(e)s d'informatique à l'**Institution des Chartreux**

pour leur aide sur ma **convention de stage**, l'organisation du **forum des stages** qui m'a permis de découvrir plusieurs entreprises et de rencontrer l'entreprise **Ciril Group** et pour l'aide qu'ils ont pu me fournir tout au long de mon stage et de mon année qui m'ont permis de mener à bien ce stage.

Ciril GROUP

- **Amaël GRIVEL** - Président de Ciril GROUP
- **GRIVEL Rémi** - Directeur Général chez Ciril GROUP
- **Philippe MARCO** - Directeur Technique (CTO) chez Ciril GROUP
- **Cyril RICHARD** - Lead Developer R&D chez Business Geografic (Ciril GROUP)
- **Olivier SOTIN** - Manager de l'équipe R&D chez Business Geografic (Ciril GROUP)
- **Fanny LE CHATELIER** - Product Owner chez Business Geografic (Ciril GROUP)
- **Jérôme ICARD** - Product Owner chez Business Geografic (Ciril GROUP)
- Les autres **membres de Ciril GROUP** et les **alternants**

pour leurs accompagnements tout au long de mon stage, le temps qu'ils ont pris pour m'expliquer des concepts et des notions fondamentales dans le monde de l'entreprise côté technique, informatique et organisationnel.

2 Qui suis-je ?

Développeur Fullstack Vue 3 - Étudiant en BTS SIO 1 option SLAM
(Développement) à l’Institution des Chartreux, Croix-Rousse (Lyon)



FIGURE 1 – Hugo Jacquel

Principalement intéressé dans le **développement low-level** (C, C++), et les différents aspects sécurités qui sont liés à chaque sous-domaine de l’informatique (notamment les concepts de reverse-engineering, analyse de CVE, exploitation de payload sur différents axe que ce soit injection SQL ou XSS, élévation de privilège, RCE, etc.), je consacre désormais une partie de mon domaine d’étude aux réseaux informatiques, au développement web (notamment avec l’apprentissage des frameworks JavaScript adapté en fonction des besoins), et à d’autres sous domaines lié à l’informatique (comme l’**OSINT** et la **théorie du signal**).

J’ai obtenu mon BAC général spécialisation **Mathématique, Numérique Science Informatique (NSI)** et option **Mathématique expert** au **LPO Lycée polyvalent Rouvière** à Toulon. Par la suite, j’ai enchaîné sur une année de prépa mathématique & informatique à **EPITA Lyon**. Après avoir tenter l’examen d’entrer à l’école 42 où j’ai pu passer un mois à programmer en C & Bash de manière intense, j’ai commencé et finalisé une première année en **BTS SIO** à **l’Institution des Chartreux** à Lyon.

J’ai pris la spécialité **SLAM (Solutions Logicielles et Applications Métiers)** pour pouvoir me spécialiser dans le développement de solutions multiples en les créant à l’aide de schématisation d’architecture et en les programmant à l’aide des technologies adapté selon les besoins.

L’entreprise **Ciril GROUP** porte un grand intérêt au sein de **ma formation d’ingénieur** car elle est spécialisée dans des notions qui m’intéresse (gestion de donnée géographique, développement orienté front et back, grand projet dense et riche en structure) mais aussi expert en terme d’organisation (avec la méthodologie agile / scrum, la hiérarchie et l’utilisation de certains outils comme des normes / standards mais aussi des logiciels d’organisation professionnel).

3 Présentation de l'entreprise



Ciril Group est une entreprise technologique spécialisée dans le développement de logiciels et de solutions informatiques pour les secteurs publics et privés. Fondée en 1985, l'entreprise a son siège social à Paris, en France, et est présente dans plusieurs pays à travers le monde.

L'histoire de Ciril Group remonte à ses débuts en tant que fournisseur de services de conseil et de développement de logiciels pour les collectivités locales en France. Au fil des années, l'entreprise a élargi sa gamme de produits et services pour répondre aux besoins croissants des clients du secteur public et privé.

Les activités principales de Ciril Group sont centrées sur trois domaines clés :

- **Solutions logicielles pour les collectivités locales** : Ciril Group propose une gamme complète de solutions logicielles destinées aux collectivités locales, telles que les mairies, les conseils départementaux et les services municipaux. Ces solutions comprennent la gestion financière, la gestion des ressources humaines, la gestion des services publics, la gestion des urbanismes, etc. Elles visent à améliorer l'efficacité opérationnelle et à faciliter la prise de décision dans ces organismes.

- **Solutions pour les entreprises et les secteurs spécifiques** : Ciril Group propose également des solutions logicielles adaptées aux besoins des entreprises et des secteurs spécifiques tels que l'éducation, la santé, les transports et l'environnement. Ces solutions sont conçues pour répondre aux défis particuliers de chaque domaine et pour aider les organisations à gérer leurs opérations de manière plus efficace et plus rentable.

- **Services de conseil et d'accompagnement** : Ciril Group offre des services de conseil et d'accompagnement pour aider les clients à mettre en place et à optimiser l'utilisation des solutions logicielles. Ces services comprennent l'analyse des besoins, la planification de projet, la formation des utilisateurs et le support technique.

Ciril Group est pionnier en France dans le domaine et dans le service du SIG et du générateur SIG sur demande grâce à la plateforme GEO. En ce qui concerne les produits spécifiques, Ciril Group propose une large gamme de logiciels intégrés et modulaires, tels que Ciril Finance, Ciril RH, Ciril Urbanisme, Ciril Environnement, Ciril Éducation, Ciril Santé, etc. Ces produits sont continuellement mis à jour pour suivre les évolutions technologiques et les besoins changeants de ses clients.

Au fil des années, Ciril Group s'est positionné comme un acteur majeur dans le domaine des solutions logicielles pour les collectivités locales et les entreprises. Son engagement envers l'innovation, la qualité et

le service client lui a permis de gagner la confiance de nombreux clients et de s'étendre à l'international. Aujourd'hui, Ciril Group continue de développer de nouvelles solutions et de renforcer sa présence sur le marché mondial de l'informatique publique et privée.



ÉDITION DE PROGICIELS
ET SYSTÈMES D'INFORMATION
DÉDIÉS AUX ACTEURS PUBLICS





ÉDITION DE SOLUTIONS
ET SYSTÈMES D'INFORMATION
GÉOGRAPHIQUES (SIG)





HÉBERGEMENT CLOUD
SAAS, IAAS, HOUSING
**CERTIFIÉ HAUTE SÉCURITÉ
HAUTE DISPONIBILITÉ**





Gamme de produits :



préparation budgétaire, exécution monétaire,
suivi analytique...



paie, carrière, effectifs, formation, recrutement,
bilan social, ...



périscolaire, petite enfance, processus
électoraux, recensement, ...



cadastral, zonages, urbanisme...

4 Lexique

Ci-dessous, retrouvez le lexique technique utilisé dans ce document :

- **Ressources** : Configuration fonctionnelle sur GEO (configuration utilisateur -> Comment et Quelles sont les actions sur les données)

- **Générateur** : Application SaaS qui permet de créer des cartes : c'est le cœur de la solution GEO, générer des cartes personnalisables selon les besoins des utilisateurs et clients.

- **Atelier Générateur Logiciel (AGL)** : Terme générique pour désigner une application SaaS qui permet de créer des logiciels pour créer quelques choses (comme GEO)

- **Gabarit (Canva)** : Terme technique pour parler d'une interface de disposition de contenu graphique (ici sur une page HTML)

- **Aigle Web Serveur (AWS)** : Terme technique pour désigner le nom de l'application qui combine le **générateur GEO** et le serveur web associé programmé en Java (frontEnd)

- **SDK (Software Development Kit)** : Ensemble d'outils et de ressources qui permet aux développeurs de créer des applications logicielles pour une plateforme spécifique. Notamment pour fournir une collection de bibliothèques, de composants, de documentations et d'exemples de code pré-construits qui simplifient et accélèrent le processus de développement en fournissant des fonctionnalités prêtes à l'emploi.

- **Briques** : Technologies utilisées dans un projet (JavaScript, Java, etc.)

- **Flux** : Protocoles utilisés dans un projet (HTTP, HTTPS, SMTP, etc.)

- **CRUD (Create Read Update Delete)** : Ce sont 4 opérations de bases pour manipuler des données dans un système d'information (Création, Lecture, Mis à jour, Suppression)

- **Front-end** : Partie de l'application / système que les utilisateurs voient et avec laquelle ils interagissent. (design et mise en forme)

- **Back-end** : Partie de l'application / système qui gère les opérations en coulisse, comme le stockage des données, la gestion des requêtes et la logique métier. Ce qui est le plus souvent, invisible aux yeux des utilisateurs.

- **Logique métier** : Règles et processus spécifiques à un domaine d'activité ou à une entreprise. Elle représente la logique et la logique de fonctionnement qui régissent les opérations essentielles d'une organisation, indépendamment des détails techniques de mise en œuvre. (traitement des données, prise de décisions, exécutions des opérations et gestion des flux, etc.)

- **Logique UI (DOM + Style)** : Gestion de l'apparence et de l'interaction d'une interface utilisateur (UI). Ici dans le cadre de l'application GEO utilisant le framework Vue, on parle de logique DOM et Style (DOM car c'est la l'architecture visuel pour les applications web) et style pour la personnalisation des couleurs, la mise en forme et le design.

- **LTS (Long Time Support) | EOL (End Of Life)** : Caractéristique appliquée à une version pour dire quelle sera maintenue par son éditeur longtemps pendant 18 mois avant de passer en OEL, où ici seul les bugs et les patch de sécurité sont appliqués pendant encore 18 mois. Après les 18 mois de la EOL, la version du logiciel n'est plus supportée par l'éditeur.

- **Framework** : Technologie logicielle qui facilite le développement d'applications en fournissant des bibliothèques et des outils (dans le cadre de GEO, les frameworks utilisés sont Vue 3 et Angular JS)

- **Middleware** : logiciels intermédiaires qui permettent la communication entre différentes applications (dans le cadre de GEO, on utilise Apache)

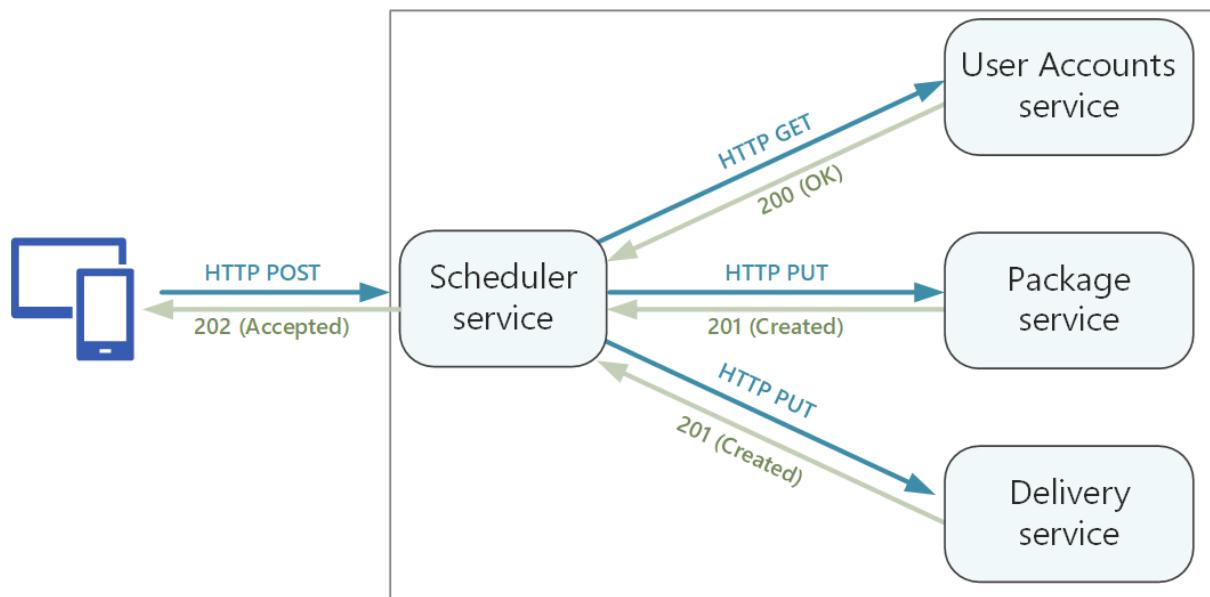
- **Stack** : Combinaison spécifique de technologies logicielles utilisées pour développer et exécuter une application ou un système (C'est l'ensemble des Système d'exploitation, Serveur Web, Base de données, langage de programmation, framework, middleware utilisés pour créer et déployer la solution)

- **Docker** : Technologie de conteneurs (application isolé du reste de l'ordinateur) permettant la centralisation via l'encapsulation des composants d'un logiciel, des services, des bibliothèques, des dépendances et les fichiers de configuration nécessaires à son fonctionnement.

- **Template** : Fichier ou une structure préconçue qui sert de base ou de point de départ pour créer d'autres documents ou applications. Il est conçu pour être réutilisé avec des modifications spécifiques selon les besoins de l'utilisateur.

- **API (Application Programming Interface)** : Ensemble de règles, de protocoles et de définitions qui permettent à différents logiciels de communiquer et d'interagir entre eux. L'API agit comme une interface intermédiaire qui permet à des développeurs d'accéder aux fonctionnalités et aux données d'une application ou d'un service spécifique de manière sécurisée et contrôlée

- **Endpoint API** : URL spécifique à laquelle une requête peut être envoyée pour accéder à une ressource ou exécuter une action spécifique. Il s'agit essentiellement d'une adresse ou d'un point d'accès sur le serveur de l'API. Chaque endpoint est associé à une fonctionnalité spécifique fournie par l'API, et il peut être utilisé pour récupérer, créer, mettre à jour ou supprimer des données à partir de la ressource ciblée.



- **Interface** : Prototype d'un objet qui permet de fournir de manière abstraite des attributs et des méthodes en précisant uniquement les types des arguments et les types de retour. Ces prototypes sont essentiels en développement car ils permettent d'être rigoureux dans l'instanciation d'objet.

- **Factory (Design Pattern)** : Modèle de conception d'instanciation qui fournit une interface pour créer des objets sans spécifier explicitement leur classe concrète. Au lieu de créer directement des instances d'objets à l'aide du constructeur, on fait appel à une méthode de création (factory method) qui s'occupe de créer l'objet approprié en fonction d'un certain contexte ou de paramètres donnés

- **CI/CD (Continuous Integration / Continuous Delivery)** : Pratique dans la gestion de projet en informatique pour automatiser et accélérer le processus de développement et de déploiement des logiciels. Il vise à garantir une livraison régulière et fiable des applications en combinant l'intégration continue, qui consiste à fusionner régulièrement les modifications du code dans un référentiel partagé, et le déploiement continu, qui automatise la mise en production des nouvelles versions du logiciel.

- **SIG | GIS** : Système d'information géographique : Un système qui permet de collecter, gérer, analyser et visualiser des données liées à la géographie.

- **Méthodologie Agile** : Approche de gestion de projet qui favorise la flexibilité, la collaboration et l'adaptation continue aux changements. Elle vise à répondre aux besoins changeants des clients et à améliorer la qualité du produit final. L'une des particularités de l'Agile est son approche itérative et collaborative, où le travail est organisé en cycles courts appelés "itérations" ou "sprints". Chaque itération se concentre sur des objectifs spécifiques et aboutit à une version utilisable du produit.

- **Sprint** : Période de temps définie et fixe au cours de laquelle une équipe de développement travaille sur un ensemble spécifique de tâches. C'est une unité de temps délimitée, généralement d'une durée de 1 à 4 semaines, pendant laquelle l'équipe se concentre sur la réalisation d'objectifs définis.

- **Test unitaire** : Ce sont des tests réalisés sur des fonctions d'un programme pour tester leurs comportements selon une multitude de paramètre. L'objectif est de voir si les fonctions testées ont le bon comportement selon les circonstances.

- **Test de non régression** : Ce sont des tests unitaires qui restent présent même après la validation du comportement de la fonction testée, l'objectif est de dire si une nouvelle fonctionnalité ou un changement dans le code va modifier le comportement d'une fonction déjà testée. C'est un principe essentiel dans le CI/CD notamment dans l'optique de "l'intégration continue".

- **Event** : Action spécifique qui se produit au sein de l'application, généralement déclenchée par une interaction de l'utilisateur (clic de souris, appui sur une touche, etc.). L'événement contient des informations sur cette action (type d'événement, les coordonnées de la souris, les touches enfoncées, etc) et sert à fournir à une fonction qui agit grâce à cet événement (callback).

- **Callback** : Fonction qui est passée en tant qu'argument à une autre fonction ou méthode : elle est exécutée ultérieurement, généralement en réponse à un événement ou à une action spécifique. (notamment pour réagir à des événements / actions déclenchés par les composants ou les directives Vue 3).

- **Lifecycle Hook** : Événement et callback associé à des événements dans la vie d'une application Vue (on parle de directives pour un composant) (voir Cycle de vie Vue 3)

- **IDE (Integrated Development Environment)** : Environnement de développement intégré qui regroupe plusieurs outils et fonctionnalités (souvent personnalisables) pour faciliter le processus de développement de logiciels. Il s'agit d'un logiciel qui offre un ensemble complet d'outils pour écrire, éditer, compiler, déboguer et tester du code source dans différents langages de programmation.

5 Documentation Technique : Vue 3



L'objectif du framework javascript Vue 3 est de réaliser du développement d'application web basé sur le Component Driven développement (CDD)

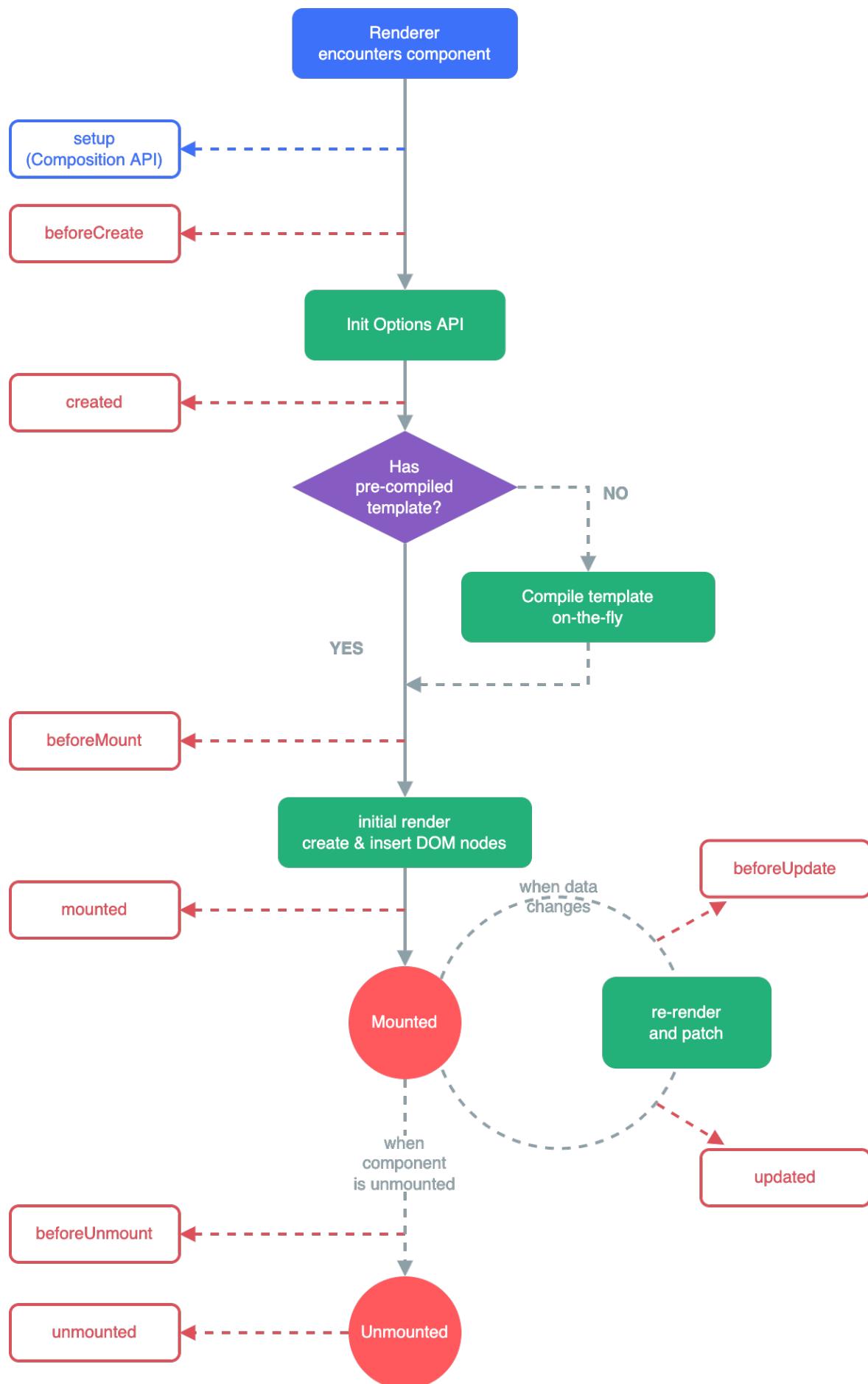
5.1 Qu'est ce qu'un component ?

Un component est une structure de données isolée des autres et qui est disponible dans l'ensemble du projet. Cette structure est composée de :

- Un **block HTML "template"** : qui est le contenu du block visuellement parlant
- Un **block script** : qui permet d'exécuter un script (associé à l'objet le plus souvent) pour le faire interagir selon des events particuliers et des variables locales
- Un **block style** : pour personnaliser le style de l'objet selon nos besoins (souvent avec la propriété CSS scoped pour qu'il n'affecte que l'objet dans lequel il est écrit)

5.2 Cycle de vie

Chaque objet à un cycle de vie où il est "monté" sur une page .vue Plusieurs events sont associé à l'objet et "trigger" (déclencher) selon les moments (avant la création, lors de la création, au moment du mount, lors de l'unmount, etc.)



5.3 Différence Vue 3 vs Vue 2

Pour ma mission j'ai du faire une liste des différences entre les 2 dernières version majeures de Vue à se basant sur le changelog (journal de mis à jour) ici :

- **Taille réduite et performances améliorées** : La version 3 de Vue.js est plus légère et plus rapide que la version 2, grâce à une refonte interne et à l'utilisation d'un nouveau compilateur. Cela se traduit par des temps de chargement plus rapides et une meilleure réactivité de l'application.

