Développement d'une plateforme de gestion de ramassages de déchets

RAPPORT DE STAGE

Du 15 janvier au 05 avril 2024

Entreprise : **Natural Solutions**

Tuteur en entreprise : **Vincent BOURGEOIS**, Développeur Full Stack

Giuliana GODAIL-FABRIZIO

BUT Informatique, 3ème année

Tuteur IUT : **Corinne PATERLINI**

Remerciements

Je tiens tout d'abord à exprimer mes sincères remerciements à Monsieur Vincent BOURGEOIS, mon maître de stage, développeur full stack, pour sa grande disponibilité et ses précieuses explications tout au long de mon travail. Grâce à son accompagnement, j'ai pu acquérir de nombreuses compétences et apprécier pleinement ma mission.

Le reste de l'équipe a également été d'un grand soutien et m'a chaleureusement accueillie. Je remercie chacun de mes collègues pour m'avoir consacré du temps lorsque j'en ai eu besoin.

Je suis aussi reconnaissante envers ma tutrice pédagogique, Madame Corinne PATERLINI, pour ses conseils avisés durant la réalisation de ce rapport et le déroulement de mon stage.

Par ailleurs, je salue l'ensemble de l'équipe pédagogique de l'IUT pour m'avoir transmis de solides connaissances qui se sont révélées essentielles pour la réussite de ce stage.

Enfin, je témoigne ma gratitude à tous ceux qui ont contribué à faire de ce stage une réussite humaine et professionnelle, ainsi qu'à toutes les personnes qui m'ont soutenue dans la rédaction de ce rapport.

Sommaire

Introduction

Face à la croissance alarmante de la pollution, la préservation de l'environnement est devenue une préoccupation mondiale majeure. Dans ce contexte, de nombreuses associations ont émergé pour lutter contre cette dégradation.

Depuis sa fondation en 2018, Wings of Ocean organise des missions de ramassages de déchets à travers la France, ainsi qu'en mer. Cependant, la gestion de ces missions peut rapidement devenir fastidieuse, répétitive et chronophage, d'autant qu'elle implique de traiter un grand volume de données.

L'automatisation de cette tâche offre une solution efficace. Grâce à celle-ci, il est possible de réduire considérablement le temps nécessaire à la gestion de ces missions tout en minimisant les risques d'erreurs liées aux fautes de frappe ou aux omissions.

Afin de simplifier la planification des missions de ramassages, la collecte d'informations et l'analyse statistique des données, Wings of Ocean s'est tournée vers Natural Solutions, une PME spécialisée dans le développement Web. J'ai été intégrée en tant que stagiaire, à l'équipe en charge de ce projet. Ma mission consiste à participer au développement d'une application répondant aux besoins spécifiques des membres de l'association.

Dans un premier temps, je présenterai Natural Solutions ainsi que son secteur d'activité. Je décrirai ensuite le cahier des charges en détaillant les missions qui m'ont été confiées avant de présenter les solutions mises en œuvre. Pour finir, je partagerai mes réflexions sur mon travail ainsi que les enseignements humains et pédagogiques que j'ai retirés de ce stage.

1 Présentation de l’entreprise

1.1 Secteur d’activité

Natural Solutions est une entreprise privée du domaine de l’informatique. Cette PME de 39 salariés développe principalement des outils (applications Web et mobiles) pour des acteurs environnementaux tels que Wings of Ocean. Sa raison d’être est de mettre le meilleur de la technologie au service des acteurs de la biodiversité.

1.2 Histoire

Natural Solutions a été fondée en 2008 à Marseille par Monsieur Olivier ROVELLOTTI, à la suite d’une mission de préservation d'espèces menacées d'extinction au Maroc. Cette expérience lui a permis de prendre conscience que les connaissances du monde numérique pouvaient être appliquées à la sauvegarde de la biodiversité et de l'environnement. Depuis, l’entreprise s’est développée et a acquis une expertise de plus en plus importante dans le domaine de la Tech au service de la Biodiversité.

Au cours du temps, les offres se sont élargies et structurées, ce qui a permis de proposer des services et des produits à divers acteurs : parcs régionaux et nationaux, réserves naturelles, associations et autres. Cela a progressivement conduit au développement de Natural Solutions à l’international : récemment, deux membres de l’équipe sont partis à l’étranger (l’un au Canada et l’autre à d’Abu Dhabi) pour y accompagner les acteurs de la biodiversité.

1.3 Organisation de Natural Solutions

Au sein de Natural Solutions, différents pôles sont établis, chacun ayant une mission spécifique. La figure ci-dessous présente une brève description de chaque pôle.

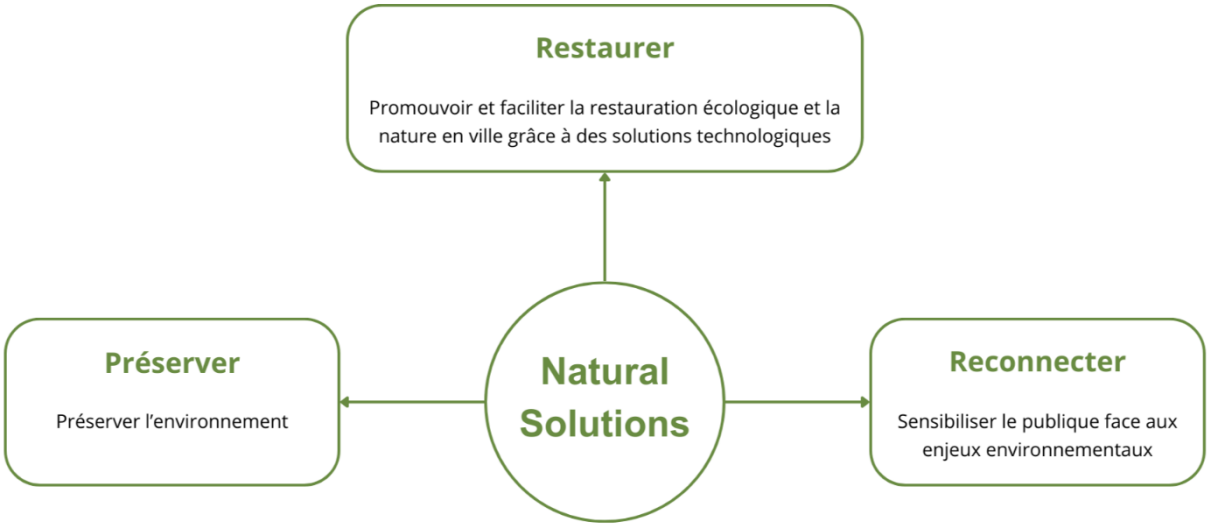


Figure I : Organisation interne de Natural Solutions

Pour ma part, j'ai intégré le pôle « Restaurer ». Au sein de ce département, plusieurs équipes se consacrent à différents projets. Mon équipe se composait de deux autres développeurs, d'un chef de projet et d'une designeuse. Notre mission était de développer une plateforme de gestion des ramassages de déchets pour l'association Wings of Ocean.

2 Présentation du sujet

Wings of Ocean est une association dédiée à la préservation de l'environnement. C'est pourquoi elle organise régulièrement des missions de ramassages de déchets dans diverses régions de France. Comptant près de 27 employés et 350 adhérents, elle bénéficie également du soutien de 700 bénévoles en moyenne chaque année.

Actuellement, la gestion des missions de ramassages et des bénévoles se fait au travers de fichiers Excel ce qui rend la tâche complexe et coûteuse en temps. De plus, la réalisation de statistiques est également très compliquée, car elle nécessite d'extraire manuellement, pour chaque ramassage, les informations relatives à chaque déchet collecté.

Pour optimiser ce processus, l'association s'est tournée vers Natural Solutions en octobre 2023 dans le but de concevoir un projet visant à automatiser la gestion des missions de ramassages. Le projet, appelé DepollutionMap, a été lancé fin novembre. En janvier, un développeur en alternance et moi-même avons rejoint l'équipe alors composée de mon maître de stage, d'une designeuse et d'un chef de projet.

Mon objectif pendant ce stage a été de contribuer au développement de DepollutionMap. Ma mission a principalement consisté à développer le tableau de bord (Dashboard) et à implémenter des fonctionnalités telles que l'ajout, la modification ou la suppression de tout ce qui se rapporte aux ramassages (partenaires, photos…).

3 Cahier des charges

Natural Solutions et Wings of Ocean ont conclu un contrat agile. En d'autres termes, les deux parties ont opté pour une approche flexible dans le développement de l'application. Contrairement à un contrat traditionnel avec des spécifications figées, un contrat agile permet des ajustements réguliers et des fonctionnalités nouvelles.

En tant que développeurs, cela implique que nous sommes en mesure d'adapter le produit tout au long du processus de développement afin de répondre aux besoins changeants du client et de livrer des fonctionnalités de manière progressive.

Ci-dessous, je vais détailler les lignes directrices fournies par le client.

3.1 Gestion des utilisateurs

Afin de garantir la sécurité de l'application et l'intégrité des données, il est requis de mettre en place un système d'authentification. Nous devons également définir des rôles spécifiques pour chaque type d'utilisateur, permettant ainsi de réguler leurs accès et leurs droits.

Trois catégories d'utilisateurs sont nécessaires :

- les **visiteurs** sont des personnes non authentifiées ayant des privilèges limités. Ils ont accès à la carte du monde affichant tous les ramassages terminés ainsi qu'à la page détaillée de chacun. De plus, ils peuvent visualiser les statistiques générées à partir des déchets collectés lors de ces ramassages ;
- les **membres de l'association** sont des personnes connectées ayant également des accès restreints. Outre les privilèges accordés aux visiteurs, ce groupe d'utilisateurs dispose de droits leur permettant de modifier uniquement les données liées à leur(s) mission(s) ;

- les **administrateurs**, utilisateurs connectés, bénéficient d'un accès complet aux fonctionnalités de l'application. Ils sont en mesure de gérer les comptes des utilisateurs et d'interagir avec l'ensemble des données.

3.2 Base de données

Pour stocker les informations de façon évolutive, nous devons définir un schéma de base de données flexible, également appelé modèle conceptuel de données (MCD).

En outre, il est indispensable d'assurer la migration des données existantes vers le nouveau schéma de base de données en veillant à conserver leur intégrité.

3.3 Interface web

Premièrement, les utilisateurs de l'application pouvant accéder à celle-ci depuis un ordinateur, une tablette ou un téléphone, il est impératif que l'interface utilisateur soit responsive[1].

De plus, étant donné que les droits des utilisateurs varient selon leur rôle, il est essentiel que l'interface s'ajuste en fonction des actions qui leur sont autorisées.

Ensuite, les utilisateurs connectés doivent pouvoir saisir, modifier et consulter les données facilement. Cela nécessite la création de formulaires dynamiques et de pages ergonomiques. De plus, ils doivent accéder à une cartographie interactive leur permettant de définir des sites de ramassages et d'y ajouter des données telles que la quantité de déchets collectés.

---

[1] Une interface responsive est capable de s'adapter et d'être fonctionnelle sur différents appareils et tailles d'écrans, offrant ainsi une expérience utilisateur optimale.

Enfin, nous devons mettre en place une visualisation des données permettant d'explorer différents types d’informations grâce une variété de graphiques.