- **Composition API** : La Composition API est une nouvelle approche de la structuration des composants Vue. Elle offre une meilleure séparation des préoccupations et facilite le partage de logique entre les components. Elle permet également de gérer plus facilement les effets secondaires et d'améliorer la réutilisabilité du code.

- **Réactivité améliorée** : La version 3 introduit un nouveau système de réactivité appelé "Proxy-based Reactivity", qui offre des performances accrues et une gestion plus fine des dépendances réactives. Cela se traduit par une réactivité plus efficace lors de la mise à jour des données.

- **Composition de transition** : La version 3 propose un nouveau moyen de gérer les transitions et les animations grâce à la composition de transition. Cela permet de créer des animations plus complexes et plus personnalisées, en utilisant des transitions de manière plus flexible.

- **Optimisations du rendu** : La version 3 inclut plusieurs optimisations du rendu, telles que le rendu conditionnel plus efficace, l'évitement des re-rendus inutiles et l'amélioration des performances lors du rendu des listes.

- **Prise en charge améliorée des types** : La version 3 offre une meilleure prise en charge des types grâce à l'utilisation de TypeScript. Cela facilite le développement et la maintenance des applications Vue.js en fournissant une meilleure validation des types et une aide à l'auto complétion dans les éditeurs de code.

5.4 Composition API

La composition API est une fonctionnalité introduite dans Vue 3 pour améliorer la manière dont on **organise** et **réutilise** la logique des components. Elle vise à résoudre certains problèmes rencontrés avec l'API Options de Vue 2, tels que la complexité croissante des components, la difficulté à réutiliser la logique entre les components et le manque de clarté dans la structure du code.

La composition API propose une approche plus fonctionnelle :

- **Dissociation de la logique UI et de la logique métier** : La composition API permet de séparer clairement la logique UI (affichage, gestion des événements, etc.) de la logique métier (appels API, gestion des états, etc.). Vous pouvez définir la logique métier dans des fonctions séparées et les réutiliser dans différents components.

- **Fonction ref()** : La fonction ref permet de créer une référence réactive à une valeur. Cela permet de gérer plus facilement les états réactifs et de partager ces états entre différentes fonctions ou components. Les valeurs encapsulées dans un ref peuvent être lues et modifiées, et les mises à jour seront automatiquement répercutées dans les components qui les utilisent.

- **Fonction inject()** : La fonction inject permet d'accéder aux valeurs fournies par des parents via le mécanisme de "provide/inject" de Vue. Cela facilite la communication entre les components sans avoir à passer les données de parent à enfant via des props.

L'objectif principal derrière l'utilisation de la composition API est la bonne dissociation de la logique UI et métier pour faciliter la lecture / lisibilité du code mais aussi pour de futurs implémentations dans le générateur GEO

6 Objectif du stage

6.1 Présentation de la plateforme GEO

Je réalise mon stage dans le département **Recherche et développement (R&D) de Ciril Group**, l'objectif de mon stage est de participer au développement de la plateforme GEO de Business Geographic.

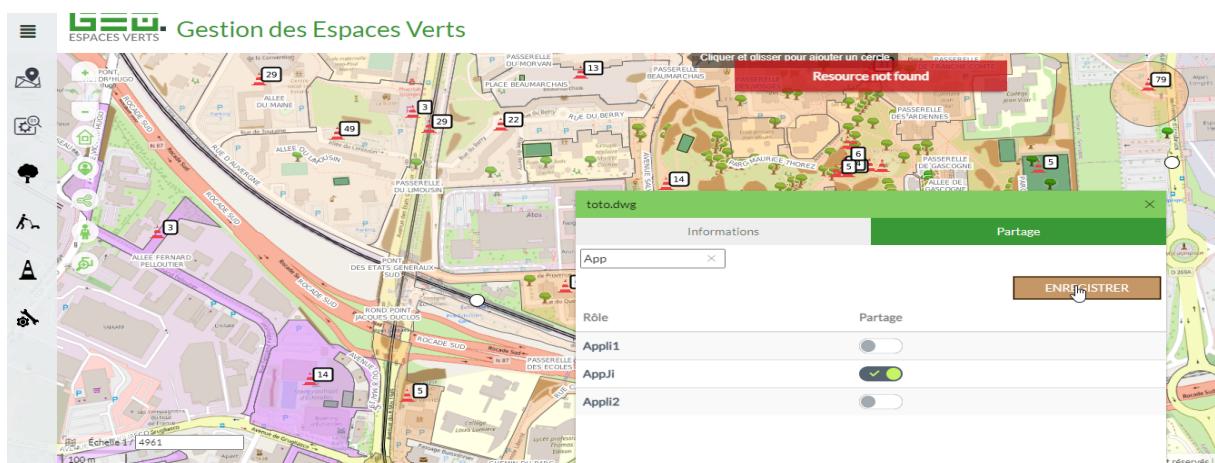


GEO est une application **SIG (Système d'Information Géographique)**, **SaaS** (Software as a Service) qui permet de créer et diffuser des applications cartographiques riches de sens, en mode HTML5 responsive, à partir de données géographiques et métier.

FIGURE 2 – Logiciel SIG Générateur GEO

L'outil GEO est ergonomique et disponible hors connexion sur différentes plateformes (le logiciel étant une webapp, il suffit d'un navigateur pour pouvoir accéder à l'application). Le logiciel embarque avec lui, plusieurs fonctionnalités :

- **Création de schéma** en overlay (superposition de la carte) personnalisable selon les besoins. (polygone, flèches, texte, surlignage de zone, etc.)
- **Option d'analyses spatiales** à valeur ajoutée : géocodage, isochronie, calcul d'itinéraires, optimisation de tournées, etc.
- **Génération de graphiques statistiques & de diagrammes** à partir des données fournies sur la carte.



La solution GEO est à destination du grand public et pour toute entreprise qui a besoin d'utiliser / gérer des données géographiques.

GEO est surtout utilisé par les **collectivités publiques (mairie, département, région, etc.)** pour faire la **gestion routière, éclairage, électrique, télécommunications, réseau d'égout, etc.**



Le générateur d'applications GEO





Les applications métier



GEO EAU POTABLE



GEO AC



GEO ANC



GEO ECLAIRAGE



GEO CADASTRE



GEO ZONAGES



GEOXALIS



GEO ESPACES VERTS



GEO SI ROUTIER



GEO VOIES



GEO ENFANCE



GEO ÉLECTIONS



Gamme de produits :



Interopérabilité et puissance : génération d'application full-web, applications professionnelles, grand public, mobile, API



Modularité et précision métier : cadastre, urbanisme, eau, assainissement, routes, éclairage public...



Aide à la décision : géomarketing, télécommunications, observatoires du territoire, finance...

Ciril GROUP possède quelques concurrents en France et dans le monde :

- **1spatial** (Concurrent et acteur mondial avec une filiale en France)
- **GEOMOD** (spécialisé dans la géomatique et de la modélisation, essentiellement maritime et gestion des eaux)
- **YPRESIA** (eau et assainissement)

6.2 Contexte Mission principale : Migration de stack Vue 2 -> Vue 3

La solution GEO est composée en plusieurs parties :

- Une partie **logiciel** : Programmé en Java, cette partie gère tout le côté software de l'application est représenté le **générateur** ainsi que les différentes fonctionnalités moteur associées au générateur.

- Une partie **web et visuel** : qui sert d'interface graphique pour les utilisateurs, anciennement programmé avec le framework web Angular JS puis migré en Vue Js.

La mise à jour du framework Angular JS de sa version 1.0.0 vers des versions plus récentes ont changer fondamentalement l'utilisation de Angular : nouvelle architecture, meilleure performance, nouveau langage de templating et une API plus riche.

Dans la théorie, la mise à jour du framework est une bonne nouvelle. Cependant dans la pratique, le changement d'infrastructure était plutôt instable et assez difficile à mettre en place.

Après décision suite à un ADR sur le sujet, la stack Angular JS comme framework web fut changer pour le framework Vue JS.

De plus, la prise de cette décision s'est accentué avec l'entrée en LTS de AngularJS 1.7 qui a été maintenu par Google (les créateurs) jusqu'au 30 juin 2021, ce qui a obligé Ciril Group à redéfinir sa stack front de GEO.

A ce moment là, la version du framework Vue était la 2. L'équipe de Business Geografic avait réalisé la migration de GEO sous Vue 2.7.

La fin de la migration se conclut par la sortie de Vue 3 et l'annonce de la fin de la LTS et du EOL de la version 2.7 du framework.

Pour des raisons de sécurités et de stabilités, il était essentiel de migrer de Vue 2.7 vers une version 3 de Vue.

Mon objectif était donc de migrer l'interface écrit en Vue 2.7 vers une version 3 pour ne pas se confronter à des risques de sécurité.

Is Vue 2 Still Supported?

Vue 2.7, which was shipped in July 2022, is the final minor release of the Vue 2 version range. Vue 2 has now entered maintenance mode: it will no longer ship new features, but will continue to receive critical bug fixes and security updates for 18 months starting from the 2.7 release date. This means **Vue 2 will reach End of Life on December 31st, 2023.**

FIGURE 3 – Annonce de la fin de maintenance "End of Life" de la version 2 du framework Javascript Vue

6.3 Missions secondaires

Tout au long du stage, je réalise plusieurs autres missions dans le cadre de mon étude personnel et de ma curiosité mais aussi des missions alternatives au sein de l'entreprise :

- Étude des Design Pattern
- Étude des Design Principles
- Étude de la méthodologie Agile

7 Déroulé du stage

7.1 Introduction et début dans l'entreprise

Pour commencer mon stage au sein de Ciril Group, **Cyril Richard**, Lead Développeur de Business Graphic, m'a expliqué l'organisation de l'entreprise et du fonctionnement de **leur système d'information géographique (SIG) GEO**.

Pour mieux connaître et m'expliquer ces concepts et l'organisation de GEO, on utilise un **diagramme C4** :

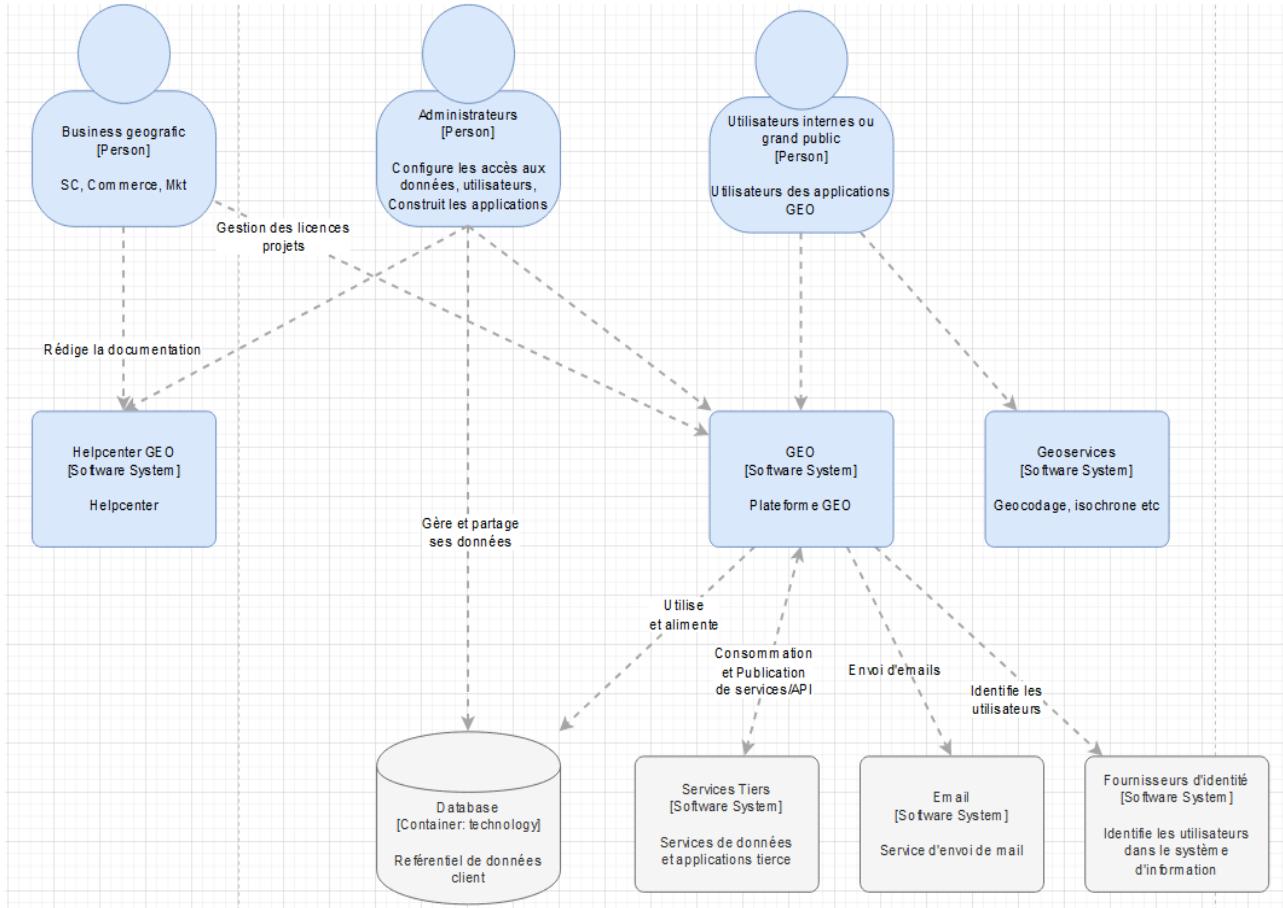


FIGURE 4 – En bleu : l'accès / connaissance des données de l'entreprise Business Geographic

Les diagrammes servent à modéliser de manière simple le fonctionnement d'un projet à travers différentes structures.

Ici sur le diagramme on visualise 3 types d'acteurs principales au sein de l'écosystème GEO :

- **Les salariés** de Business Geographic : Ce sont les membres de l'entreprise, ils ont accès aux Helpcenter GEO (qui est un service client personnalisé pour pouvoir faire évoluer les demandes clientes et les traiter). Mais aussi à l'application GEO en SaaS (directement hébergé chez l'entreprise) et disponible au grand public.

- **Les clients (directs)** de Business Geographic : Ce sont les clients de Business Geographic, le plus souvent des collectivités publiques ou des organisations nationales qui souhaitent utiliser le générateur GEO pour pouvoir créer leurs propres cartes (que ce soit à destination internet / au sein de l'entreprise ou pour ses clients (au grand public). Ils ont eux aussi accès au Helpcenter GEO car ce sont les clients et qui veulent faire évoluer l'application vers des innovations et des besoins spécifiques.

- **Les clients des clients** de Business Geographic : Ce sont les clients des entreprises / organisations qui achète le générateur GEO. Par exemple, on se situe dans cette catégorie si on utilise une application

de carte personnalisé. Les clients ont accès au générateurs GEO (le produit de Business Geographic) (car se sont eux aussi des potentiels clients de GEO) et l'accès au "logiciel" (généré grâce au générateur) de l'entreprise cliente de Business Geographic

L'importance de ce diagramme réside dans les couleurs est les **accessibilités** des acteurs sur les données :

- **En bleu** : l'accessibilité / connaissance des données de l'entreprise Business Geographic (comme par exemple les ressources initiales pour le bon fonctionnement de l'application)

- **En gris** : les données que Business Geographic n'a pas accès (comme par exemple la base de données client car ce sont des données confidentielles)

L'objectif de ces types de diagrammes et de décrire avec précision le fonctionnement d'un mécanisme et son organisation en mettant en détails tout les aspects nécessaires (normes, standards, protocoles, technologies, et.)

Business Geographic respecte le consortium de l'**OGC (Open Geospatial Consortium)** qui permet de respecter un certain nombre de standards dont :

- **WMS (Web Map Service)** : permet la diffusion de cartes géographiques sur le Web. Il fournit des images de cartes géoréférencées à partir de données stockées dans un serveur SIG. Le client peut alors envoyé des requêtes selon un certains nombre de paramètres pour afficher la carte selon ses besoins.

- **WFS : Web Feature Service** : Permet d'interroger et de récupérer des données géographiques vectorielles, telles que des points, des lignes et des polygones, à partir d'un serveur SIG. Cela permet aux clients de récupérer des informations géographiques détaillées et d'effectuer des analyses spatiales avancées. L'ensemble de ces capacité à récupérer ces types de données sont appelé des fonctionnalités (Feature)

- **WMTS : Web Map Tile Service** : permet d'améliorer les performances de diffusion des cartes géographiques en ligne. Il utilise une approche basée sur les tuiles (tiles), permettant un affichage rapide et fluide des cartes, en particulier lors de l'exploration et du zoom interactifs (performance).

7.2 Mission principal : Migration de stack Vue 2 -> Vue 3

7.2.1 Installation de la stack GEO

Pour commencer ma mission, après avoir récupérer mon matériel à la DSI, j'installe la stack GEO (l'ensemble des ressources des technologie pour pouvoir faire tourner l'application) :

- **Stack-geo** : Image docker qui permet de démarrer les ressources (plugins, sous-services de GEO, base de données) pour faire fonctionner GEO (backend)

- **AWS (Aigle5 Web Server)** : Générateur GEO + gabarit GEO -> service web implémenté en Java, à la fois backend logiciel et frontend pour l'affichage

- **Geo - sources** : Extension / Nouvelle version du gabarit GEO appelé "Facette" (car fonctionnement sur une architecture basée sur la recherche à facettes)

Plusieurs objectifs viennent à moi sur cette mission :

- Comprendre **le fonctionnement** de l'application GEO dans son **architecture**

- **Isoler** la logique métier (algorithme) de la logique UI (interface et affichage graphique) pour permettre l'utilisation du générateur GEO et de Facette pour de nouveaux produits / projet au sein de Ciril Group.

7.2.2 Découverte du framework Vue 3 "Histoire"

Pour comprendre les termes techniques : Documentation Technique : Vue 3

Pour démarrer mon stage et pour "expérimenter" le développement de component sur Vue 3, je m'intéresse au component "Histoire" sur Vue.

L'objectif d'Histoire est de réaliser des "Storys" :

- L'organisation et la documentation de components pour les autres développeurs (car les components peuvent être destiné pour l'utilisation d'autres développeurs) Component Driven développent (CDD)

- Les components sont isolés les uns des autres (environnement isolé)
- Découverte de nouvelles fonctionnalités et components
- Le test des components qu'on développe notamment visuellement avec la Visual Regression Test

L'objectif d'Histoire est de faciliter le développement de component Vue 3, de pouvoir les tester rapidement et efficacement. Les components étant isolé on peut :

- utiliser la même pipeline (CI/CD) (développement, intégration et déploiement)
- Avoir un HMR (Hot Module Replacement) (action d'ajout/modification/suppression de modules dans une application pendant qu'elle tourne sans le rechargement total) plus rapide.

7.2.3 Séparation logique UI & logique métier

Le véritable objectif de la mission n'est pas entièrement basé sur le besoin de mettre à jour la stack web de GEO (Vue) mais de **séparer la logique métier de la logique UI**. Pour permettre l'utilisation du **générateur** et de **facet** extérieure au produit GEO (et au-delà du SIG)

Je premier lieu je regarde les lignes pour analyser ce qu'elles font et pour identifier les problèmes relatives à la mise à niveau de version.