4 Déroulement du stage

4.1 Organisation générale du travail

Chez Natural Solutions, on applique la méthodologie Scrum[2] et on se réunit quotidiennement à 9h15 pour une réunion de synchronisation. Cette réunion a pour objectif principal de suivre l'avancement du projet et de solliciter de l'aide ou des conseils si nécessaire.

Une équipe Scrum est composée des membres suivants :

- un Product Owner : il est chargé d'établir les priorités des fonctionnalités à développer, de les tester, et de maximiser la productivité de l'équipe tout en maintenant un haut niveau de qualité et de satisfaction des clients ;
- une équipe de développement : elle transforme les besoins définis par le Product Owner en fonctionnalités utilisables.

Notre travail est organisé en sprints. Un sprint, d'une durée de 2 semaines chez Natural Solutions, est une période durant laquelle nous nous concentrons sur un ensemble de fonctionnalités définies en amont. À la fin de chaque sprint, nous organisons une revue avec nos clients dans le but de leur présenter les fonctionnalités développées au cours du sprint et de recueillir leurs retours. Pour planifier efficacement chaque sprint, nous utilisons Jira, un outil de gestion de projets permettant d'assigner des tâches à des membres de l'équipe.

D'autre part, pour respecter au mieux les bonnes pratiques de développement collaboratif, j'ai été amenée à travailler avec GitLab[3]. Nous avons créé plusieurs

[2] La méthodologie Scrum est une approche empirique, dynamique et participative de la conduite du projet. Il s'agit d'une méthode agile.

[3] GitLab est une plateforme de gestion du cycle de vie des applications (ALM) qui fournit des fonctionnalités de développement collaboratif.

branches[4] du projet. Chaque fois que je réalisais une tâche, je la mettais sur une nouvelle branche. Lorsque j'avais terminé une tâche et que les changements que j'avais effectués fonctionnaient correctement, je soumettais une demande de fusion[5] à mon maître de stage.

4.2 Technologies utilisées

À mon arrivée dans l'équipe, les choix des technologies étaient déjà établis. Ils étaient basés sur les technologies habituellement utilisées au sein de Natural Solutions.

TypeScript a été choisi comme langage de programmation afin d'améliorer la qualité globale du code et de détecter les éventuelles erreurs liées au type (chaîne de caractères, nombre…) dès la phase de développement.

Pour la base de données, PostgreSQL a été choisie en raison de sa popularité et de son extension PostGIS, qui permet de gérer les données spatiales. En ce qui concerne le stockage des images et des fichiers, le choix s'est porté sur Minio, une solution open source[6] offrant une haute disponibilité des ressources.

---

[4] Une branche est une copie du projet dans laquelle les développeurs peuvent travailler sur des fonctionnalités ou des corrections de bugs sans impacter les autres versions du projet.

[5] Dans le cadre du développement collaboratif, une demande de fusion ou « merge request » est une requête soumise par un contributeur du projet. Elle vise à intégrer les modifications qu'il a effectuées dans une branche spécifique avec la branche principale du projet. L'objectif est d'incorporer ces changements dans la version finale du logiciel.

[6] Une solution open source est un logiciel dont le code source est accessible publiquement, permettant à quiconque de le consulter et de le modifier selon certaines conditions.

Le choix du framework[7] Hasura s'est imposé pour faciliter l'interaction avec la base de données et la communication entre le backend[8] et le frontend[9]. Cette technologie donne accès à une interface graphique permettant de gérer la structure des données, d'exécuter des requêtes GraphQL[10] et de sécuriser les données selon les droits de l'utilisateur. Cette décision a été complétée par l'intégration du framework React Admin pour simplifier la création des pages destinées aux administrateurs. En effet, React Admin permet une interaction fluide avec les requêtes GraphQL fournies par Hasura. Cette combinaison simplifie la récupération et la modification des données, réduisant ainsi la charge de développement en évitant de se focaliser sur la gestion des données.

Pour garantir une interface agréable et cohérente, l'équipe a opté pour la bibliothèque Material UI. Elle fournit des composants préconçus qui rendent l'application esthétique et simple d'utilisation.

Enfin, afin de simplifier le déploiement de l'application, l'équipe a choisi d'utiliser Docker. Cette technologie permet d'encapsuler une application dans un conteneur[11], assurant ainsi sa portabilité et facilitant sa gestion sur différentes plateformes.

---

[7] Un framework est un ensemble d'outils et de composants logiciels organisés conformément à un plan d'architecture.

[8] Le backend est la partie « cachée » d'une application. Il traite les requêtes, gère les données et assure le bon déroulement des opérations nécessaires pour que l'application fonctionne de manière fluide et sécurisée.

[9] Le frontend, également appelé « côté client », désigne la partie visible et interactive d'une application ou d'un site web avec laquelle l'utilisateur interagit directement. C'est l'interface utilisateur, comprenant le design, les éléments visuels et les interactions.

[10] GraphQL est une technologie de requêtes et de manipulation de données, développée par Facebook.

[11] Un conteneur Docker est une instance exécutable d'une image Docker. Une image est une boîte contenant à la fois l'application elle-même et tout ce dont elle a besoin pour fonctionner, comme le système d'exploitation et les bibliothèques.

4.3 Permissions utilisateurs

4.3.1 Contexte

Il existe deux types d'utilisateurs possibles dans l'application :

- l'utilisateur non connecté qui visite le site pour découvrir les missions réalisées par l'association (anonymous) ;
- l'utilisateur connecté qui peut être un super administrateur (admin) ou un utilisateur également membre de l'association mais ayant des privilèges restreints (editor).

Selon son rôle, l'utilisateur a accès à différentes ressources et actions possibles. Limiter l'accès à certaines pages ou éléments de page pourrait sembler suffisant, cependant, cette approche demeure fragile en termes de sécurité. En effet, un utilisateur malveillant pourrait potentiellement interagir avec la base de données en utilisant des outils tels que Postman[12], mettant ainsi en péril la sécurité des données.

4.3.2 Réalisation de la tâche

Afin de pallier cette vulnérabilité, j'ai implémenté un système de gestion des droits pour chaque type d'utilisateur, en utilisant Hasura. Cette solution permet de déterminer si un utilisateur a le droit de lire, ajouter, modifier ou supprimer des données en fonction des autorisations qui lui sont attribuées. Cela permet de garantir que chaque utilisateur a un accès approprié aux informations en fonction de son rôle et de ses responsabilités. Par exemple, pour une table, un utilisateur peut avoir le droit de lire toutes les données, mais seulement de modifier certaines d'entre elles, tandis qu'un autre utilisateur ne pourra que les lire.

Pour mettre en œuvre cette solution, j'ai dû, pour chaque table de la base de données, cocher les autorisations associées à chaque type d'utilisateur. Lors de cette

[12] Postman est un logiciel permettant de créer et de tester des requêtes HTTP. En l'occurrence, il peut être employé pour exécuter des requêtes qui interagissent avec la base de données.

opération, les fichiers détaillant les propriétés de chaque table ainsi que les droits d'accès des utilisateurs, se sont automatiquement mis à jour.

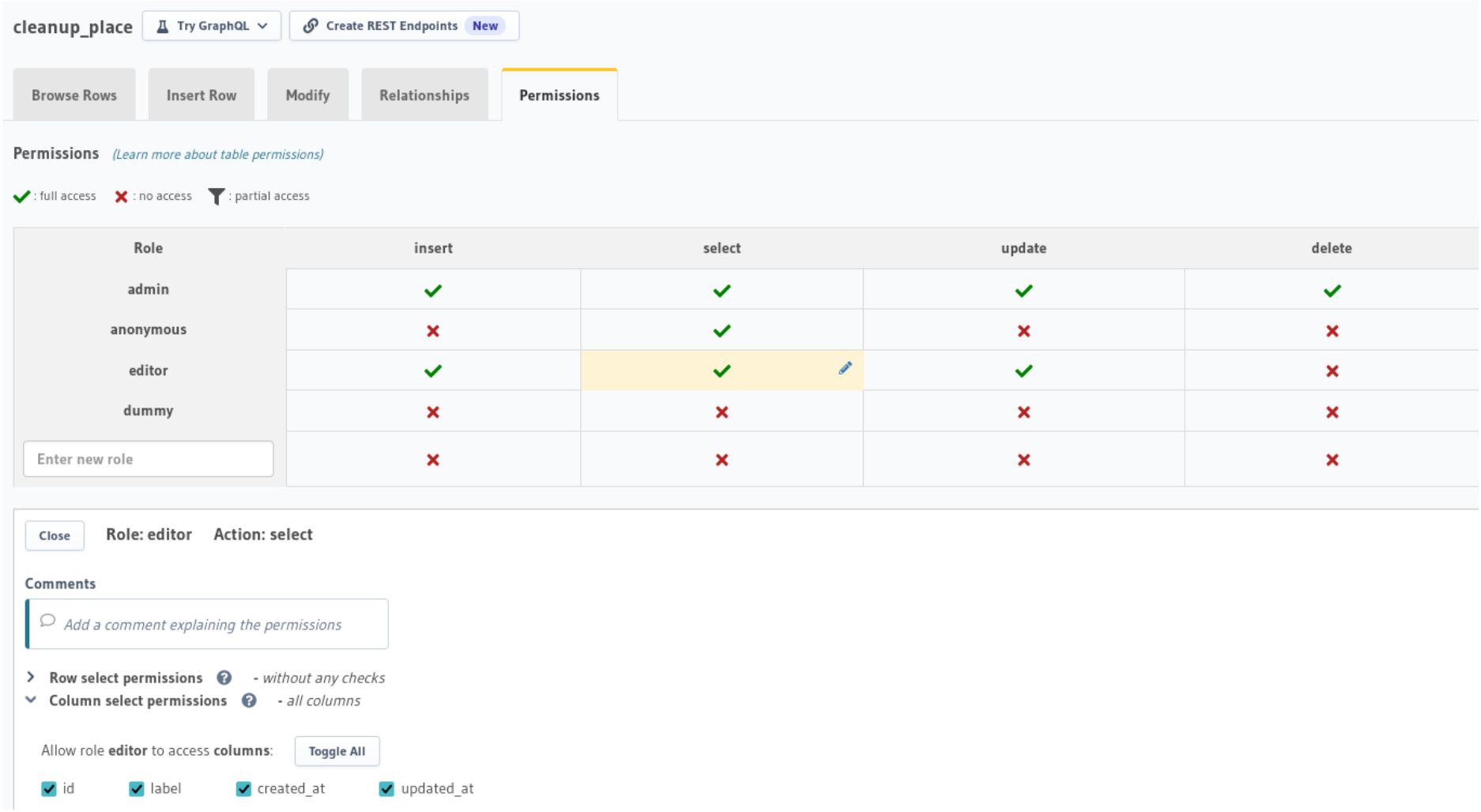


Figure II : Gestion des droits des utilisateurs pour la table « cleanup_place »

Désormais, en cas de tentative malveillante visant à compromettre les données de l'application, l'utilisateur obtiendra un message d'erreur l'informant qu'il n'est pas autorisé, en fonction de son rôle, à effectuer les actions souhaitées. Ce système de gestion des autorisations présente toutefois un inconvénient majeur : il doit être mis à jour à chaque expansion de la structure de la base de données ou ajout de types d'utilisateurs.