Une partie du code incorrect vient de l'utilisation des filtres (des fonctions réutilisables qui permettent de modifier la valeur d'une expression dans les templates. Ils sont utilisés pour formater ou transformer les données avant de les afficher), ici utiliser pour traduire du texte selon le langage utilisé pour GEO.

```

const registerUI = registrationCallbackFactory([
  { type: 'filter', alias: 'translate', impl: translateFilter },
  { type: 'filter', alias: 'base64Img', impl: base64ImgFilter },
  { type: 'filter', alias: 'highlight', impl: highlightFilter },
  { type: 'filter', alias: 'extensionName', impl: extensionNameFilter },
  { type: 'filter', alias: 'resourcePreview', impl: resourcePreviewFilter },
  { type: 'filter', alias: 'bgFormatDate', impl: formatDate },
  { type: 'directive', alias: 'bg-click-outside', impl: clickOutsideDirective },
  { type: 'directive', alias: 'bg-infinite-scroll', impl: infiniteScrollDirective },
  { type: 'directive', alias: 'bg-resource-drag', impl: resourceDragDirective },
  { type: 'directive', alias: 'bg-resource-interaction', impl: resourceInteractionDirective },
  { type: 'component', alias: 'bg-card', impl: Card },
  { type: 'component', alias: 'bg-box', impl: Box },
  { type: 'component', alias: 'bg-resource-card', impl: ResourceFormatCard },
  { type: 'component', alias: 'bg-generator-header-tool', impl: GeneratorHeaderTool },
  { type: 'component', alias: 'bg-generator-header', impl: GeneratorHeader },
  { type: 'component', alias: 'bg-generator-top-tools', impl: GeneratorTopTools },
  { type: 'component', alias: 'bg-dropdown-vue', impl: Dropdown },
  { type: 'component', alias: 'bg-dropdown-line-vue', impl: DropdownLine },
  { type: 'component', alias: 'bg-facet-panel', impl: FacetPanel },
  { type: 'component', alias: 'bg-facet-tag', impl: FacetTag },
  { type: 'component', alias: 'bg-facet-taglist', impl: FacetTagList },
  { type: 'component', alias: 'bg-facet-taglist-add', impl: FacetTagListAdd },
  { type: 'component', alias: 'bg-facet-result', impl: FacetResult },
  { type: 'component', alias: 'bg-facet-search', impl: FacetSearch },
  { type: 'component', alias: 'bg-facet-view', impl: FacetView },
  { type: 'component', alias: 'bg-resource-actions', impl: ResourceActions },
  { type: 'component', alias: 'bg-creation-launcher', impl: CreationLauncher },
  { type: 'component', alias: 'bg-resource-explorer-hub', impl: ResourceExplorerHub },
  { type: 'component', alias: 'bg-resource-explorer-panel', impl: ResourceExplorerPanel },
  { type: 'component', alias: 'bg-generator-view', impl: GeneratorView },
  { type: 'component', alias: 'bg-resource-toggle-sort', impl: GeneratorToggleSort }
]);

```

export { BgInjectionContainer, registerUI };

J'ai tout simplement repris la fonction et en l'important dans chacun des fichier à l'exécuter avec "une syntaxe classique".

```

FacetPanel.vue 9 | TS index.ts | TS TranslateFilter.ts X
src > modules > ui > common > filters > TS TranslateFilter.ts > ...
1 import { GeneratorUiFacadeImpl } from '../../../../../common/facade/GeneratorUiFacadeImpl';
2
3 export default function translateFilter(value: string) {
4   const bgFacade = GeneratorUiFacadeImpl.getGeneratorUiFacade();
5
6   if (!bgFacade.getI18nService()) {
7     return value;
8   }
9   return bgFacade.getI18nService().instant(value);
10}
11

```

Au fil de mes journée de débogage, je rencontre plusieurs variétés d'erreur :

- Incompatibilité de librairie
- Erreur de type / conversion incorrect (par manque des classes renseigné)

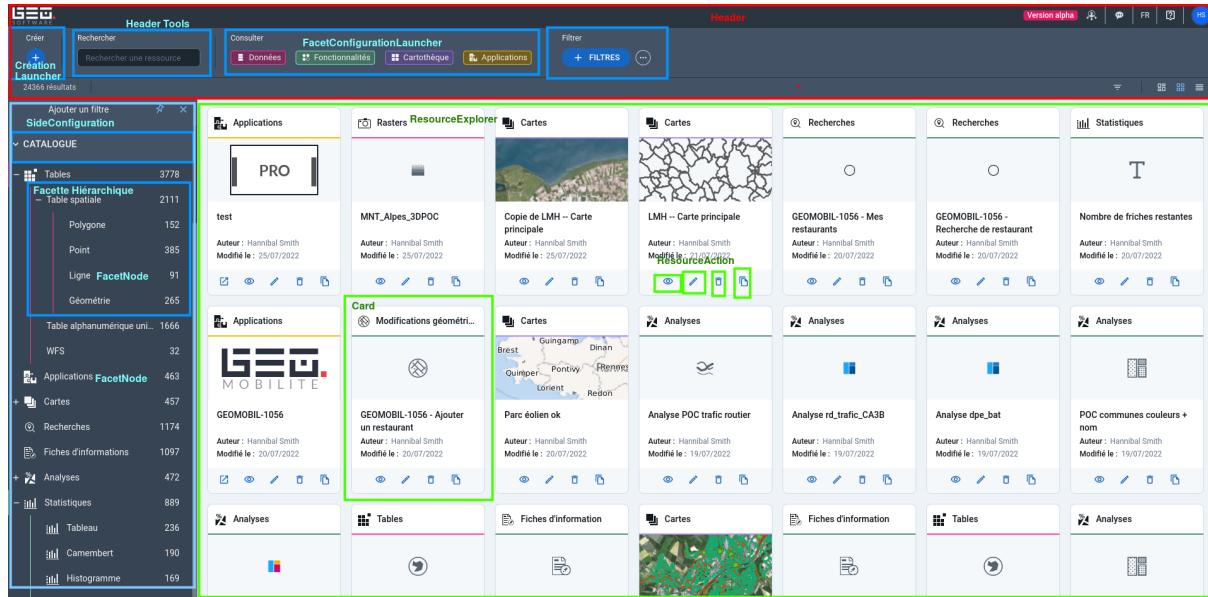
- Prototype de méthodes / attributs inexistant dans certaines interface

Lors de débogage, il est essentiel de faire un rapport d'activité en citant les modifications qui ont été tenté / celle qui n'ont pas été faites dans un fichier texte avec le nom des fichiers. Les modifications réalisées sont directement dans le fichier de différence du commit sur la branche dédié créer pour l'occasion.

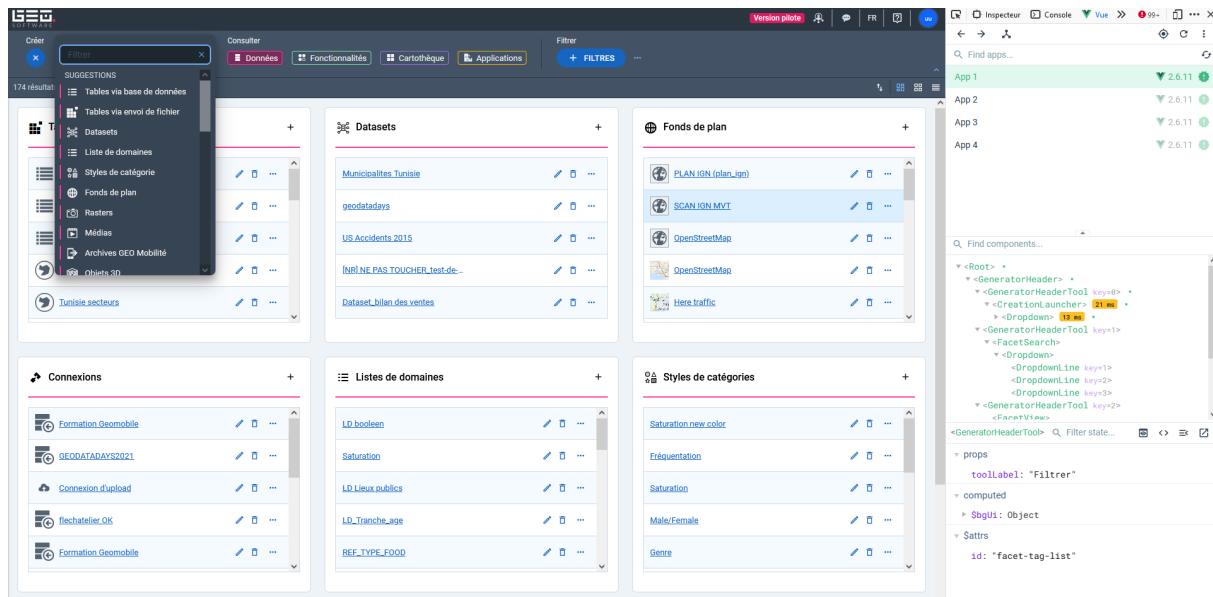
```
4  ↘ # HJ
5
6
7  ↘ ## Global
8
9    -> Pb extends BgVue (dcp je l'enlève ?)
10
11 ↘ ## FacetTagListComponent.vue
12
13   -> Impossible d'utiliser window (j'ai essayé de faire un const window
14     = Window (+ modification dans le tsconfig pour importer DOM pour TS
15     mais sans succès))
16
17
18
19 ↘ ## FacetTreeNodeComponent.vue
20
21   -> instance / méthodes / attributs qui n'existe pas
22
23   -> inst.node.displayChildren (qui existe) mais affecte le type bool à
24     node dans FacetTreeNodeComponent (à la place de inst.node.
25     displayedChildren)
26
27
28   ligne 10 : const facetHubService = args.facade.getFacetHubService();
29
30   la méthode getFacetHubService(); n'existe pas
31
32 ↘ ## GeneratorToggleSortComponent
33
34   item (type string) incompatible implicit ou force cast en FacetEnum
35
36   |
37
```

7.2.4 Entraînement de reproduction

Grâce aux ADR et à la documentation fournie sur le Gitlab de l'entreprise, j'arrive à comprendre l'interface et la structure de Facette :



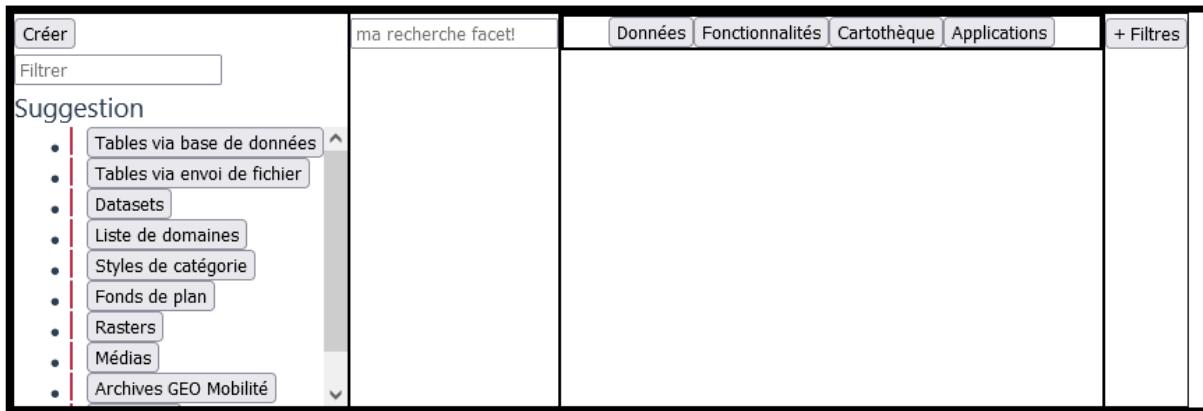
En plus de la documentation détaillant tout les composants et leurs aspects, je lance mon application GEO en local et j'applique le nouveau gabarit (Facette) pour visualiser la structure en direct grâce à une extension DOM de Vue



Pour m'entraîner à la reproduction de l'interface, je me renseigne sur le packeting Node js (création d'une librairie en javascript) dans le but de la réutiliser et de "l'isoler" pour mieux l'utiliser par la suite.

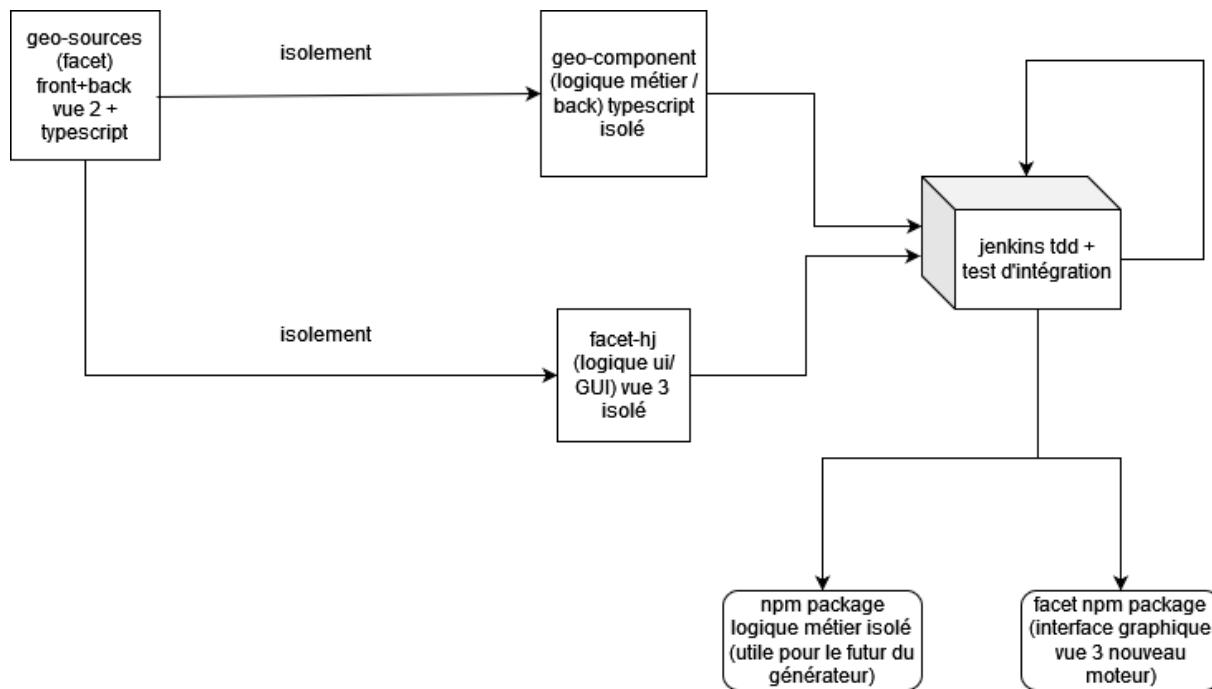
La partie fonctionnelle, logique métier étant déjà implémenté en TypeScript (langage javascript avec du stack de "type" par-dessus pour forcer les développeurs à typer leurs variables pour faciliter la rédaction et le débogage du workflow)

Ainsi j'arrive rapidement à recréer une pseudo-interface de facet pour pouvoir implémenter la logique métier à l'intérieur. J'utilise l'extension Vue 3 sur le navigateur pour me guider sur le fonctionnement de tel ou tel component.



7.2.5 Packeting Node JS NPM

Pour résumé l'objectif de mon stage en se basant sur le schéma ci-dessous : Facette est représenté par une interface graphique (logique UI = DOM + Style) et une logique métier. L'objectif est de séparer ces deux logique en paquet NPM distinct. Puis, par la suite de les déployer et les tester avec Jenkins (utilisation et explication du logiciel ici)



Pour me former au packaging NPM, je créais un projet à part `MigthyButton` pour créer un component simple : un simple bouton avec du css dans le but de transformer mon component en paquet Vue.

Puis en faisant diverse recherche sur Internet pour comprendre le fonctionnement et l'exportation de component Vue 3 en paquet NPM, je commence à modifier le fichier `main.ts`

```

▼ MightyButton.vue      TS main.ts      ●      TS vite.config.ts      {} package.json      {} tsconfig.app.json
src > TS main.ts > [●] default
1   import MightyButtonVue from "./components/MightyButton.vue";
2   import { type App, type Plugin } from "vue";
3
4   const plugin: Plugin = {
5     install: (app: App, options?: any) => {
6       app.component("MightyButton", MightyButtonVue);
7     },
8   };
9
10  export default plugin;

```

Grâce à la configuration du package.json et du tsconfig.json, j'ai pu exporter mon component à l'aide de `npm link` et j'ai pu l'importer dans un autre projet et l'utiliser.

```

▼ App.vue      X      ★ favicon.ico
src > ▼ App.vue > ...
1   <script setup lang="ts">
2   import HelloWorld from './components/HelloWorld.vue'
3   import TheWelcome from './components/TheWelcome.vue'
4   //const {MigthyButton} = require('mightybutton');
5   import MigthyButton from "../node_modules/mightybutton/src/components/MightyButton.vue"
6   </script>
7
8   <template>
9
10  <h1>où est le bouton ?</h1>
11
12
13  <MigthyButton
14    text="Oui je suis le test">
15
16  </MigthyButton>
17
18
19
20
21
22
23  </template>
24
25 > <style scoped> ...
52  </style>
53  |

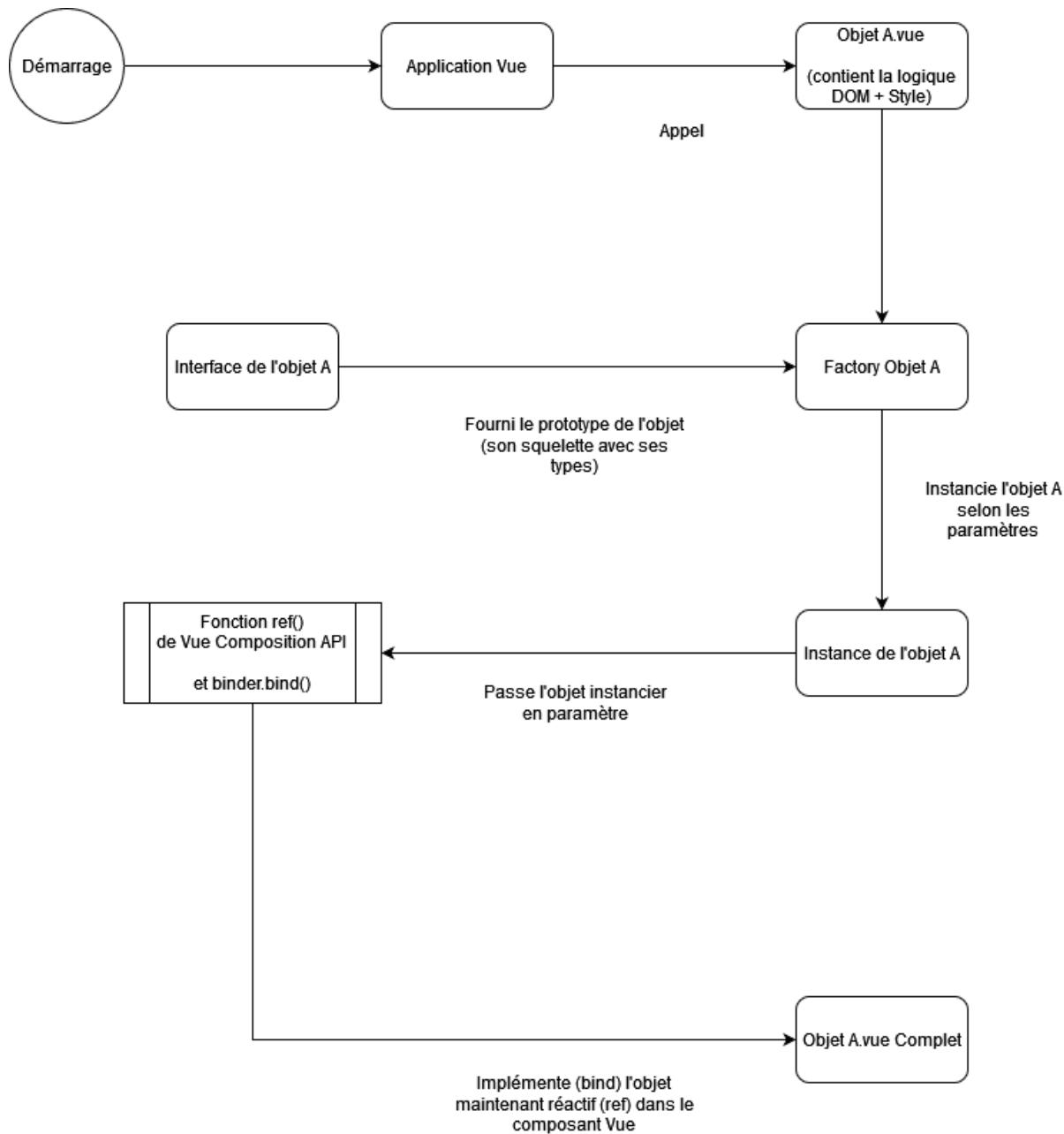
```

Une fois réussi, j'ai pu commencer à faire la même chose avec geo-component (dans la limite du possible avec les bugs que j'ai pu résoudre et les autres que je n'ai pas pu). En attendant une relecture pour fusionner avec la branche originelle (Merge-request, abrégé MR).

7.2.6 Ré-implémentations en Composition API

Pour comprendre les objectifs de la Composition API dans Vue 3, rendez-vous ici)

Pour pouvoir aider à la dissociation entre la **logique métier** (algorithme) et la logique UI (DOM + Style), on utilise la Composition API en utilisant la méthodologie suivante :



Ainsi un objet est séparer en trois fichiers distincts :

- Une **interface** qui contient les détails abstraits de l'objet (avec ces méthodes et attributs typés)
- Une **Factory** qui permet de faciliter l'instanciation de l'objet en se basant sur le prototype fourni par l'interface et d'associer un algorithme aux méthodes fournies (le tout très flexibles selon le contexte d'instanciation)
- Un **component Vue** qui contient la logique UI auquel on associera de manière réactive l'objet logique.

```

import type { HeaderTool } from '@bg/geo-types';

export interface GeneratorHeader {
  tools: HeaderTool[];
  isEdition: boolean;
  extractToolCssClasses: (tool: HeaderTool) => string;
  extractToolNameKey: (tool: HeaderTool) => string;
}

```

FIGURE 5 – Aperçu de l'interface (prototype) associé à l'objet Generator Header

```

import type { GeneratorHeader } from '../../types/generator/GeneratorHeader';
import type { HeaderTool } from '@bg/geo-types';
import type { ComponentFactoryArguments } from '@types/common/ComponentFactoryArguments';
import { CheckUtils } from '@bg/check-utils';

export function GeneratorHeaderFactory(params: ComponentFactoryArguments<void>): GeneratorHeader {
  CheckUtils.checkNotNull(params.store);
  return {
    extractToolCssClasses(tool: HeaderTool): string {
      if (Array.isArray(tool.classNames)) {
        return tool.classNames.join(' ');
      }
      if (typeof tool.classNames === 'function') {
        return tool.classNames(this.isEdition).join(' ');
      }
      return '';
    },
    extractToolNameKey(tool: HeaderTool): string {
      if (typeof tool.nameKey === 'string') {
        return tool.nameKey;
      }
      if (typeof tool.nameKey === 'function') {
        return tool.nameKey(this.isEdition);
      }
      return '';
    },
    get isEdition() {
      return params.store.get AppModule().isEditorInEdition;
    },
    get tools(): HeaderTool[] {
      return params.store.getHeaderToolModule().tools;
    }
  };
}

```

FIGURE 6 – Aperçu de la factory (instanciateur) associé à l'objet Generator Header

Ainsi désormais, mon objectif est de prendre le code écrit en Vue 2 où **un seul fichier centralise la logique métier et UI** et le dissocié en 3 fichiers pour la suite.

La création du fichier component et de l'interface reste simple car il n'y a que très peu de changements à effectuer (généralement de la syntaxe dans la partie template du component et éventuellement des fonctionnalités à re-implémenter qui n'ont pas été portés de Vue 2 vers Vue 3).

```

<template>
  <header class="generator-header">
    <GeneratorHeaderTool
      v-for="(item, index) in inst.tools"
      :key="index"
      class="generator-header__tool-component"
      :class="inst.extractToolCssClasses(item)"
      :toolLabel="inst.extractToolNameKey(item)"
      :id="item.htmlId"
    >
  </GeneratorHeaderTool>
</header>
</template>
<script lang="ts" setup>
import { GeneratorHeaderFactory } from '@/components/generator/GeneratorHeaderFactory';
import type { Ref } from 'vue';
import { inject, ref } from 'vue';
import type { GeneratorHeader } from '@/types/generator/GeneratorHeader';
import type { ComponentBinder } from '@/types/common/ComponentBinder';

let binder: ComponentBinder = inject('ComponentBinder');
const inst: Ref<GeneratorHeader> = ref(binder.bind<GeneratorHeader, void>(GeneratorHeaderFactory));
</script>

```

FIGURE 7 – Association réactive de l'objet logique `Generator Header` à son component graphique et visuel associé

Cependant, pour la création des **factory** associé aux objets, il peut y avoir une difficulté notamment lors de la traduction des **components instance** (lié à l'option API de Vue 2) qui sont des fonctionnalité lié à un fichier .vue et non dans notre cas à un fichier purement algorithmique en .ts.

Notamment plus précisement sur des **lifecycle hook / event / callback** particuliers et spécifique à Vue comme par Exemple **Vue.nextTick()**

nextTick()

A utility for waiting for the next DOM update flush.

- **Type**

```
function nextTick(callback?: () => void): Promise<void>
```

ts

- **Details**

When you mutate reactive state in Vue, the resulting DOM updates are not applied synchronously. Instead, Vue buffers them until the "next tick" to ensure that each component updates only once no matter how many state changes you have made.

`nextTick()` can be used immediately after a state change to wait for the DOM updates to complete. You can either pass a callback as an argument, or await the returned Promise.

Ce genre de problématique ne concerne qu'une poignée faible de fonctions dans les interfaces. Pour l'instant, je prioritise l'implémentation des autres factory pour progresser dans l'objectif.

Cependant je renseigne toujours les erreurs rencontrés dans un fichier TODO.md et je les explique en français pour quelle soit compréhensible pour moi et pour les autres développeurs sur le projet (les erreurs renvoyés par l'IDE ne sont pas toujours clair à comprendre lors de reprise de projet)

7.2.7 Problème d'héritage objet & d'alias

Au fur et à mesure des jours, j'arrive à régler quelques problèmes d'intégrations, cependant, quand j'arrive sur des components qui sont encapsulés dans d'autre components, je tombe sur la problématique de faire passer des propriétés d'un object A vers un object B (étant donné qu'on veut séparer la logique UI / logique métier) car on ne peut pas utiliser les *props* de Vue 3.

Finalement l'un des alternant présent dans l'entreprise trouve une solution en utilisant de la générericité (programmation générique) ce qui permet d'outrepasser le typage forcé en TypeScript.

Par la suite, pour pouvoir utiliser les autres objets créer dans le projet, il faut pouvoir les importer (pour connaître leurs prototypes) dans les autres fichiers. Cependant, pour faciliter le développement local avec Vite (serveur web local pour Vue ou React (un autre Framework Javascript)), il est possible de faire un système d'alias pour les imports (ce que les développeurs avant moi avait utilisé pour le projet GEO).

```
75 import Box from '@geo-src/components/ui/Box.vue';
76 import ResourceGroupedLayout from './ResourceGroupedLayout.vue';
77 import ResourceFormatTable from '../format/ResourceFormatTable.vue';
```

FIGURE 8 – Système d'alias à la ligne 75 caractériser par le @ au lieu du chemin relatif ligne 76 / 77

Cependant, ce système peut (pour une raison encore inconnu, sûrement une instance corrompu de ce service), ne pas fonctionner correctement et rendre la compilation du projet impossible :

```
Kollect stories start all
  Local: http://127.0.0.1:4080/
    Failed to load url to expose
Failed to load url @geo-src/modules/facets/store/modules/generator/facets/FacetConfigurationModule (resolved id: @geo-src/modules/facets/store/modules/generator/facets/FacetConfigurationModule). Does the file exist?
Failed to load url @geo-types (resolved id: @geo-types). Does the file exist?
Failed to load url @geo-src/modules/facets/store/modules/generator/facets/FacetResourceModule (resolved id: @geo-src/modules/facets/store/modules/generator/facets/FacetResourceModule). Does the file exist?
Failed to load url @geo-src/modules/facets/store/modules/utils/facetCategoryQueryBuilder (resolved id: @geo-src/modules/facets/store/modules/utils/facetCategoryQueryBuilder). Does the file exist?
Failed to load url @geo-src/modules/facets/store/modules/facetTreeConfigurationModule (resolved id: @geo-src/modules/facets/store/modules/FacetTreeConfigurationModule). Does the file exist?
Failed to load url @geo-src/modules/facets/stores/facetTreeQueryBuilder (resolved id: @geo-src/modules/facets/stores/facetTreeQueryBuilder). Does the file exist?
Failed to load url @geo-src/modules/facets/stores/facetSideQueryBuilder (resolved id: @geo-src/modules/facets/stores/facetSideQueryBuilder). Does the file exist?
Failed to load url @geo-types (resolved id: @geo-types). Does the file exist?
Failed to load url @types (resolved id: @types). Does the file exist?
Failed to load url @types/generator/common/InterfaceGeneratorModuleFactory (resolved id: @types/generator/common/InterfaceGeneratorModuleFactory). Does the file exist?
Failed to load url @types (resolved id: @types). Does the file exist?
Failed to load url @types (resolved id: @types). Does the file exist?
Failed to load url @types (resolved id: @types). Does the file exist? (x)
Failed to load url @types (resolved id: @types). Does the file exist? (x)
Failed to load url @modules/facets/store/modules/generator/facets/FacetResourceModule (resolved id: @modules/facets/store/modules/generator/facets/FacetResourceModule). Does the file exist?
Failed to load url @modules/facets/store/modules/generator/facets/FacetSearchModule (resolved id: @modules/facets/store/modules/generator/facets/FacetSearchModule). Does the file exist?
Failed to load url @types (resolved id: @types). Does the file exist?
Failed to load url @types (resolved id: @types). Does the file exist? (x)
Failed to load url @types (resolved id: @types). Does the file exist? (x)
Failed to load url @modules/facets/store/modules/generator/facets/FacetResourceModule (resolved id: @modules/facets/store/modules/generator/facets/FacetResourceModule). Does the file exist?
Failed to load url @modules/facets/store/modules/generator/facets/FacetSearchModule (resolved id: @modules/facets/store/modules/generator/facets/FacetSearchModule). Does the file exist?
Failed to load url @types (resolved id: @types). Does the file exist?
Failed to load url @types (resolved id: @types). Does the file exist?
Error: Failed to load url @types (resolved id: @types). Does the file exist?
at loadAndTransform (file:///C:/Users/hjacque/Desktop/Histoire/isolation-test/geocomponents/src/stories/App2.story.vue)
Collect stories end App2
Collect stories start App3
Error while collecting story C:/Users/hjacque/Desktop/Histoire/isolation-test/geocomponents/src/stories/App2.story.vue
@modules/facets/stores/facetTreeQueryBuilder (resolved id: @modules/facets/stores/facetTreeQueryBuilder). Does the file exist?
  at loadAndTransform (file:///C:/Users/hjacque/Desktop/Histoire/isolation-test/geocomponents/src/stories/App2.story.vue)
Collect stories end App3
Error while collecting story C:/Users/hjacque/Desktop/Histoire/isolation-test/geocomponents/src/stories/App2.story.vue
@types (resolved id: @types). Does the file exist?
  at loadAndTransform (file:///C:/Users/hjacque/Desktop/Histoire/isolation-test/geocomponents/src/stories/App2.story.vue)
```

FIGURE 9 – Console de debug avec les messages d'erreurs expliquant l'impossibilité de trouver les fichiers à cause des erreurs d'alias : "Failed to load url <...> Does this file exist?"

Ces erreurs ont été résolus en mettant le chemin d'accès direct.

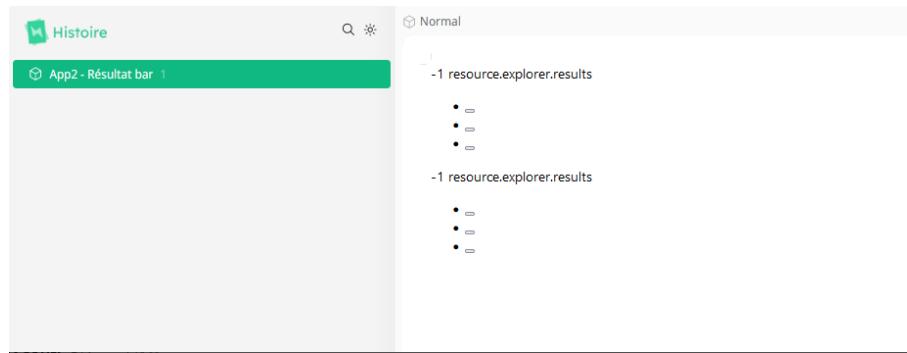
7.2.8 Isolation et TDD des applications Vue dans Histoire

Pour pouvoir continuer le développement et l'implémentation de l'application, on passe à la phase de TDD en testant les sous-applications de Facette dans le framework Vue "Histoire".

Pour pouvoir faire fonctionner ces applications, il faut pouvoir "émuler" des données en local. Ces données sont initialement fourni par le store et la facade qui représentent des services de l'application GEO.

Pour pouvoir émuler le store et la facade, il faut recréer les objets associés à ces services en respectant les types de données qu'ils reçoivent et qu'ils renvoient".

Au fur et à mesure de la re-implémentation des objets, un résultat commence à se distinguer graphiquement :



Cependant, après l'implémentation d'un service utilisant les énumérations javascript, une erreur lié au compilateur empêche la compilation et la progression du projet. Aucune solution n'a été trouvé pour résoudre ce problème étant donné le manque de connaissance sur Internet de ce problème.

7.3 TDD : Debug Test Unitaire sur Elastic Search & Lucène | Java Maven

A deux semaines de la fin de mon stage, étant donné que la mission sur Vue à était pousser à son maximum et qu'il faut attendre les travaux d'autres développeurs pour pouvoir progresser sur la montée de version complète (notamment dans les détails les plus technique liés aux langages et à l'architecture de GEO), on m'affecte une nouvelle mission : débugger les tests unitaires qui permettait de tester le bon fonctionnement de Lucène et ElasticSearch sur aigle-5 core ou "AAS" (Aigle Application Server) (le côté back, logiciel, l'infrastructure programmée en Java avec le framework Spring).

7.3.1 Qu'est-ce que ElasticSearch & Lucène

Lucène est une bibliothèque open source écrite en Java qui fournit des **fonctionnalités de recherche textuelle**. Elle offre des **capacités d'indexation et de recherche avancées**, permettant de construire des systèmes de recherche personnalisés. Lucène est souvent utilisée comme base pour d'autres outils de recherche, notamment Elastic Search.

Elastic Search, quant à lui, est une **solution de recherche et d'analyse distribuée**, construite au-dessus de Lucène. Il fournit une interface RESTful qui permet aux développeurs **d'interagir avec les données de manière simple et efficace**. Elastic Search est conçu pour être hautement évolutif et offre des fonctionnalités telles que **la recherche en texte intégral (fulltext)**, **l'agrégation de données**, **la recherche géo-spatiale** et **la recherche en temps réel**.

L'implémentation et le maintien d'Elastic Search et Lucène est très important et essentiel pour GEO car l'application fonctionne à l'aide de données geo-spatial, la recherche par annuaire inversé permet de trouver et indexer par catégorie selon des champs lié aux données beaucoup plus efficacement et rapidement que la recherche classique en SQL.

7.3.2 TDD & Test Unitaire : Procédure

Pour pouvoir tester la fiabilité du code qu'on a produit, on le passe dans des **tests unitaires** en réalisant donc du **TDD (Test Driven Development)**, cette pratique est essentiel dans n'importe quelle entreprise qui développe, que ce soit des outils internes à l'entreprise et surtout quand ce sont des solutions pour les clients. Tester le comportement des fonctions et des fonctionnalités de la solution sur plusieurs axes c'est assuré son fonctionnement et ces limites d'utilisation tout en garantissant l'expérience utilisateur des clients qui utiliseront le produit.

Sur le repo git de AAS, se trouve 3 fichiers de tests unitaires pour Lucène et Elastic Search : Pour réaliser du TDD, il faut (de préférence) utiliser un IDE qui propose des fonctionnalité de débogage et permettre l'analyse ligne par ligne de la compilation du programme pour voir les variables et leurs valeurs en temps réels : une avancée *step by step*.

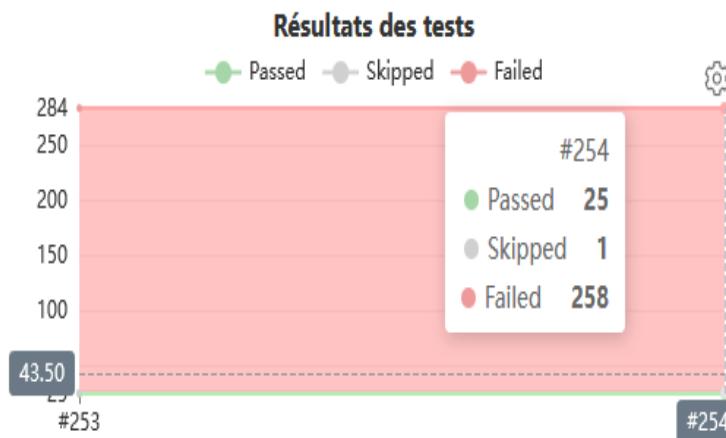


FIGURE 10 – Aperçu du résultat des test avant mon arrivé

Le TDD est une activité où il faut être patient, analyser et comprendre l'environnement qu'on essaie de faire fonctionner, comprendre ce qu'on veut tester et comment on le test et jusqu'à où. Quelle est la raison du dysfonctionnement / de l'erreur ?

Les premiers jours de TDD sur Lucène se sont résumé à de l'exploration de code, beaucoup de step by step pour identifier le processus de recherche utilisé et comment était-il organisé (dans notre cas la recherche à facette). Comme dans le projet Vue 'Facette' (interface du générateur GEO), chaque étape est décomposé en classe et chaque classe s'encapsule dans une autre, chaque classe n'ayant qu'un but pour le S du principe SOLID (Single Responsibility).

Puis au fur et à mesure de la compréhension du code, du fonctionnement de l'architecture et des classes. Je commence à voir et à comprendre comment fonctionne "actuellement" le code et comment il "devrait" fonctionner. J'arrive à régler les premières erreurs qui sont liées à un paramétrage manquant dans la construction d'une recherche avec Lucène :

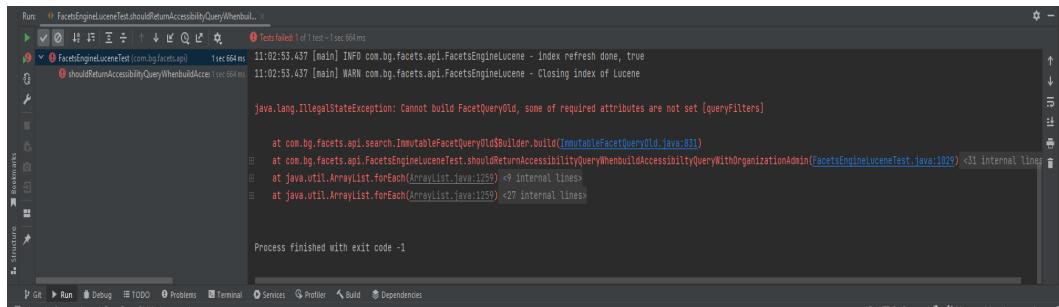


FIGURE 11 – Test unitaire qui échoue avec le message d'erreur à propos d'un paramètre manquant

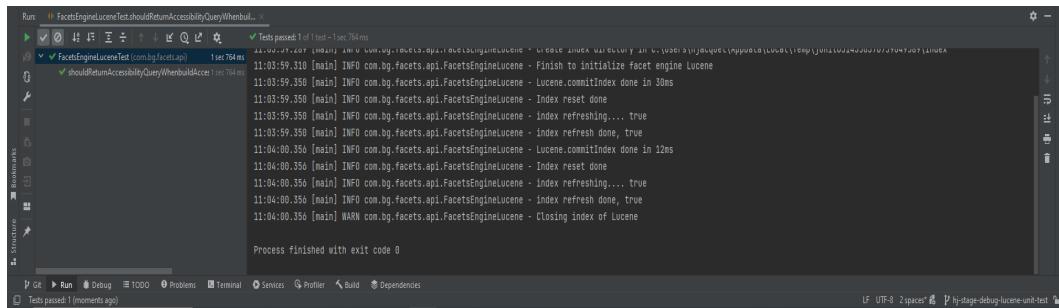


FIGURE 12 – Le même test unitaire cette fois qui est validé

Sur plusieurs fichiers, des problèmes similaires à celui de `queryfacetResult` mais avec des particularités plus techniques avec des fausses données (mocked data) (pour simuler les données sans être dans un vrai environnement et sans prendre de véritables données) : Il ne faut pas en créer énormément sinon ça ralentit le test et souvent le bloque. Il faut aussi faire des vérifications sur ce que fait la fonction véritablement (en regardant le nom du Test qui est volontairement long pour expliquer ce qu'on teste précisément) (utilisateurs autorisés à faire une requête ?, appartient t'il à une organisation x ?, etc.).

```

LOG.info("Filtered resources size: " + filteredResources.size());
int limit = fakeResources.size();

// WHEN
final GlobalQueryFilters globalQueryFilters =
    GlobalQueryFilters.builder().user(user).build();

ImmutableFacetQueryOld build =
    FacetQueryOld
        .builder()
        .user(user)
        .offset(DEFAULT_OFFSET)
        .limit(limit)
        .queryFilters(globalQueryFilters)
    ].build();
FacetQueryResultOld search = engine.search(build);

// THEN
assertThat(search.resources()).hasSize(filteredResources.size());

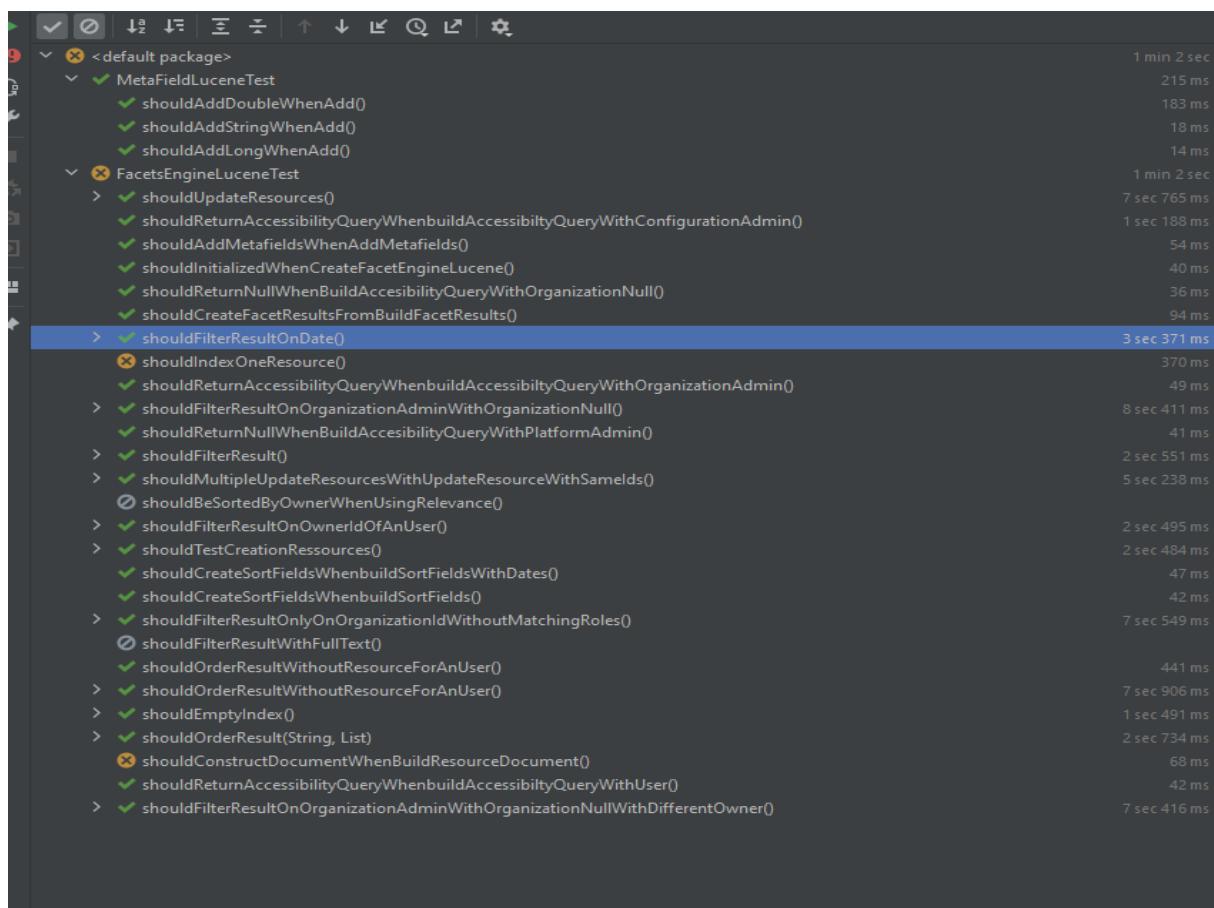
assertThat(search.resources()) ListAssert<ResourceResult>

```

FIGURE 13 – Exemple de requête Lucène avec le système de filtrage appliquée

Après avoir réglé le maximum de test que j'ai pu comprendre (certains test étant très complexe / demandant beaucoup de temps / de documentation sur Lucène), je décide de commit et push mon code sur ma branche git et de créer une merge request (demande de fusion).

Aperçu du dernier lancement des tests unitaire avant de pousser les modifications sur le repo git distant :



7.3.3 Merge Request : Review et Sonarqube

Pour finaliser mes modifications, il faut une revue de mes modifications pour quelle puisse 'fusionner' (s'intégrer) dans la base de code principale déjà existante (la branche main du git) là où est le code est utilisé par tous.

The screenshot shows a GitLab merge request page. At the top, it displays the path: GEO RD > geo > aigle5-core > Merge requests > !1685. The merge request is labeled 'Open' and was created 25 minutes ago by 'Hugo JACQUEL' (Developer). Action buttons include 'Edit', 'Mark as draft', and a dropdown menu. The title of the merge request is 'FacetsEngineLuceneTest : Unit Test almost 100% GOOD'. Below the title, there are navigation links for 'Overview' (7), 'Commits' (13), and 'Changes' (8). A summary box indicates '7 unresolved threads' with icons for comments and file attachments. The main content area includes sections for 'Analyse de code' (Code Analysis) and 'Indicateurs bloquants' (Blocking Indicators). Under 'Analyse de code', it says 'Barrière Qualité : OK' with sub-points for 'Bugs, bloquants: 0, critiques: 0' and 'Vulnérabilités, bloquantes: 0, critiques: 0'. Under 'Indicateurs supplémentaires' (Additional Indicators), it lists 'Bugs: 0, Mauvaises pratiques: 13, Dette Technique: 2 h' and 'Vulnérabilités: 0'. Below this, a note states: 'MR pour le fix des UT Lucene sur AAS :) 2/3 test compliqué à résoudre à cause des messages d'erreurs incompréhensibles et d'autres pb. Mais le reste OK 100%' and 'BOUND 500 -> 50'.