Par ailleurs, dans le but d'assurer la cohérence du site et d'éviter des erreurs qui pourraient désorienter l'utilisateur, j'ai veillé à ce qu'il ne puisse voir sur l'interface que les pages et les boutons avec lesquels il est autorisé à interagir.

4.4 Duplication d'un ramassage

4.4.1 Contexte

Chaque mission de ramassages est constituée d'une ou plusieurs collectes (ou ramassages) de déchets. Chaque collecte est principalement définie par :

- une date ;
- des zones précises, où les participants concentrent leurs efforts de nettoyage ;
- une mission ;
- un niveau de caractérisation lié au degré de dangerosité ou à l'impact environnemental des déchets ciblés ;
- des partenaires ;
- des photos ;
- un point de départ où les participants se rassemblent.

L'objectif de cette tâche est de permettre à l'utilisateur de créer une copie d'un ramassage, tout en lui offrant la possibilité de modifier des informations sans altérer le ramassage original. Cette fonctionnalité vise à économiser du temps, notamment en évitant aux utilisateurs de recréer plusieurs fois les mêmes zones.

4.4.2 Réalisation de la tâche

J'ai d'abord ajouté un bouton « Dupliquer » sur la page d'un ramassage. Celle-ci est accessible à tous les utilisateurs, mais seuls ceux connectés ont accès à ce bouton. Lorsque l'utilisateur est redirigé sur une autre page, après avoir cliqué sur le bouton « Dupliquer », j'ai envisagé deux approches : soit coder la création de cette page, soit reprendre une page existante. Après en avoir discuté avec mon maître de stage, nous avons choisi la seconde option pour mutualiser le code et maintenir la cohérence de l'expérience utilisateur.

J'ai donc réutilisé la page de création d'un ramassage. Quand l'utilisateur clique sur « Dupliquer », le chemin (ou URL) de la page où il se trouve est modifié pour être remplacé par le chemin d'accès de la page de création d'un ramassage.

Sachant que la page de création d'un ramassage est utilisée à la fois pour la création et la duplication, il est impératif de distinguer les deux cas de figures. Dans le cas de la duplication, on ajoute à l'URL l'identifiant du ramassage dont on veut faire la copie, dans l'autre cas, ce paramètre n'est pas renseigné.

Dans la page de création, je vérifie si l'identifiant d'un ramassage est présent. Si c'est le cas, j'envoie une requête à la base de données pour récupérer les informations de ce ramassage. J'injecte ensuite les données du ramassage à dupliquer dans les champs du formulaire correspondant. L'utilisateur est alors libre de modifier les informations selon ses besoins, sans affecter le ramassage original. En revanche, conformément au souhait du client, les photos ne sont pas préremplies.

Lors de mes tests, j'ai découvert un problème majeur : la modification des coordonnées des zones du ramassage original ou de sa copie après duplication entraînait une modification des deux ramassages. Après avoir examiné les informations de la base de données, j'ai constaté que chaque zone était associée à un identifiant unique. Lors de la duplication d'un ramassage, les zones non modifiées du formulaire prérempli conservaient les mêmes identifiants que celles du ramassage

original. Pour résoudre ce problème, j'ai modifié le processus de duplication en générant aléatoirement de nouveaux identifiants pour chaque zone.