FIGURE 14 – Aperçu de la merge request sur Gitlab avec l'indicateur automatique de qualité de code

Pour que la demande de fusion soit accepté, elle doit être relu par une personne de la R&D pour qu'elle puisse rigoureusement regarder si :

- le code poussé correspond bien au nom de la merge request / problématique du ticket Jira
- Si il n'y a pas de bug, d'interrogation sur une pratique particulière
- des erreurs de formatage particuliers / mauvaises pratiques en commentaires ou dans le code de manière général

Ainsi, la personne effectue pour chaque problème à rapporter un "mini thread" (petit channel de discussion) où il reprend la parti du code où est l'erreur et explique qu'est-ce qu'il ne va pas :

started a thread on the diff 10 minutes ago ^ Hide thread

lib/facets/facets-engine-lucene/src/main/java/com/bg/facets/api/FacetsEngineLucene.java

```

250 250    }
251 251
252 252    @Override
253 - public void updateResources(Set<FacetResource> facetResources) throws FacetIndexException {
253 + public void updateResources(Set<FacetResource> facetResources) throws FacetIndexException, FacetSearchException {
254 + // Rajout Commit Index Pour solve la persistance des données

```

· 10 minutes ago Owner (1) (1) ⋮

commentaire ou TODO ? en anglais, jamais français, pour nous - et peut-être plutôt au-dessus de la méthode (donc du @Override)

Edited by [REDACTED]

Reply... Resolve thread Diff

started a thread on the diff 8 minutes ago ^ Hide thread

lib/facets/facets-engine-lucene/src/test/java/com/bg/facets/api/FacetsEngineLuceneIndexesTest.java

```

67 67        engine.close();
68 68    }
69 69
70 - @Test
70 + // @Test

```

· 8 minutes ago Owner (1) (1) ⋮

Laisser @Test mais l'ignorer, avec @Ignore ou @Disabled ... suivant la version de junit ici

Reply... Resolve thread Diff

FIGURE 15 – Commentaire de qualité du code sur ma merge request par une personne de la R&D

Il y a aussi un robot automatisé qui passe par dessus le code proposé pour regarder les potentiels erreurs et comment l'optimiser pour le rendre meilleur : **SonarQube**.

SonarQube est une **plateforme d'analyse statique du code** utilisée pour évaluer la **qualité du code source**. Elle identifie les **erreurs** de programmation, les **vulnérabilités de sécurité** et les **problèmes de performance**. SonarQube fournit des rapports détaillés qui aident les développeurs à comprendre les problèmes et à les corriger.

L'intégration de SonarQube avec un repo GitLab présente plusieurs avantages. Elle permet d'améliorer la qualité du code en identifiant les erreurs et en encourageant l'écriture d'un code propre. SonarQube détecte également les vulnérabilités de sécurité et aide à renforcer la sécurité du code. Il aide aussi à prévenir les problèmes de performance en identifiant les inefficacités. SonarQube permet de suivre la dette technique et peut être intégré dans un processus d'intégration continue pour une analyse régulière du code.

En utilisant SonarQube avec GitLab, les développeurs peuvent améliorer la fiabilité, la sécurité et l'efficacité de leur développement logiciel. SonarQube s'active automatiquement lors d'une merge request et permet d'avoir un premier retour automatique sur la qualité de son code en moins de 2min.

The screenshot shows the SonarQube interface with the following details:

- Project Information:** tmp.merge-request:730:1685, master branch.
- Last analysis:** had 2 warnings on June 30, 2023, at 16:34.
- Code Smells:**
 - Line 445: Rename this local variable to match the regular expression `'[a-z][a-zA-Z0-9]*\$'` (Mauvaise pratique).
 - Line 451: Remove this unused "resourcesSize" local variable. (Mauvaise pratique).
 - Line 455: Rename this local variable to match the regular expression `'[a-z][a-zA-Z0-9]*\$'` (Mauvaise pratique).
 - Line 461: This block of commented-out lines of code should be removed. (Mauvaise pratique).
 - Line 467: Remove this useless assignment to local variable "user". (Mauvaise pratique).
 - Line 473: Remove this unused "user" local variable. (Mauvaise pratique).
 - Line 479: Remove this useless assignment to local variable "ResSearchNb". (Mauvaise pratique).
 - Line 485: Rename this local variable to match the regular expression `'[a-z][a-zA-Z0-9]*\$'` (Mauvaise pratique).
 - Line 491: Remove this unused "ResSearchNb" local variable. (Mauvaise pratique).
- Unused Code:**
 - Line 451: int Nbdocs = engine.getNumDocs();
 - Line 455: assertThat(Nbdocs).isEqualTo(10);
 - Line 461: Nbdocs = engine.getNumDocs();
 - Line 467: List<String> resources = fakeResources.stream().map(r -> r.getId().toString()).collect(toList());
 - Line 473: Nbdocs = engine.getNumDocs();
 - Line 479: int resourcesSize = resources.size();
 - Line 485: assertThat(engine.getNumDocs()).isEqualTo(resources.size());
 - Line 491: searcherManager = (SearcherManager) FieldUtils.readDeclaredField(engine, "searcherManager", true);
- Annotations:** The code is annotated with various SonarQube icons (e.g., warning, info, fixme) and severity levels (Mineur, Ouvr^et, Non affecté).

FIGURE 16 – SonarQube révélant des pratiques de mauvais code sur le code que j'ai pu faire notamment sur des variables inutilisées

Après plusieurs review en utilisant un formatage bien spécial de l'entreprise pour garantir une bonne lisibilité du code, ma branche est enfin intégré à master (projet principal) :

Name	Last commit	Last update
doc	chore: add git-blame-ignore-revs	6 days ago
lib	Merge branch 'hj-stage-debug-lucene-unit-t...'	9 minutes ago
scripts	fix #10126 (with previous)	5 years ago
test	Merge branch 'master' into feature/GEO-181...	3 months ago
webapp	better healthcheck	1 month ago
.editorconfig	feat: GEO-14829 - .properties files in java 8 ...	1 year ago
.git-blame-ignore-revs	chore: add git-blame-ignore-revs	6 days ago
.gitignore	Set artifact version to 2.4.0-SNAPSHOT (GEO...	1 year ago

FIGURE 17 – Aperçu de aigle-core avec comme dernier commit , la fusion avec ma branche

C'est sur cette **merge-request** que ce fini mon stage au sein de CIRIL Group.

8 Méthodologie Agile et organisation de l'entreprise

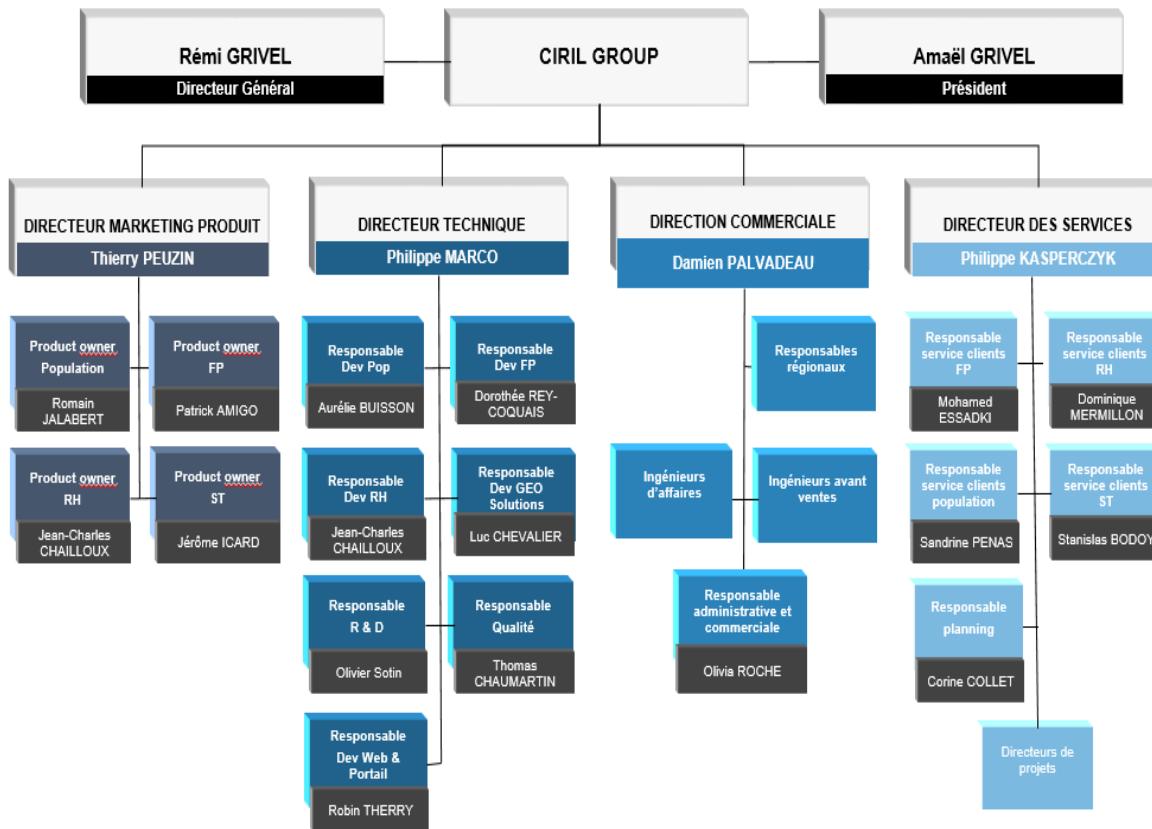


FIGURE 18 – Organigramme de l'entreprise Ciril GROUP

8.1 Définition et présentation de la méthodologie

La méthodologie agile vise à **améliorer la flexibilité et l'efficacité** du processus de développement de logiciels au sein de Ciril GROUP, tout en répondant aux **besoins et attentes des clients**. Elle favorise une approche **itérative et collaborative**, en mettant l'accent sur la livraison régulière de fonctionnalités fonctionnelles et de valeur métier (Sprint).

L'objectif principal de l'agilité est **d'assurer une plus grande réactivité aux changements et aux retours d'expérience**. Au lieu d'un plan rigide et figé, les équipes travaillent par incrément de temps plus courts (4 semaines chez Ciril GROUP). Cela permet une adaptation plus rapide aux évolutions des besoins des clients et du marché.

La méthodologie inclue un rôle essentiel : le Product Owner, qui représente les **intérêts des clients** et définit les **priorités du projet**, ainsi que l'équipe de développement qui est responsable de la réalisation des fonctionnalités. La communication constante et la collaboration étroite entre ces parties prenantes sont essentielles pour **s'assurer que les objectifs du projet sont atteints**.

On utilise alors **la backlog**, qui est une liste priorisée des fonctionnalités à développer, les user stories, qui décrivent les besoins et les cas d'utilisation des utilisateurs, et les réunions régulières de planification, de révision et de rétrospective (chez Ciril GROUP, les Standup et les points Synchro). Ces pratiques aident à maintenir une transparence et une communication efficace au sein de l'équipe.



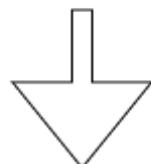
Les clients GEO ont des discussions avec les Product Owner sur de nouvelles fonctionnalités à intégrer à GEO (ou un font un rapport sur un bug)



Le Product Owner annonce les nouvelles fonctionnalités à implémenter / bug à corriger au Manager de l'équipe, il annonce leurs priorités et l'importance de x objectifs



Le Manager d'équipe prend en compte la demande du PO, et regarde dans son effectif d'équipe qui sont les personnes qui ont du temps / facilités avec les sujets discuté :
" Je peux mettre 2 personnes sur cette fonctionnalité pendant 1 semaine"



	Création du ticket Jira et administration de la tâche / objectif à des membres de équipes selon plusieurs facteurs	
--	--	--

FIGURE 19 – Exemple d'utilisation de la méthode Agile au sein de l'entreprise

8.2 Outils utilisés

8.2.1 Gestion et stockage du code avec Gitlab

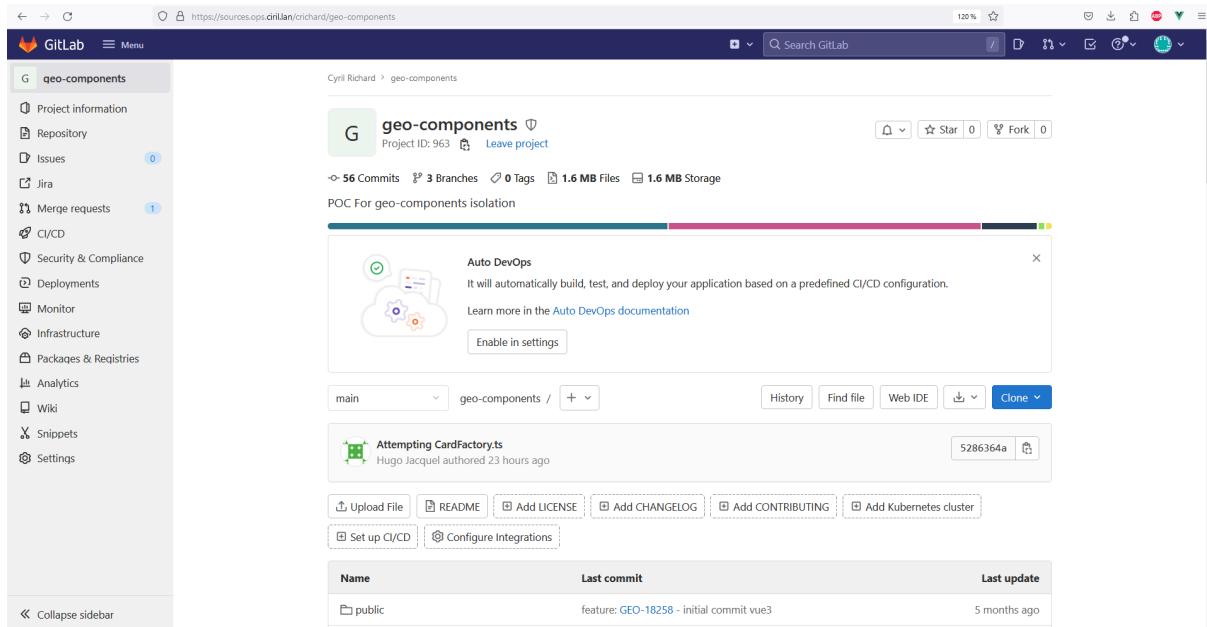


FIGURE 20 – Aperçu de la plateforme Gitlab de l’entreprise sur le repo geo-component

GitLab est une plateforme de **gestion de dépôts Git** qui offre de nombreux avantages en termes de collaboration et de contrôle de version. Ciril GROUP utilisent GitLab pour héberger leurs repos Git, qui contiennent le code source de leurs projets. Grâce à GitLab, les équipes de développement peuvent travailler simultanément sur des branches séparées, ce qui facilite la gestion des différentes fonctionnalités, correctifs ou améliorations en parallèle.

Les branches permettent de travailler de manière isolée sur des modifications spécifiques sans impacter le code principal. Lorsque les modifications sont terminées, les équipes créent des Merge Requests (demandes de fusion) pour intégrer leurs modifications dans la branche principale. Cette approche favorise la collaboration et permet aux équipes de passer en revue et de valider les modifications avant de les fusionner.

Le système de versionning intégré à GitLab permet de garder une trace précise de toutes les modifications apportées au code. Chaque commit est enregistré et peut être référencé à tout moment pour revenir à une version antérieure du code si nécessaire. Cela offre un historique complet des changements et facilite la détection et la résolution des problèmes.

Pour conclure, l'utilisation de GitLab est essentielle pour Ciril GROUP, car elle favorise une gestion efficace du code source, une collaboration fluide entre les équipes de développement, et offre une traçabilité et un contrôle de version complet. Elle permet d'optimiser le processus de développement, de réduire les conflits de code, d'améliorer la qualité du logiciel et de faciliter le déploiement continu.

The screenshot shows a GitHub commit history for the 'geo-components' branch. The commits are listed in chronological order from June 2023:

- 26 Jun, 2023 3 commits
 - mapping all alias good | error FacetTreeConfigurationModule '_ROOT_ properties on undefined object
Hugo Jacquel authored 1 day ago
 - Appmodule OK
Hugo Jacquel authored 1 day ago
 - Starting Mock GeneratorStore & GeneratorFacade w/ Service mapping > appmodule & facetResource
Hugo Jacquel authored 1 day ago
- 21 Jun, 2023 3 commits
 - App 2 working | Pire invention le CSS
Hugo Jacquel authored 6 days ago
 - Correction FacetTreeNode & Tentative args ResourceFormat
Hugo Jacquel authored 6 days ago
 - Correction ResourceFormatCardComponent
Hugo Jacquel authored 6 days ago
- 20 Jun, 2023 3 commits
 - Correction en cours | Update du TODO
Hugo Jacquel authored 1 week ago
 - Update FacetNode Interface prototype
Hugo Jacquel authored 1 week ago
 - FacetNode 100%
Hugo Jacquel authored 1 week ago
- 19 Jun, 2023 16 commits
 - Correction FacetTagList
Hugo Jacquel authored 1 week ago
 - DropdownLine OK
Hugo Jacquel authored 1 week ago
 - Dropdown OK
Hugo Jacquel authored 1 week ago
 - Issue with ResourceAction ?
Hugo Jacquel authored 1 week ago
 - layout/ sub-dir finish
Hugo Jacquel authored 1 week ago

FIGURE 21 – Commit GIT effectué sur une de mes branches de développement sur geo-component

8.2.2 Organisation Agile & Kanban avec Jira

Pour la gestion du projet GEO, l'entreprise Ciril Group utilise Jira : une interface web de gestion de projet : utilisé et essentiellement pour la gestion de ticket principalement par le service Marketing et celui du Service Client (qui sont proche des clients de Ciril Group et mettent à jour l'infrastructure avec de nouvelles fonctionnalités / modifications à apporter à GEO et les autres applications)

GEO / GEO-18256

Migration Vue 2 -> Vue 3 (GEO Generateur)

Modifier Commentaire Attribuer Suite >> Fermée >> En test

Work Time Calendar Timer
0m

Informations

Type:	Epopée	État:	A PLANIFIER
Priorité:	Medium	(Afficher le workflow)	
Affecte la/les version(s):	Aucune	Résolution:	Non résolu
Composants:	Aucune	FixVersion:	Aucune
Étiquettes:	Aucune		
Epic Name:	Migration Vue 2 -> Vue 3 (GEO Generateur)		
Module:	GEO-Generateur		

Chaque ticket correspond à des actions / des modifications à faire sur le code et aussi un taux de priorité (par rapport à la notion de facteur de risque) :

Amélioration

- Bug
- Epopée
- Initiative
- Récit
- Tâche
- Assistance

Priorité* **Medium**

Highest
High
Low
Lowest

FIGURE 22 – Type de ticket sur Jira

FIGURE 23 – Niveau de priorité pour les tickets Jira

Chaque ticket Jira est créé grâce à :

- une demande client (ajout de fonctionnalité, amélioration)
- un bug report (réparation, issue)
- changement majeur de stack ou d'organisation / refonte de code intense

Les créateurs des tickets Jira peuvent être :

- des PO (Product Owner) : après retour client ou de tests du produit par eux-même (sur des fonctionnalités précises car les PO décident des fonctionnalités à intégrer dans X version de GEO)
- des membres de Ciril group : en implémentant de nouvelles fonctionnalités, tentative de build, changement de code : l'importance très présente de la notion de test de non-régression.

Sur chaque ticket Jira on trouve plusieurs informations :

- Le nom et la description associés
- les étapes de reproduction pour arriver au bug
- des Screenshots édités pour illustrer (notamment des présents sur des problématiques d'IHM)
- Date de création, mis à jour
- Le nom de la personne rapporteuse / créatrice
- Le nom de la personne responsable (à qui a été attribué le ticket)
- Des liens en rapport avec les tickets (documentation, adr, mention gitlab pour les fix de bug et les MR)

Exemple de ticket Jira sur GEO :

GEO / GEO-10243

Rajouter l'action "Vue détaillée" sur les composants carte/analyse dans les tableaux de bord

Workflow

Modifier Commentaire Attribuer Suite >> Fermée >> En test Workflow Exporter

Work Time Calendar Timer
0m ✓ || X

Informations

Type: Amélioration État: A PLANIFIER (Afficher le workflow)

Priorité: Medium Résolution: Non résolu

Affecte la/les version(s): Aucune FixVersion: GEO 2.6.0

Composants: Aucune

Étiquettes: GEOKey engagementClient ressourceCarte ressourceTableauxDeBords

Informations principales Estimations et suivi Références Externes

Versions cibles: GEO 2.6.0
Epic Link: geo-key-rgaa
Module: GEO-Key
Blocage facturation projet: Non
Partage club utilisateur: Non

Dates

Création: 05/09/2018 13:50
Mise à jour: Il y a 1 semaine

Ouvrir dans l'Intranet +

Dupliques potentiels

- GEO-192
- GEO-193
- GEO-190
- GEO-188
- GEO-180
- GEO-186
- GEO-178
- GEO-168

Par la suite, le ticket Jira peut avoir plusieurs statuts :

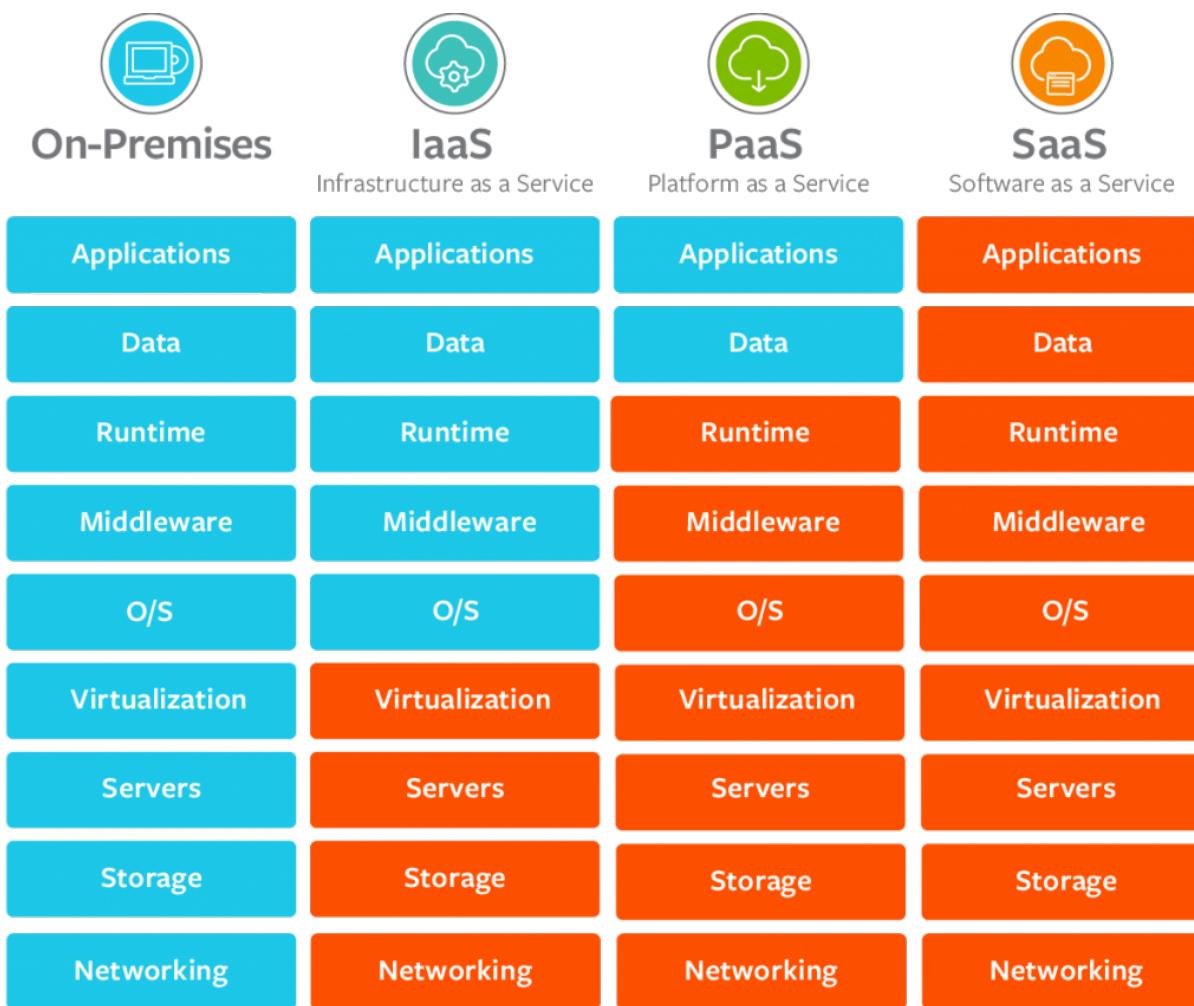
- Ouvert à la résolution
 - En attente d'information
 - Étude technique / revue
 - En test
 - à planifier / attribuer
 - Fermé (résolu)
 - Ré-ouvert
- > **Les réouvertures** peuvent survenir lors de réaction de **test de non régression**.
- > Un ticket **n'est pas "abandonner"**, dans le cas où un ticket n'est pas résolu dans le sprint/backlog : il sera **ré-attribuer aux prochain sprint** (ou pas sous décisions des product owner)

8.2.3 CI/CD : Organisation automatisé des tests de déploiement et unitaire avec Jenkins

À l'occasion de la création de ma paquet NPM pour geo-types et geo-components, on utilise le logiciel Jenkins qui permet du traitement du code via des tests unitaires, tests de déploiement selon certains paramètres le tout automatisé.

Sur Jenkins on travaille sur plusieurs versions :

- La Dev : Machine Virtuel classique
- La "Enviro" : Machine adaptée hors VM (docker)
- SaaS : Version de GEO hébergé chez Ciril Group (le client n'utilise que le service proposé)
- On premise : Version de GEO installé chez le client (self-hosted) (similaire à une IaaS)



You Manage Other Manages

8.3 Système de version avec GEO

Lorsque je suis arrivé dans l'entreprise pour mon stage (le 1er juin 2023), l'application GEO est passée en version 2.4.2.

La gestion / le système de versionning utilisé pour GEO est le suivant :

Versionning GEO

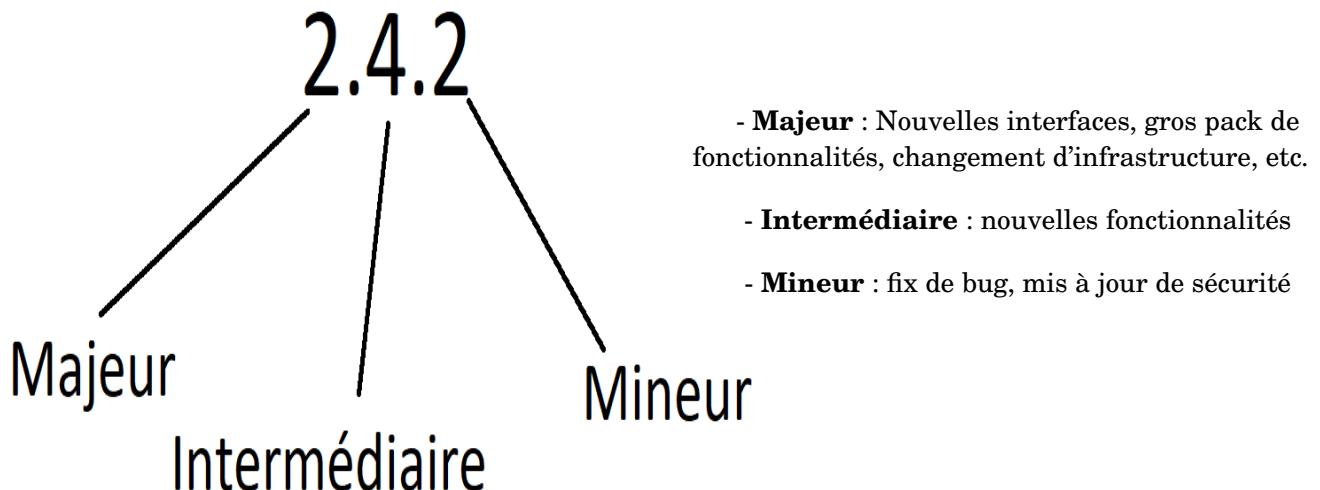


FIGURE 24 – Système de versionning GEO

8.4 Implémentation dans l'entreprise

8.4.1 Point Synchro & Stand up

Pour la résolution de ticket, les managers utilisent des mesures pour caractériser les capacités techniques de leurs équipes et pouvoir optimiser leurs moyens et leurs forces :

- La méthodologie "verticale" : assigné tout au long du projet à une poignée de personnes des tâches bien précise sur des thèmes bien spécifiques : Se concentrer uniquement sur la partie 3D, sur une certaine partie
- La méthodologie "horizontale" : assigné à tout le monde des tâches diverses et similaires pour fluidifier l'organisation et la rédaction (compétences techniques vs compétences sociales)

En général, on utilise les deux méthodologies en même temps car il est essentiel que la répartition des tâches soit équitable mais dans le cadre de projet très technique, il peut être judicieux d'attribuer une grande tâche par personne pour qu'elle puisse pleinement se concentrer dessus.

Chaque jour, des réunions "stand up" (Daily Scrum/Agile) sont réalisé pour faire un rapport quotidien :

- Progression de chacun dans l'objectif de ses tâches
- Réorganisation de la répartition des tâches selon les besoins
- Exprimer les difficultés rencontrées sur des tickets pour que tout le monde en ait connaissance

Ainsi ces réunions quotidiennes permettent de suivre l'évolution du Gantt (Diagramme de Gantt utilisé en gestion de projet pour visualiser l'évolution des tâches à travers des graphes et selon le temps, intégré à la plateforme Jira)

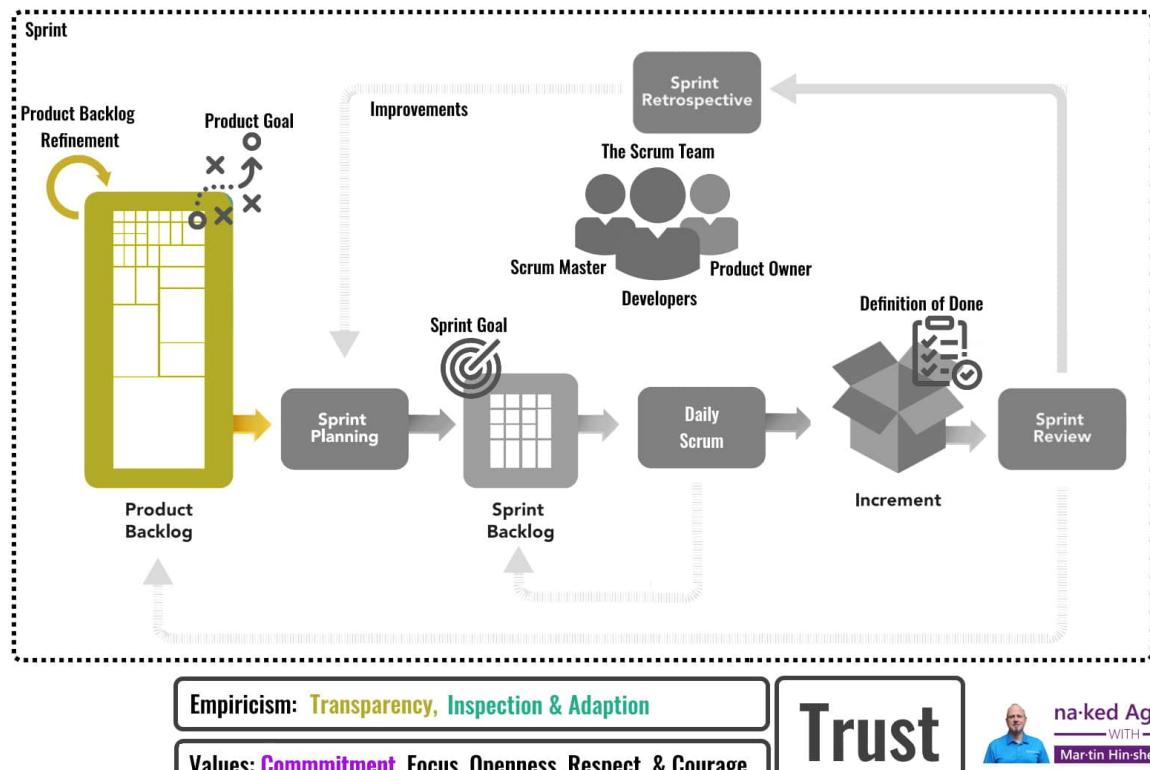
Tout au long du sprint, on réalise des réunions plus poussé nommé backlog grooming qui passe en revue au peigne fin (donc plus long) les éléments du sprint pour tenir à jour et pour permettre une bonne synergie.

Une fois la période de Sprint fini, le Sprint review est une grande réunion où chaque membre est réuni et on peut voir les résultats du Sprint réalisé (1 mois).

La fin d'un Sprint conclu généralement sur une nouvelle version de GEO et ses modules (le générateur, le gabarit, fonctionnalités) sur :

- des nouvelles fonctionnalités
- des bugs

SCRUM FRAMEWORK



L'objectif du point synchro est plus orienté "marketing" : powerpoint présentation des performances de l'équipe sur les objectifs fixés par la roadmap (et les sprint).

La présentation présente différentes statistiques via un tableau (**Macro Plan** sur trois quadri-mestres avec engagements dégressifs) :

- Le thème dans lequel est placé le ticket (Innovation, Outils, Métiers, Sécurité & Dette)

- La description du ticket (mis à jour infra, migration stack vue, demande client stylo focus, stabilisation de tel version, backport x fonctionnalité du y version)

- Statut (+, =, flèche qui monte, flèche qui descend)

- Période : le nombre de jour contribué réellement pour une tâche / un ticket (fini ou pas)

- Roadmap : le nombre de jours / l'avancée initialement prévue dans la roadmap / pour le sprint.

Macro Plan - R&D Socle GEO

Theme	Description	Quadrimestre			Roadmap		
		Q2 - fin Aout	Q3 - fin Décembre	Q1 - fin Avril	2.5	2.6	3.0
OUTILLAGES	Etude, conception et POC pour la CMDB GEO / SNOW	=	5				
METIERS	SMART optimisation de la publication GEO pour Toulouse	+	3			X	
METIERS	Admin déléguée - corrections critiques & nouvelles évolutions	+	10		X	X	
SECU & DETTE	Support PostgreSQL 15	=	3		X		
METIERS	Demandes GEO pour les GEO Solutions - Lot 2	↗	20		X		
METIERS	GEO API Front pour les GEO Solutions	↗	5		X		
METIERS	Audit accessibilité Portail Enfance lot 1	+	10				
OUTILLAGES	Maintenance infra production de version GEO 2.5	+	8	2			
OUTILLAGES	Tests automatisés en phase avec les attentes QA des GEO Solutions	=	1	4	2		
OUTILLAGES	Environnements à la demande GEO / Docker	=	1	7	2		
SECU & DETTE	Travaux JAVA 11/17	=	20				
METIERS	Refonte module Envoi de données	+	15	3		X	
OUTILLAGES	Implémentation CMDB lot 1	=	8	8			
METIERS	Events Interactions autour des transactions (inclus GEO-3620)	=		15		X	
METIERS	Demandes clients HAUTES socle - Export recherche	=		9		X	
OUTILLAGES	Système de mise à jour automatique basé sur la containerisation	↖		2	1		
		Total jour	136	104	60		
		Capacité équipe	136	104	60		
		% engagement	60,00%	44,00%	20,00%		

À la fin de chaque standup, une version résumée est mis en ligne sur un site relié à l'intranet de l'entreprise dans le but de garder une trace de suivi sur l'évolution et des choix mis en place par les standup :

The screenshot shows a Jira board for the 'GEO Core' project under the 'Infos du jour' section. The board has three columns: 'Standups', 'GEO Core', and 'Table des matières'. The 'Standups' column lists 'Standups GEO R&D' and 'Infos du jour'. The 'GEO Core' column contains tasks grouped by status:

- A noter:**
 - Backlog grooming ce matin pour Geo sur docker
- En cours:**
 - Admin déléguée
 - partie back de l'import export est faite
 - pour le front: problématique de choix d'affichage de l'arbre des domaines
 - première proposition d'interface en cours
 - Tests U Lucene/Elastic
 - Hugo a pu rebrancher une 20ene de tests, bravo !!
 - Fix d'un test facet -> problème d'injection avec angular
 - Java 17 / AAS
 - On arrive à requeter certains des endpoints
 - Quelques soucis de serialization sur d'autres (notamment Facet)
- GEO Tech:**
 - A noter
 - *
- Table des matières:**
 - Général
 - GeoClip
 - A noter
 - En cours
 - GEO Key Civil
 - A noter
 - En cours
 - GEO Key
 - A noter
 - En cours**
 - GEO Core
 - A noter
 - En cours
 - GEO Tech
 - A noter
 - En cours

FIGURE 25 – Extrait d'un des standup sur GEO-Core

8.4.2 Sprint Review

Mon arrivée dans l'entreprise commença par une réunion de **Sprint Review** avec toutes les équipes.

L'objectif de ce Sprint Review est de faire une rétrospective sur tout les objectifs définis à l'occasion de ce Sprint :

- Test unitaire / test de non régression
- Problèmes de compatibilité PHP / Java et autres
- SDK Gabarit : Cette fonctionnalité est en lien avec ma mission principal car dans l'optique d'isoler la logique UI / DOM de la logique métier pour isoler le générateur GEO et pouvoir créer des interfaces personnalisées pour des futurs logiciels hors géographie.
- Meilleur API JS avec la gestion **d'end-point** précis pour mieux fluidifier la communication intra-logiciel.
- Création d'un docker GEO (que j'ai eu l'occasion d'essayer en le faisant fonctionner pour démarrer l'application GEO simplement)
- Amélioration des polygones et de la gestion des textures dans le gabarit GEO.
- Gestion du CRUD avancée avec des cas précis (notamment lors de conflit avec la base de données sur des dépendances / foreign key)
- Création d'un module de gestion de carte en 3D : notamment en lien avec des scène 3D et de la visualisation de bâtiments en 3D avancée.
- Formatage des données et sérialisation (vers des formats type .CSV / .XSLX)

Ce moment fut essentiel pour moi pour comprendre :

- L'étendu du projet GEO dans **sa complexité** (500k de ligne frontend et 500k backend)
- Ses contraintes techniques, fonctionnelles et organisationnel
- Les équipes qui travaillent sur ce projet et l'organisation autour ainsi que la supervision des tâches avec le logiciel Jira

8.4.3 Rétrospective

Les **Rétro** (diminutif de rétrospective) sont des réunions avec l'ensemble de l'équipe après une réunion Sprint Review pour faire un bilan non technique du Sprint / des 3 dernières semaines passés au sein de l'entreprise. Ces réunions permettent à l'ensemble de l'équipe de faire par de leurs avis sur comment s'est déroulé le Sprint qu'ils ont vécu : aspect positif tout autant que négatif. L'objectif est de faire le bilan humain et d'adapter la stratégie agile pour être meilleur.

8.4.4 Entretien perso

Une fois toutes les 3 semaines (durée d'un Sprint), le manager de l'équipe R&D prend le temps de voir individuellement chaque membre de l'équipe pour pouvoir discuter sur des sujets variés (lié au travail ou pas). Cette réunion n'est pas inclus dans la méthodologie Agile mais permet à chaque membre de l'équipe de faire un point sur sa situation professionnel au sein de l'entreprise et de pouvoir discuter.

TODO

8.5 Documentation ADR

Pour mieux me projeter dans ma mission et la comprendre, je commence à lire des Architectural Decision Records (ADRs).

C'est un standard pour documenter un dossier pour une idée, un concept, fonctionnalité ou autre chose.

Chacun de ces dossier explique des choix de l'entreprise en fonction de leurs facteurs de décision et des avantages et inconvénient.

Name
..
docs
M+ 0001-angularjs-next.md
M+ 0002-documentation.md
M+ 0003-batch-solution-auth.md
M+ 0004-zonejs.md
M+ 0005-vuejs.md
M+ 0006-vuejs-version.md
M+ 0007-fe-facade.md
M+ 0008-fe-state-management.md
M+ 0009-fe-typage.md
M+ 0010-telechargement-differentiel.md
M+ 0011-moteur-rendu-3d.md
M+ 0012-architecture-3d.md
M+ 0013-indexation-facet.md
M+ 0014-generator-extensibility.md
M+ 0015-validation-de-données.md
M+ 0016-generator-extension-dev-pattern.md
M+ 0017-3d-bim.md

Chaque dossier ADR possède :

- Un statut (proposition, WIP (en cours), accepté)
- Un groupe de personnes décideurs
 - Une date
- Un contexte et une problématique (pourquoi y'a t'il besoin de monter un dossier)
 - Les options considérées comme solutions à la problématique
 - L'option choisit (avec ces raisons, conséquences positives et négatives)
 - Les conséquences positives et négatives des autres options qu'on n'a pas été sélectionné
 - une webographie qui a servi à la prise de décision et pour contextualiser par rapport à d'autre ADR

FIGURE 26 – Aperçu des ADR sur le Gitlab de Ciril Group

La lecture des ADR me permet de mieux comprendre le contexte de l'entreprise et le cheminement utilisé pour arriver aux choix d'aujourd'hui. L'utilisation de diagrammes et de schéma comme celui des IHM (Interface Homme Machine)

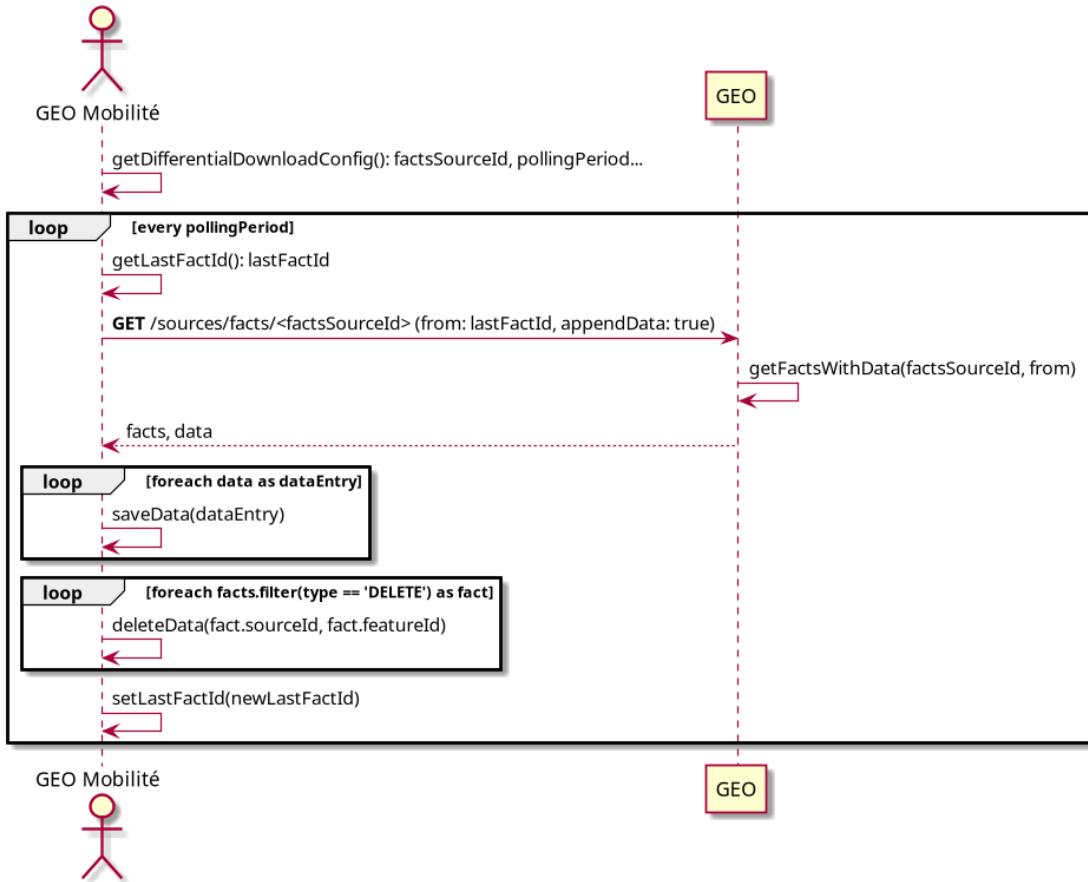


FIGURE 27 – Exemple d’IHM utilisé pour la gestion des endpoints dans GEO

9 Conclusion général

9.1 Conclusion du stage

L'ensemble du stage fut pour moi une expérience très enrichissante. Même si le projet et les missions associées à mon stage ne sont ni les plus amusantes ni les plus faciles, c'est en confrontant la difficulté avec persévérance que j'ai pu voir de l'évolution tout au long de ce mois malgré la densité et la complexité du projet.

Mon objectif pendant mon stage était d'observer et comprendre l'organisation en entreprise et de pratiquer pour apporter mon soutien au sein des objectifs de l'entreprise, j'ai pu apprendre beaucoup de choses à questionnant l'organisation mais aussi le fonctionnement de l'application avec l'équipe R&D ainsi que les product owner (PO).