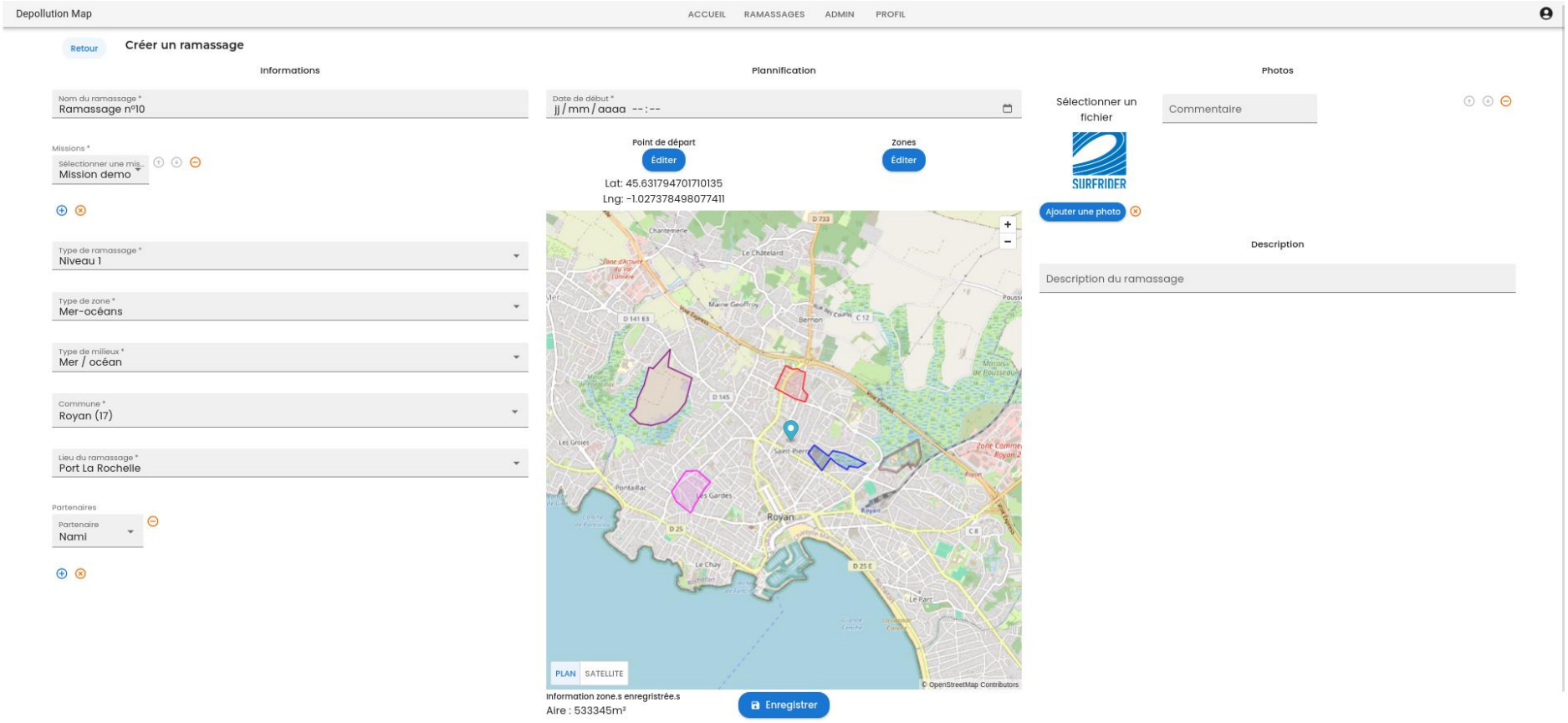


Figure III : Formulaire de création d'un ramassage prérempli avec les informations du ramassage à dupliquer

4.5 Historique de ramassage

L'objectif de cette tâche est de garder, pour chaque ramassage, un historique de toutes les collectes réalisées ou prévues au même endroit.

4.5.1 Indiquer un lieu

Afin de mettre en place cette fonctionnalité, j'ai commencé par implémenter un système permettant d'indiquer le lieu d'un ramassage.

4.5.1.1 Modification de la base de données

J'ai d'abord créé une migration Hasura[13] pour ajouter une table dans la base de données. Celle-ci est destinée à conserver les noms de tous les emplacements.

Ensuite, j'ai établi un lien entre ma nouvelle table et les ramassages. Étant donné qu'une collecte est liée à un seul lieu et qu'un lieu peut être associé à un ou plusieurs ramassages, j'ai ajouté, dans la table des ramassages, une colonne ou « clé étrangère ». Les valeurs de cette colonne font référence à une ligne de la table des lieux.

Une fois le backend terminé, je me suis concentrée sur la modification du frontend.

4.5.1.2 Modification de l'interface utilisateur

Pour permettre aux utilisateurs connectés de spécifier le lieu d'un ramassage, j'ai adapté les formulaires de création et de modification d'une collecte pour y intégrer une liste déroulante de tous les lieux enregistrés dans la base de données. Pour

[13] Une migration Hasura est un processus de mise à jour de la structure de la base de données gérée par Hasura. Elle permet aux autres collaborateurs d'accéder aux nouvelles fonctionnalités ou données ajoutées à la base de données après la mise à jour.

générer cette liste, j'ai utilisé le composant « SelectInput » de la bibliothèque React Admin. Il m'a suffi de spécifier le nom de la table des lieux et React Admin s'est chargé de récupérer les informations dans la base de données.

À la demande du client, j'ai intégré la possibilité, pour tous les utilisateurs connectés, d'ajouter un lieu directement depuis le formulaire de création d'un ramassage. Pour cela, j'ai réutilisé la liste déroulante des lieux existants. Le composant « SelectInput » offre, en effet, la possibilité d'afficher une notification interactive (ou popup) dès que l'utilisateur sélectionne l'option « Créer un lieu ». À partir de cette notification, l'utilisateur peut saisir le nom du lieu qu'il souhaite créer. Une fois la création confirmée, React Admin se charge d'actualiser la table des lieux.

Du côté des administrateurs, j'ai ajouté une rubrique « Lieu de ramassage » dans le menu. Cette section mène sur une page où sont affichés tous les emplacements. Pour leur donner la possibilité de créer ou de modifier des lieux, j'ai implémenté les formulaires adéquats. Une fois de plus, il m'a suffi de renseigner pour chaque champ le nom des colonnes associées dans la table des ramassages et React Admin s'est occupé du reste.

4.5.2 Affichage de l'historique

Par la suite, j'ai écrit une requête GraphQL destinée à interroger la base de données afin de récupérer, pour le ramassage souhaité, l'historique des collectes ayant eu lieu au même endroit.

Pour afficher cet historique, j'ai modifié la page d'un ramassage en me conformant aux maquettes conçues par la designeuse. Afin de faciliter la consultation des ramassages présents dans l'historique, j'ai permis à l'utilisateur de cliquer sur chacun d'eux, ce qui permet de le rediriger automatiquement vers la page de celui-ci. Selon le souhait du client, j'ai limité l'affichage de l'historique à trois ramassages. Dans le but de compléter l'information disponible, j'ai inclus un bouton « Tout afficher » qui ouvre une popup contenant la liste de tous les ramassages réalisés au même

emplacement, classés en deux catégories : ceux déjà réalisés et ceux à venir par rapport à la date actuelle.