La mission sur laquelle j'ai pu échanger pendant 1 mois n'est pas une tâche facile pour n'importe quelle entreprise qui livre des solutions logiciels (surtout pour un produit comme GEO qui est un projet relativement grand en terme de fonctionnalité, code et architecture). Lors de difficultés, il n'y avait pas tout le temps des gens disponibles pour pouvoir m'aider (la R&D étant un département de l'entreprise où il y a toujours du travail et beaucoup de réunion avec les PO), il faut donc savoir trouver des solutions individuellement en autonomie.

J'ai pu prendre conscience des différentes problématiques front et back lors de ce projet, la séparation des logiques est une notion fondamentale dans la nouvelle organisation et le nouveau fonctionnement du générateur GEO. Grâce aux différentes documentations, aux documents sur l'innovation / améliorations (qui ne peuvent être révélé), j'ai pu prendre consciente de l'enjeu économique et entrepreneurial de ma mission. Enfin, j'ai pu faire une grande introspection sur moi-même et gagner en maturité côté gestion de projet et humains, ce qui m'est extrêmement bénéfique dans ma poursuite d'étude et dans ma future vie professionnelle.

9.2 Compétence validé en Bloc 1

Tableau des compétences Bloc 1 BTS SIO

Gérer le patrimoine informatique : Programmation de code versionné avec l'utilitaire **Git** puis envoyé sur un dépôt **Gitlab** tout en utilisant le système de **branche** pour le déploiement individuels de fonctionnalité ou de correction de bug en finalisant les modifications par une **merge-request** selon un protocole de normes et de standard lié à l'entreprise pour la mise en conformité du code et sa vérification pour son intégration définitive au sein de l'application. Réalisation de documentation adéquate pour expliquer le fonctionnement d'une architecture et de la gestion des services aux futurs développeurs sur le projet. Respect des **normes et standards** de l'entreprise et l'**OGC** concernant la gestion et l'utilisation de données géo-spatiales.

Répondre aux incidents et aux demandes d'assistance et d'évolution : Système de ticket et de gestion d'incident avec le logiciel Jira (similaire à GLPI, adapté pour la méthodologie Agile). Procédure de gestion de demandes :

- **Ticket Jira** : Processus de demande client auprès des PO (Product Owner) de GEO qui redirige l'objectif / le ticket vers le manager d'une équipe adapté au produit (R&D par exemple), puis gestion et diagnostique ainsi que résolution du ticket selon la procédure de l'entreprise et l'évolution du statut.

- **Merge request Gitlab** : Procédure d'intégration du code au projet principal par une review humaine sur plusieurs aspects (fonctionnel, esthétique, coding style) et automatisé grâce à un robot qui passe par dessus le code. L'objectif étant traiter une demande de suivis et d'orientation de l'évolution du code selon des normes à son intégration dans l'application.

Développer la présence en ligne de l'organisation : Participation à l'évolution de **l'interface graphique du générateur GEO (Facette)** dans le but d'une **meilleure expérience utilisateur** en terme de rapidité , fluidité du logiciel mais aussi en terme **d'harmonisation de l'architecture logiciel** en séparant les **logiques UI et métier**.

Travailler en mode projet : Intégration de l'équipe R&D de Business Geographic avec un fonctionnement via la méthodologie Agile combiné avec des différentes réunions (Stand Up, point synchro, Sprint Review, Rétrospective, Entretien perso). Gestion de ticket et gestion de projet avec le logiciel Jira (dont l'utilisation de Kanban). Versionning via Git et Gitlab pour voir l'évolution du projet et pouvoir interagir facilement en cas de besoin spécifique.

Mettre à disposition des utilisateurs un service informatique : Le projet Facette est une interface graphique front-en en Vue JS 3 (**application web**). Tout au long de son développement et sa montée de version, des plan de test et des rapports sur le framework dédié Histoire ont été réalisé, ainsi que une **documentation technique et utilisateur**. Elle a été testée avec Jenkins (utilitaire de **CI/CD**).

Organiser son développement professionnel : Mise en place d'une veille orienté **cybersécurité** quotidienne en suivant des personnes spécialisés sur LinkedIn pour être tenu au courant des **dernières actualités / CVE / risques et vulnérabilités** sur des technologie. (**Julien Metayer, Jérôme Bezet-Torres, Mathis Hammel**,<https://veille-cyber.com/>,<https://medium.com>, twitter, flux rss spécialisé, etc).

Dans le but de pouvoir développer mes compétences et **d'assurer une veille technologique et cybersécurité**, je réalise des **formations / MOOC / parcours de certification** en continue. Ces certifications et MOOC sont disponibles sur Mon LinkedIn.

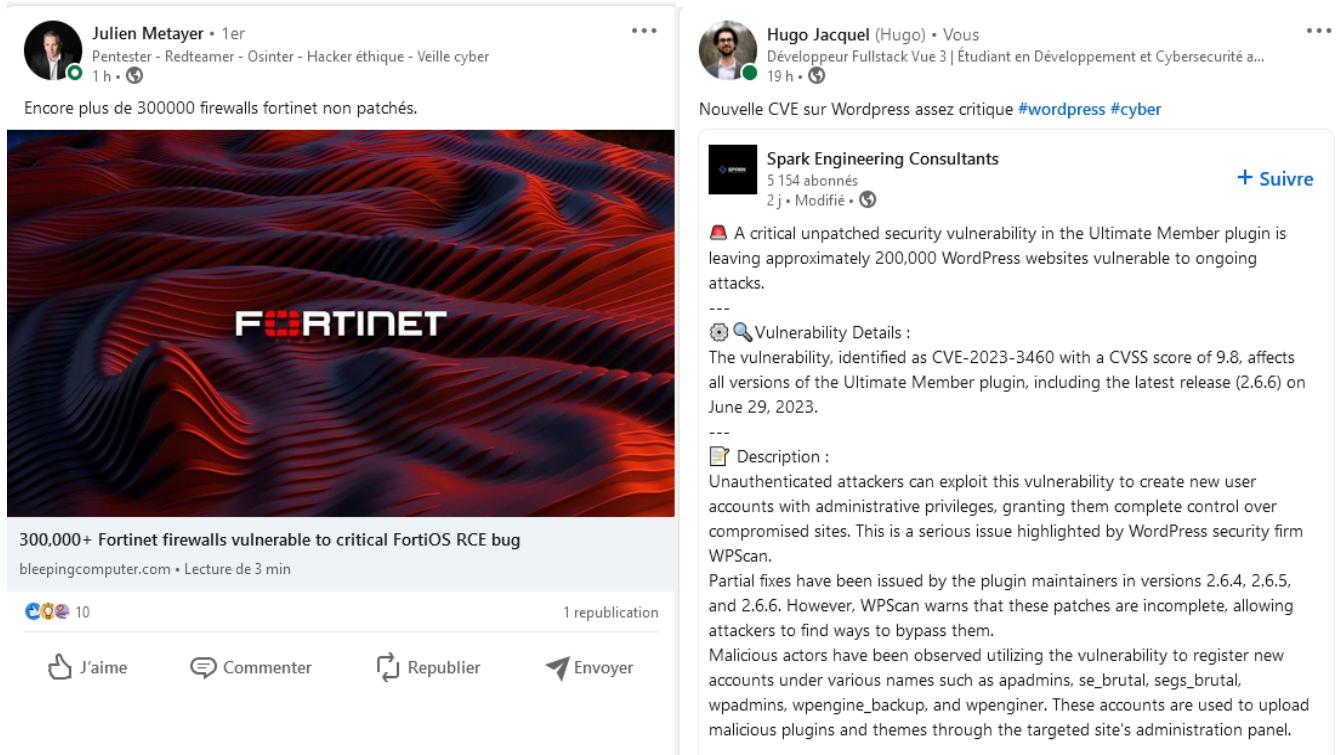


FIGURE 28 – Screenshot LinkedIn lié à la veille cyber

10 Documentation utilisateur

10.1 Présentation de l'application

Bienvenue dans la nouvelle interface utilisateur du générateur GEO : **Facette** !

The screenshot shows the Facette application interface. At the top, there is a search bar labeled "Rechercher une ressource" and various filter icons. Below the search bar, there are two main sections: "Applications" and "Tableaux de bords". The "Applications" section contains several items, including "test GP", "test module.js", "Ma nouvelle application", "tableau pivot", and "test". The "Tableaux de bords" section contains items like "tableau pivot", "test", "Density of accident (per...)", "الخل", and "لدون". On the left side, there is a sidebar with a "CATALOGUE" section containing categories like "Applications", "Tableaux de bords", "Images", "Thèmes", "Extensions", and "Alertes". There is also a "AUTEUR" section with entries like "geo_account" and "testing testing".

La nouvelle interface à était conçue pour **intégré une recherche à facettes** qui permet de faciliter l'utilisation de l'application, la recherche d'information et de vos ressources GEO.

Tout d'abord en haut de l'application se trouve la recherche par facettes avec :

- Un bouton pour la création de vos ressources GEO (image, carte 2D / 3D, tableau de données, polygones, marquages, extensions, fonctionnalités, applications, etc.)

- une barre de recherche intelligente pour retrouver vos ressources GEO et celles de vos collaborateurs
- Une série de bouton pour consulter des données spécifiques : Données, Fonctionnalités, Cartothèque, Applications)
- Un bouton pour filtrer les données (auteur, date de création, date de modifications, est-elle partagé ?, etc.)

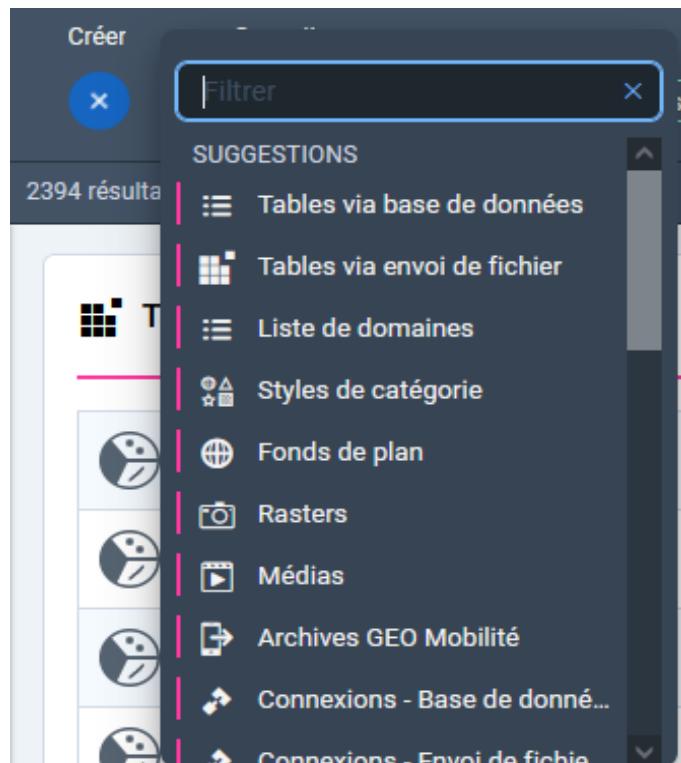


FIGURE 29 – Bouton de création de ressources dans GEO Générateur

Pour finir avec le menu du haut, il y a la possibilité de trié les données et d'organiser leurs visualisation sur Facette.

Sur la gauche, un menu de configuration est disponible dans le but d'afficher les ressources disponibles directement selon leurs catégories et leurs sous catégories de manière hiérarchique. Par exemple des ressources "polygone" sont rangé dans la sous-catégorie "Table spatiale" elle même dans la catégorie Tables.

Pour finir, au centre de l'application : la visualisation des ressources avec pour chacune :

- Son nom
- Un aperçu (si disponible)
- Sa catégorie
- Son auteur
- La date de création, modification
- Une série d'action possible (exporter, visualisation, édition, copier)

10.2 Utilisation des ressources GEO

10.2.1 Données

Les **données** regroupe l'intégralité des ressources GEO qui permettent de fournir du contenu à une application (hors carte car c'est une catégorie à part de ressources), parmi ces données vous pouvez **importer** et **créer** des **bases de données, médias, images, tableau excel** ou n'importe quelle structure de données compatibles). Le générateur GEO vous permet de les éditer directement dans le logiciel pour qu'elle soit prêt à l'emploi.

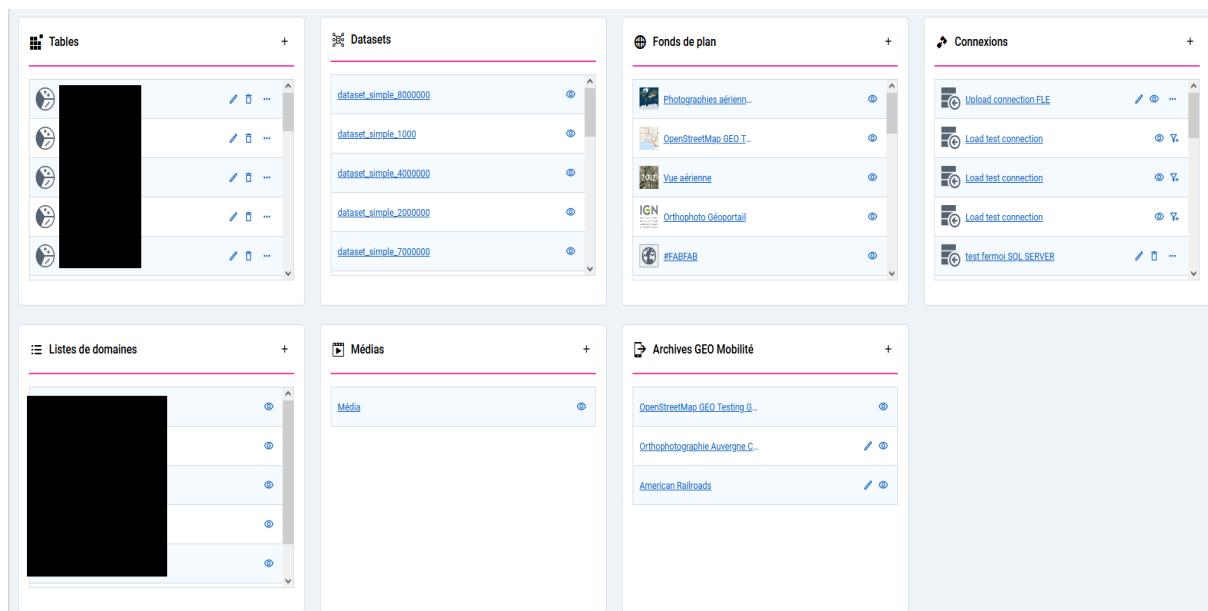


FIGURE 30 – Exemple d'interface de donnée possible dans le générateur GEO

The screenshot shows the 'Données' tab selected in the top navigation bar. Below it, a table named 'gadm36_FRA_3' is displayed in a detailed view. The table structure includes:

Nom	Calculé	Type	Alias du champ	Affichage	Description
CC_3	Non	STRING	CC_3	Texte brut	
ENGTYPE_3	Non	STRING	ENGTYPE_3	Texte brut	
geo_creationdate	Non	TIMESTAMP	Creation date	Texte brut - Temporel	
geo_fid	Non	INTEGER(10)	Reserved	Texte brut	
geo_modificationdate	Non	TIMESTAMP	Modification date	Texte brut - Temporel	
geom	Non	GEOMETRY(EPSG:4326)	Geom	GEOMETRY	
GID_0	Non	STRING	GID_0	Texte brut	
GID_1	Non	STRING	GID_1	Texte brut	
GID_2	Non	STRING	GID_2	Texte brut	
GID_3	Non	STRING	GID_3	Texte brut	
HASC_3	Non	STRING	HASC_3	Texte brut	

FIGURE 31 – Édition d'une base de données SQL dans le générateur GEO

10.2.2 Fonctionnalités

Les **fonctionnalités** sont des fonctions qui permettent de relier des ressources (données et cartes) avec l'interaction d'un utilisateur (notamment lors de la création d'une application au sein du générateur GEO). Par exemple des fonctionnalités de recherches selon des critères précis (on peut imaginer une application qui permet de centralisé les restaurants selon des critères comme la localisation, des spécialités, etc.)

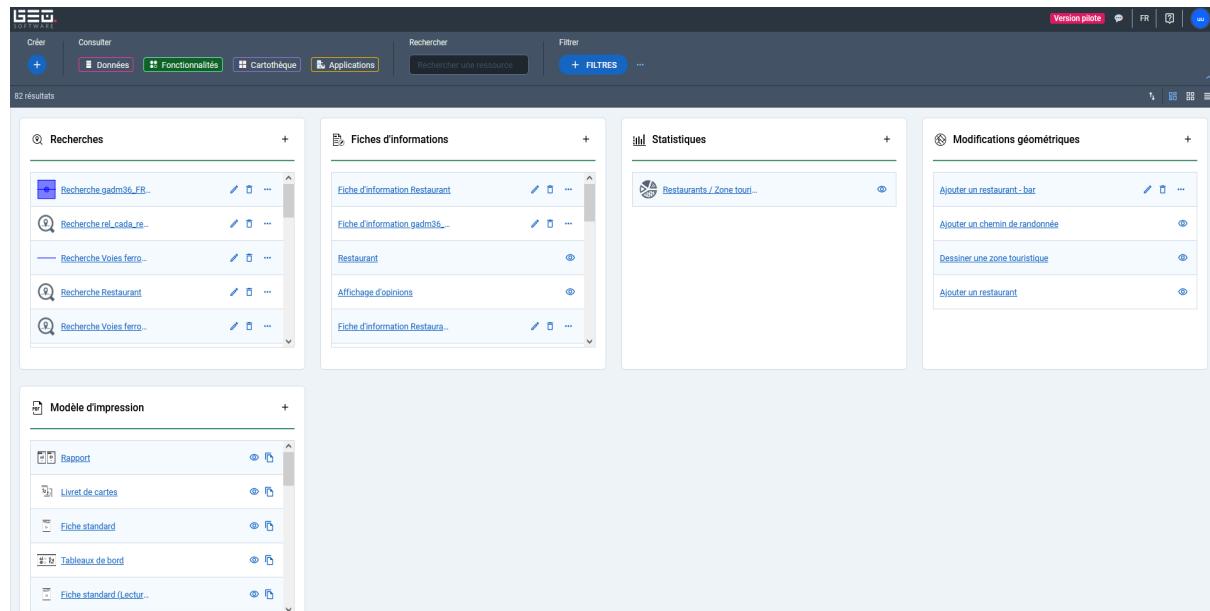


FIGURE 32 – Exemple d'interface pour les fonctionnalités dans le générateur GEO

10.2.3 Cartothèque

La **Cartothèque** désigne l'ensemble des ressources qui sont reliés à des cartes (2D ou 3D). GEO possède directement un éditeur de carte interactif permettant l'analyse et la segmentation des parties d'une carte selon vos critères où vos besoins (élections, gestion des eaux, électricité, routes, etc.). L'éditeur permet la création de polygones et d'objets dynamiques adaptés aux besoins.

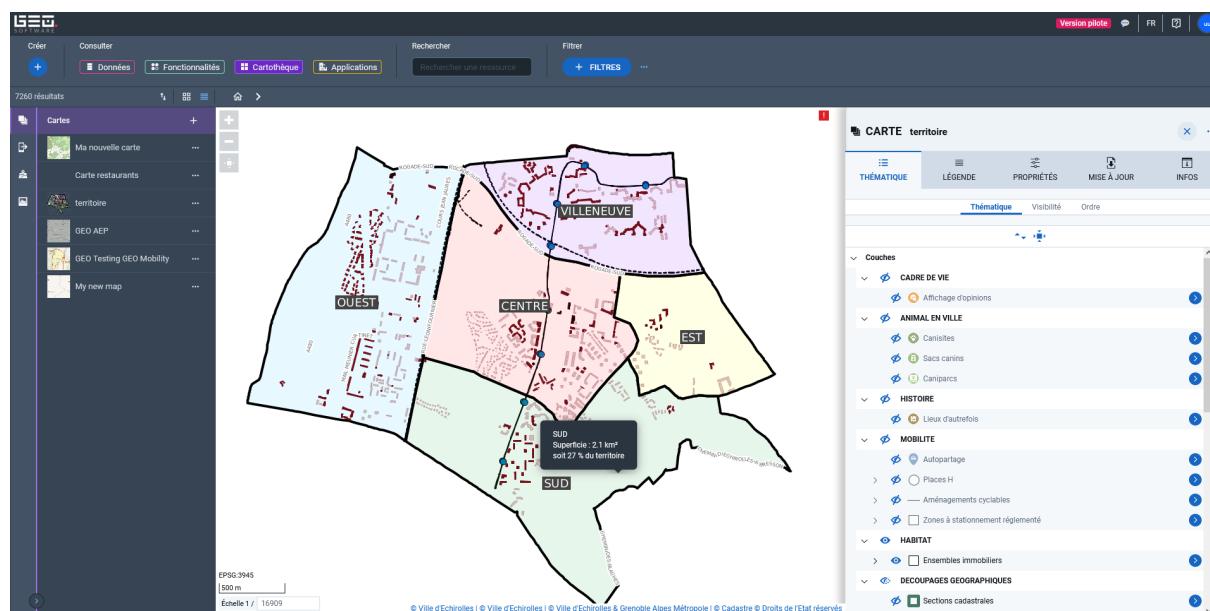


FIGURE 33 – Interface de l'éditeur de carte dans le générateur GEO

10.2.4 Applications

Enfin, le générateur GEO vous permet de fusionner l'ensemble de vos ressources et de vos fonctionnalités en créant une **application**, GEO possède un éditeur d'application pour concevoir vos applications selon vos besoins et critères. Par la suite, cette application peut être exporter pour être utiliser grâce au moteur GEO.