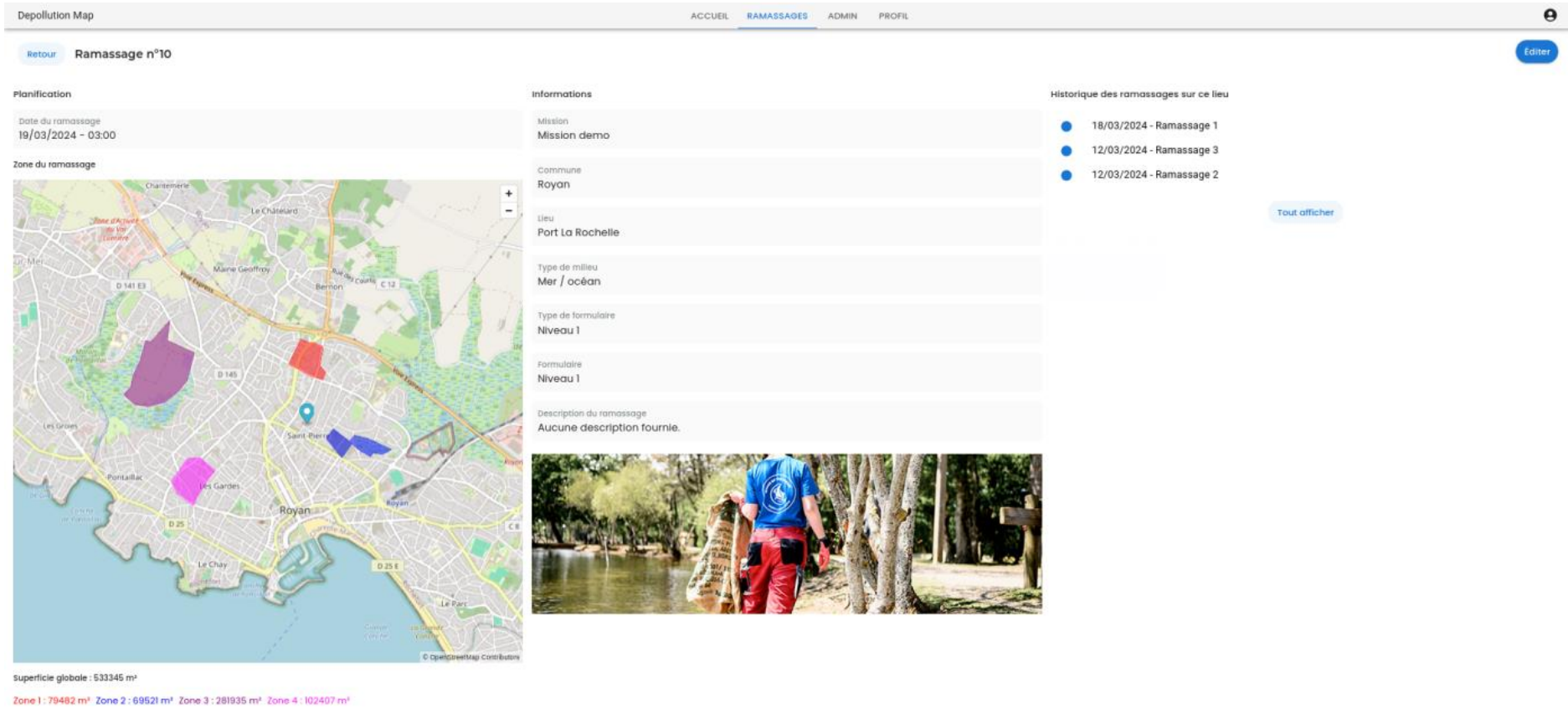


Figure IV : Page d'un ramassage et historique de celui-ci

4.6 Insertion de données CSV en base de données

4.6.1 Contexte

Parmi les fonctionnalités de l'application, les utilisateurs connectés doivent pouvoir saisir, pour chaque ramassage, la quantité, le poids et le volume des déchets collectés. Ces données seront par la suite utilisées pour la réalisation de statistiques. Pour répondre à ce besoin, il était essentiel d'enregistrer dans la base de données existante tous les déchets pouvant être ramassés. À cet effet, notre client nous a fourni un fichier CSV indiquant pour chaque déchet sa catégorie et son niveau de caractérisation basé sur son degré de dangerosité ou son impact environnemental.

J'ai été chargée de développer un script capable de lire ce fichier et d'importer les données de manière adéquate dans la base de données. Notons qu'il est nécessaire d'exécuter manuellement ce script à chaque actualisation du fichier CSV par le client.

4.6.2 Réalisation de la tâche

Pour atteindre mon objectif, j'ai développé ce script dans un fichier externe à l'application afin qu'il ne s'exécute que sur demande.

La première étape a donc consisté à établir une connexion entre le script et la base de données grâce au module « pg » de Node.js.

Ensuite, j'ai récupéré les données du fichier. Pour cela, j'ai utilisé le module « fs » de Node.js. Le résultat se présentait sous la forme d'un tableau comportant le même nombre de lignes que le fichier CSV. En revanche, les données des diverses colonnes étaient concaténées dans une même colonne mais elles restaient identifiables du fait qu'elles étaient séparées par des virgules. En vue de reconstituer les colonnes du fichier, j'ai envisagé de scinder chaque ligne au niveau du caractère virgule. Cependant, les colonnes du fichier CSV pouvaient aussi contenir des virgules au niveau desquelles il ne fallait pas faire de séparation. Étant donné qu'après la

lecture du fichier CSV par le module « fs » ces virgules étaient encadrées de guillemets il était aisé de les ignorer en ayant recours à une expression régulière[14].

Enfin, le résultat de l'extraction faite à partir du fichier CSV fourni par le client doit être intégré dans trois tables distinctes de la base de données. Il est important de noter que cette étape doit suivre un ordre précis pour éviter des erreurs liées au fait que certaines tables vont chercher des informations dans d'autres tables.
Afin de suivre les bonnes pratiques de développement, certaines colonnes des tables de la base de données, comme celles des identifiants, ont une nomenclature spécifique. Lors de l'importation, j'ai donc veillé à normaliser les données destinées à ces colonnes en éliminant les accents et autres caractères spéciaux.
D'autre part, certaines colonnes nécessitent un format de données spécifique, tel que le booléen (vrai ou faux). Cependant, dans les résultats de l'extraction à partir du fichier CSV, nous trouvons « x » pour vrai et une case vide pour faux. Par conséquent, j'ai vérifié si les cases correspondantes à ces colonnes étaient remplies par « x » et je leur ai attribué les valeurs « true » ou « false ».