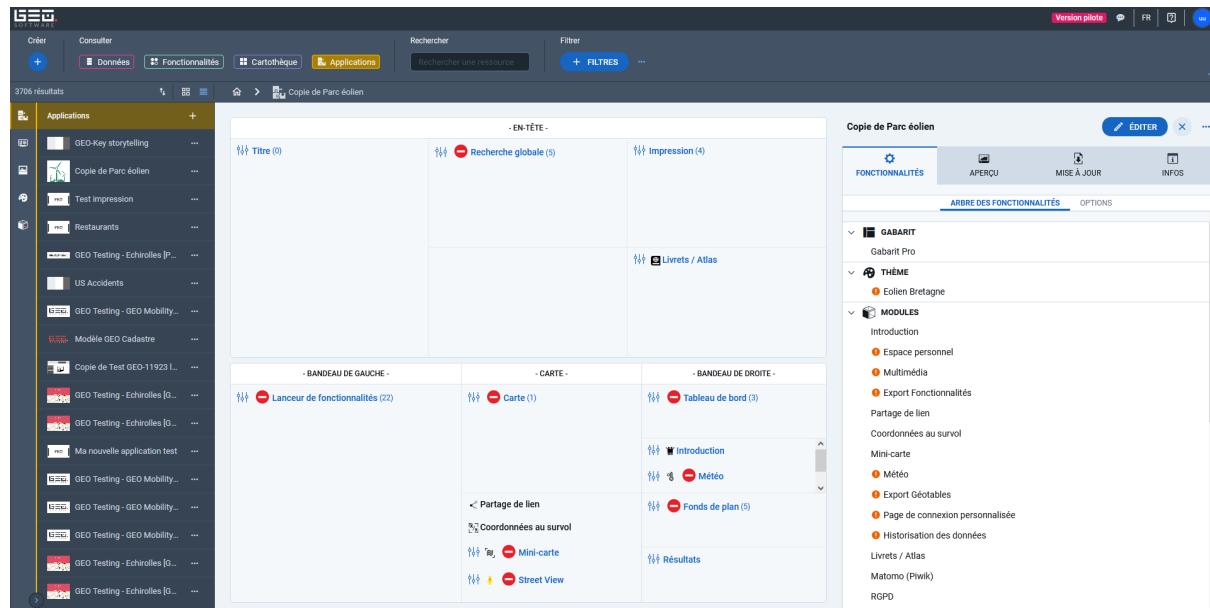


FIGURE 34 – Interface de création d'application dans le générateur GEO

10.3 Installation

Pour pouvoir acquérir l'application GEO, vous pouvez obtenir un entretien avec l'équipe CIRIL Group pour souscrire au produit et pouvoir :

- Obtenir un accès via une licence sur le site web de GEO et pouvoir l'utiliser sur tout vos appareils (ordinateur, téléphone, tablette)
- Demander une installation dans vos locaux sur vos machines avec une équipe dédié de Business Geographic.

11 Documentation technique

Nous présenterons ici des concepts fonctionnels et techniques liés à la deuxième version du générateur, souvent désignée sous le vocable **Facette**.

Le développement de Facette était initialement prévu pour la sortie de GEO 2.0, ce faisant, même s'il est effectivement livré depuis cette version, il n'est cependant activé que depuis la 2.3. Cependant Facette est souvent synonyme de GEO 2.

Une recherche à facettes désigne un mécanisme de filtrage d'un ensemble de données à travers le croisement de plusieurs dimensions appelé facette.

Dans ce cadre, une recherche est souvent exprimée comme suit : Je cherche les ressources GEO de la famille des 'fonctionnalités', pour l'auteur 'X' ayant le mot-clé 'Z' et lié à l'application 'Tutoriel'. En d'autre terme, nous cherchons à limiter notre espace de recherche en filtrant sur les facettes :

- famille de fonctionnalité : une propriété d'une ressource GEO.
- auteur : une propriété d'une ressource GEO.
- mot-clé : une propriété d'une ressource GEO.
- application : un lien de dépendance entre deux ressources GEO

Mettre en place ce système impose une évolution sur notre paradigme de recherche en utilisant d'une structure de donnée optimisée pour ce genre d'opération, un index inversé implémenté par la librairie Lucène.

11.1 Vocabulaire Technique

The screenshot shows the GEO software interface with a red border around the top header and a green border around the main content area. The header includes 'Header Tools' (Créer, Rechercher, Consulter), 'FacetConfigurationLauncher' (Données, Fonctionnalités, Cartothèque, Applications), 'Filtrer', and 'FILTRES'. The main content area is divided into several sections: 'CATALOGUE' (Tables, Facette Hiérarchique, Cartes, Recherches, Analyses, Statistiques), 'Applications' (PRO, Card, GEOMOBIL-1056, Analyses, Tables, Fiches d'information, Cartes), 'ResourceExplorer' (Raster, Cartes, GEOMOBIL-1056 - Ajouter un restaurant, Analyses, Tables, Fiches d'information, Cartes), 'Recherches' (GEOMOBIL-1056 - Mes restaurants, GEOMOBIL-1056 - Recherche de restaurant, Analyses, Analyses, Analyses), and 'Statistiques' (Nombre de friches restantes, POC communes couleurs + norme, Analyses). A sidebar on the left shows a tree view of 'SidiConfiguration' with categories like 'Tables', 'Facette Hiérarchique', 'Cartes', etc.

Sur la disposition de contenu :

- **Header** : Contenu supérieur de l'application web, composé de trois lignes :
 - > des fonctionnalités historiques (gestion de droit / notification / langues ...), liste non extensible.
 - > des outils (HeaderTool), une zone extensible permettant d'interagir ou de filtrer des ressources.
 - > l'accès aux modes de représentation (catégorisé / en carte / en liste).

- **Catalogue** : Première catégorie présente dans le panel de filtre de la configuration latérale.

- **SideConfiguration** : présent uniquement dans Facette, c'est une structure de donnée qui représente la configuration du contenu latéral et en premier lieu le component - `FacettePanel.vue` qui gère le rendu d'un `FacetteTree`. celle de l'arbre des facettes. Cette configuration est exprimée directement sur le routing permettant une application précoce et de rester cohérent avec le fonctionnement de *GEO V1*

- **View** : décrit historiquement le contenu d'une application, géré par le `ViewStackService`. Tous les éditeurs sont des `View`.

- **StackableView** : type de `View` qui peut se superposer les une sur les autres, c'est notamment le cas des éditeurs. Celle-ci sont gérées dans le service `ViewStackService`.

- **Hub** : représentation d'un ensemble de catégories de ressources.

- **Editor** : représentation centrée sur une ressource qui permet son édition et de naviguer vers des éditeurs de ressource en relation avec celle en édition.

Sur les facettes :

- **FacetteConfiguration** : une configuration de facette est un paramétrage permettant de déterminer le comportement de l'arbre de filtres en fonction d'un état de routing, d'un état de l'UI, et bientôt de l'application d'un raccourci.

- **FacetteConfigurationNode** : une entrée de configuration simple composée d'une propriété key désignant une valeur de facette, et d'une propriété facet.

- **FacetteConfigurationLauncher** : un raccourci accessible dans l'UI permettant l'activation d'une configuration de facette.

- **FacetteContext** : un contexte de facette est l'ensemble des inputs d'un utilisateur dans une navigation donnée. Au cours d'une navigation plusieurs contextes existent, mais un seul est actif et appliqué.

- **FacetteQuery** : une requête de facette est la transcription d'un ensemble de paramètres permettant au backend d'exécuter une opération de filtrage.

- **FacetteFilter** : propriété d'une `FacetteQuery`, c'est une structure de donnée serializable qui décrit les combinaisons de filtres à appliquer.

- **Facette Hiérarchique** : Une facette hiérarchique est une facette imbriquée dans autre facette.

- **FacetteTreeConfigurationNode** : structure de donnée utilisée pour la description complète d'un arbre de facette.

- **FacetteTree** : structure de donnée utilisée pour la représentation de donnée, en particulier celle du panneau de facette. Cette structure est générée par une configuration exprimée en `FacetteTreeConfigurationNode`.

- **FacetteNode** : structure de donnée utilisée pour la représentation de donnée et membre d'un `FacetteTree`.

- **FacetteNodeData** : Donnée provenant de la configuration d'un nœud et présente dans un `FacetteNode`. Cette donnée provient du serveur.

Sur les outils :

- **CreationLauncher** : Un component qui permet d'accéder à une liste de **GeneratorCreationLauncher**.
- **GeneratorCreationLauncher** : Permet d'appeler un éditeur en création pour une ressource donnée.
- **FacetteConfigurationLauncher** : Lanceur permettant d'exécuter une **FacetteConfiguration** dans un contexte de lecture.

Sur les ressources :

- **ResourceAction** : décrit une action cliquable à partir d'une ressource, que ce soit directement à partir de la vue d'exploration ou dans l'éditeur.
- **ResourceDescriptionDataService** : service centralisant un ensemble de propriété propre à un type de ressource.

11.2 Cycle de vie : Architecture logicielle

Le **cycle de vie** de l'application est celui d'une **application Angular JS classique** :

- L'application s'exécute à partir du moment où angular.bootstrap est appelé (voir le fichier main.ts).
 - Les callbacks de configuration sont exécutés en premier, parmi les plus notables :
 - > Initialisation du système d'internationalisation.
 - > Configuration de Vue.
 - > Initialisation d'icônes.
 - > Routing http de l'application.
 - Les callbacks de run sont exécutés à la suite, parmi les exécutions notables :
 - > post install du système d'internationalisation.
 - > enregistrement des extensions du générateur.
 - > enregistrement de l'UI.

Attention **toute ces exécutions sont synchrones**, il n'y a pas de gestion de **l'asynchronisme**.

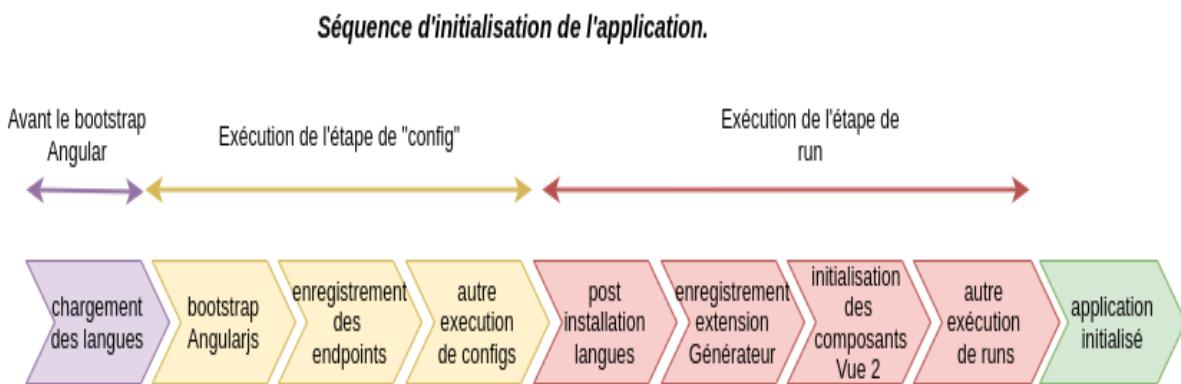


FIGURE 35 – Schéma regroupant les différentes séquences d'initialisation / chargement des modules de l'application

Au cours de l'exécution de l'application il faut comprendre que :

- la description de l'arbre en `FacetteTreeConfigurationNode` disponible dans le `FacetteTreeConfigurationModule` n'est jamais modifié (même si techniquement possible). Le paradigme est de dire que le backend fournit une description de l'arbre de filtre exhaustive qui n'est pas altéré par le front-end.

- le `FacetteTree` est fréquemment reconstruit quand on communique avec le backend pour exécuter des recherches. Le point d'entrée de cette logique est la méthode `refreshTree` dans `FacetteQueryService`.

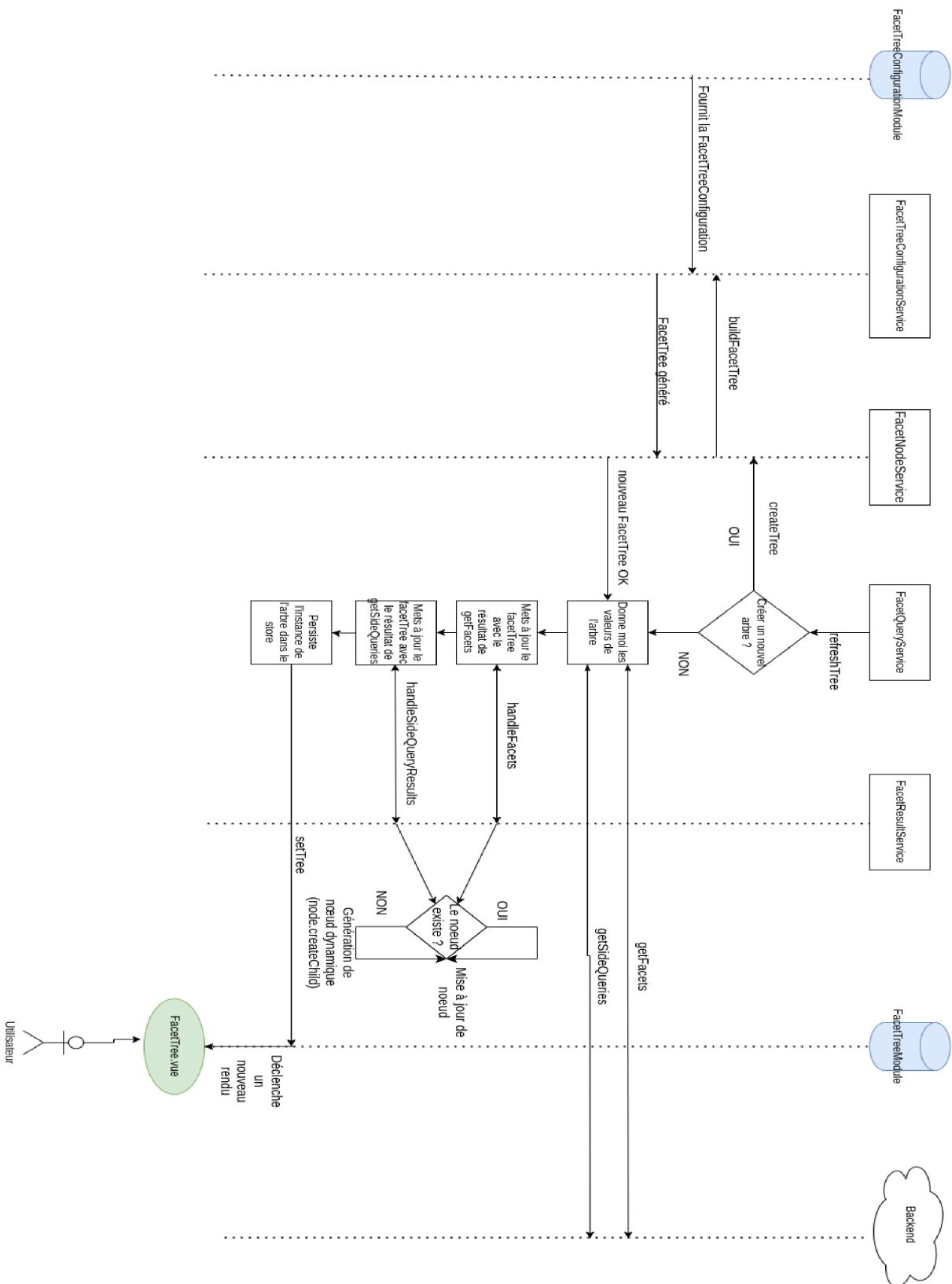


FIGURE 36 – Schéma IHM sur la mise à jour d'un arbre de ressources GEO ‘FacetteTree’

11.3 Logique de construction de requêtes

L'application communique fréquemment avec le backend pour appliquer l'ensemble des filtres que l'utilisateur demande.

Ces derniers sont construits de différentes manières en fonction du type d'information demandé.

D'autre part, il est important de comprendre historiquement le `FacetteTree`, responsable de la représentation de l'arbre, est la structure de donnée utilisée également pour la construction de requête.

Or, cette implémentation souffre de limitations :

- Il est impossible de manipuler un nœud qui n'existent pas dans l'arbre, compliquant l'implémentation de certaines fonctionnalités (contexte de filtre passé par URL, filtrage depuis une ressource, ...).

- L'implémentation confère un statut particulier à la facette catalogue, point qui a terme posera problème (extensibilité des `FacetteConfigurationLauncher` par exemple).

11.3.1 Typologie de requête

Plusieurs requêtes sont jouées, le détail est visible dans `FacetteRestClient`. Nous détaillerons ici les deux grandes familles :

- les requêtes qui mettent à jour le `FacettePanel` et principalement le `FacetteTree`.

- les requêtes qui récupèrent les ressources à afficher que ce soit pour une visualisation par catégorie ou non.

Dans le détail, pour le requêtage de ressource, on dissocie requête pour une typologie de ressource spécifique (cadre de la vue catégorisé) et requêtage non catégorisé. La différence s'exprime sur l'ensemble de nœud sélectionné, en relation ou non à une valeur spécifique.

Les requêtes utilisées pour mettre à jour l'arbre reposent sur deux types :

- les appels sur l'endpoint `getFacettes` qui mettent à jour "naïvement" l'arbre, sans prendre en compte les facettes hiérarchiques.

- les `sideQueries`, qui désignent des requêtes de complément pour récupérer les valeurs de facette pour une hiérarchie donnée.

11.3.2 Structure de requête

Deux interfaces à comprendre :

- La structure de donnée avant la transformation en requête : `FacetteSearchParameters`.

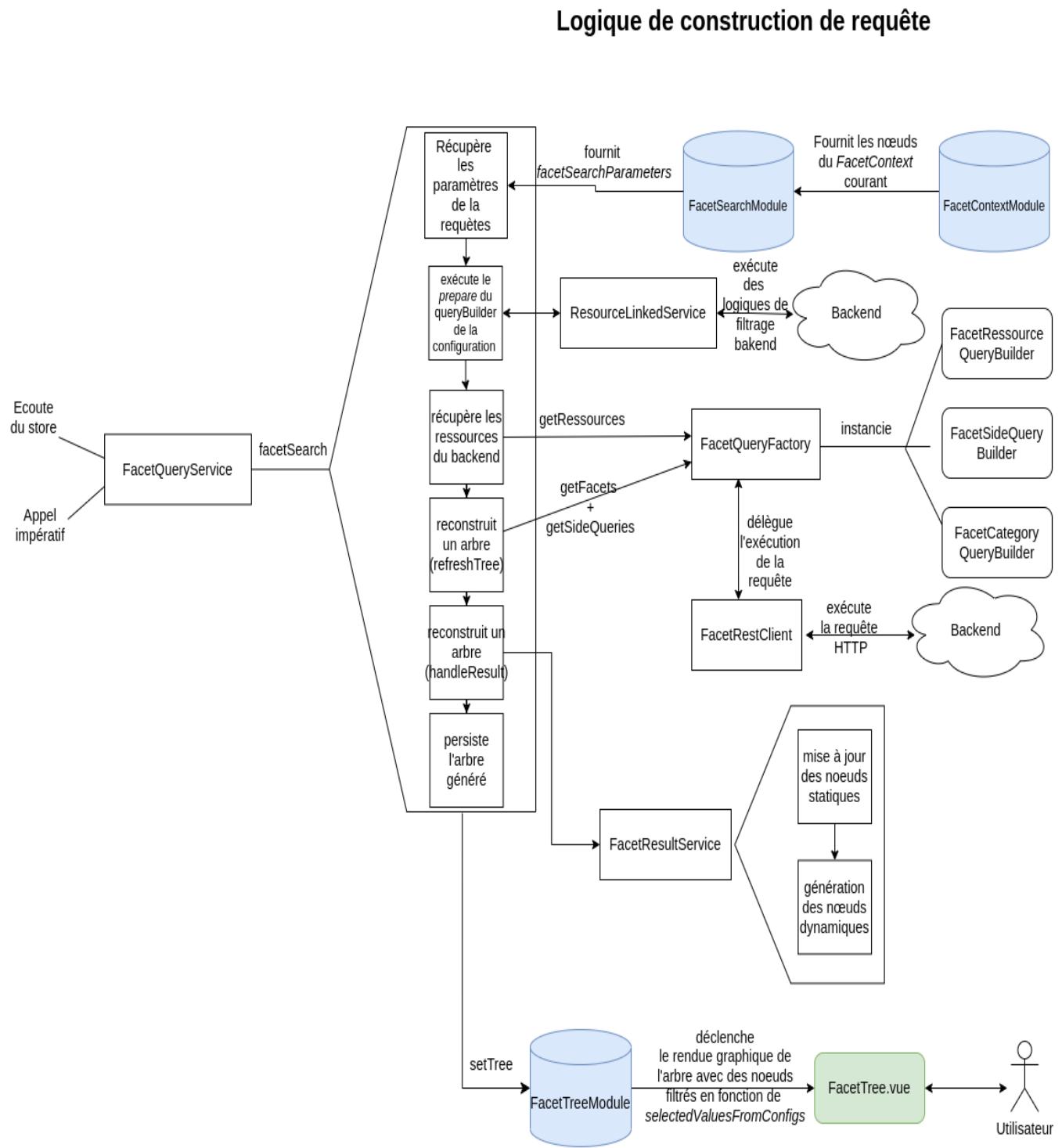
- La structure de donnée communiquée au serveur : `FacetteQueryParameters`.

Quelques propriétés intéressantes dans `FacetteSearchParameters` :

- `facetParams` : permet de récupérer les valeurs des facettes, c'est-à-dire le nombre de résultats disponibles pour chaque dimension après application des filtres.

- `nodes` : désigne les combinaisons de facet et de valeur à transcrire en filtre. Les nœuds proviennent du `FacetteContext` courant afin de garantir une isolation des inputs en fonction d'un contexte d'utilisation dédié.

Ci-dessous, Schéma IHM diagramme C4 sur l'ensemble du cycle de construction et instanciation d'une recherche Facette :



11.4 Installation

Avant de passer à la phase d'installation de Facette, veillez à avoir sur votre ordinateur :

- Java (version 8 ou ultérieur)
- Les framework Spring, & Maven et Play !
- NPM et Node JS (Dernière LTS)

Pour installer la nouvelle interface graphique Facette, il faut :

- Cloner le repo **aigle-5 web** (`git clone`)

1) Lancer les 2 conteneurs dockers pour la connexion avec les services GEO et la base de donnée interne en allant dans le dossier `stack-geo` et en exécutant le script `./up.sh` avec votre terminal ou **Git bash** si vous êtes sous Windows.

- 2) Démarrer l'installation du serveur web avec `mvn -Pdev compile` à la racine du projet , puis le lancer avec le service d'authentification avec `mvn play2:run`.

- 3) Une fois le serveur web démarrer, rendez vous dans le dossier `geo-sources` avec `cd geo-sources` , puis faite l'installation des paquets nécessaire avec `npm install` et lancez Facette avec `npm build:watch` .

- 4) Rendez vous sur `localhost:9000` pour vous connectez et accéder au générateur GEO, puis une fois authentifier, activé la nouvelle interface depuis une icône spécifique sur votre profil.