4.7 Association d'un ramassage avec une ville

Suite à la demande du client, chaque ramassage doit pouvoir être lié à une ville située en France métropolitaine ou outre-mer. De plus, il est requis que les départements et les régions soient également spécifiés.

Pour accomplir cette tâche, j'ai tout d'abord effectué une recherche sur Internet afin de trouver un fichier contenant toutes les régions, départements et villes du pays. J'ai eu la chance de trouver un fichier au format SQL adapté à mes besoins.

Cependant, ce fichier présentait un grand nombre d'informations superflues, ce qui rendait leur suppression manuelle impossible. L'idée est donc d'importer le fichier

[14] Une expression régulière ou regex est un motif de recherche utilisé pour identifier et manipuler des chaînes de caractères selon des règles spécifiques.

tel quel dans la base de données et ensuite de supprimer les informations inutiles. Il se trouve que l'outil DBeaver offre la possibilité d'exporter le contenu d'une base de données de façon sélective.

Dans ce but, j'ai utilisé DBeaver pour accéder à la base de données de l'application et exécuter les requêtes SQL du fichier. Cette approche a permis de créer et de remplir les tables des régions, des départements et des villes. Ensuite, j'ai extrait les colonnes pertinentes de ces tables et je les ai exportées au format SQL dans le but d'insérer ultérieurement leur contenu dans la base. L'extraction ainsi réalisée et les tables précédentes n'ayant pas la même structure, il a fallu les supprimer puis les recréer. L'exécution des requêtes SQL du fichier exporté a permis d'alimenter les nouvelles tables.

À ce stade, la table des villes était liée à celle des départements elle-même liée à celle des régions. Ensuite, j'ai créé une liaison entre la table des villes et celle des ramassages. Étant donné qu'un ramassage est associé à une seule ville tandis qu'une ville peut accueillir plusieurs ramassages, j'ai ajouté une colonne dans la table des ramassages dont les valeurs font référence à une ligne spécifique de la table des villes. Afin que mes collègues puissent accéder à ces modifications dans la base de données, j'ai généré une migration Hasura pour chaque changement effectué.

Après avoir réalisé les modifications dans la base de données, j'ai mis à jour l'interface utilisateur. Sur la page de création et de modification d'un ramassage, j'ai ajouté une liste déroulante contenant toutes les villes de France, afin que l'utilisateur puisse en sélectionner une. Comme la liste compte plus de 35 000 villes, j'ai implémenté un système d'auto-complétion permettant aux utilisateurs de trouver la ville recherchée en saisissant quelques lettres de son nom. Du côté administrateur, j'ai apporté des modifications à la page répertoriant tous les ramassages afin d'inclure pour chacun, la ville et le département auxquels ils sont associés. Enfin, j'ai modifié la page d'un ramassage, accessible à tous les utilisateurs, pour y afficher le nom de la ville.

4.8 Carte de la page d'accueil

4.8.1 Contexte

Chaque ramassage organisé par Wings of Ocean est associé à un formulaire de caractérisation. Il permet aux participants de saisir les informations relatives aux déchets collectés, telles que le nombre, le poids et le volume. Les formulaires de caractérisation peuvent être dans l'un des quatre états suivants : à faire, en cours, en attente de validation ou validé par un administrateur. Lorsqu'un formulaire est validé, cela signifie que les données sont considérées comme correctes et que le ramassage est terminé.

Les ramassages terminés sont accessibles à tous les utilisateurs, même à ceux qui ne sont pas connectés. Chaque utilisateur peut consulter en détail les informations concernant ces ramassages. Dans cette partie, mon travail a consisté à afficher les collectes terminées sur la carte et à fournir aux utilisateurs la possibilité de les filtrer en fonction :

- de la date de réalisation ;
- de la localisation (ville, département, région) ;
- des missions ;
- du niveau de caractérisation ;
- des partenaires.

4.8.2 Affichage des ramassages

La première étape a été d'extraire tous les ramassages ayant un formulaire de caractérisation validé par un administrateur et de les stocker dans un tableau. Pour ce faire, j'ai rédigé une requête GraphQL qui parcourt tous les ramassages. Pour chacun, elle examine le formulaire de caractérisation associé et vérifie si son statut est terminé. Si tel est le cas, la requête retient le ramassage.

À mon arrivée dans l’équipe, la carte du monde était déjà affichée. J’ai donc positionné chaque ramassage sur cette carte en utilisant leur point de départ.

Initialement, j'avais utilisé des marqueurs pour les représenter, mais pour rendre la carte plus lisible lorsque plusieurs collectes étaient concentrées au même endroit, mon maître de stage m'a conseillé de regrouper ou de séparer les marqueurs en fonction du niveau de zoom. Cette fonctionnalité est appelée la clusterisation. Pour répondre à cette consigne, j'ai cherché des informations sur la bibliothèque que j'avais utilisée pour les marqueurs, à savoir « react-map-gl ». Elle dispose d'une documentation très complète ce qui m'a permis de réaliser rapidement cette tâche. Dans un souci de compréhension, j'ai pris soin d'afficher le nombre de collectes présentes dans chaque groupe de ramassages (ou cluster).

Afin de simplifier l'expérience utilisateur, j'ai fait en sorte que la carte se recentre et s'agrandisse automatiquement lorsqu'il clique sur un cluster. Quand le niveau de zoom est suffisant, les ramassages contenus dans le cluster apparaissent distinctement.

D'autre part, si un utilisateur clique sur un ramassage, une popup s'affichera à côté, indiquant son nom, le lieu où il se déroule, la date de sa réalisation et une photo s'il en contient. Pour accéder à des informations plus complètes, l'utilisateur peut cliquer sur un bouton le redirigeant vers la page de ce ramassage. J'ai réalisé cette popup avec la bibliothèque « Material UI ».

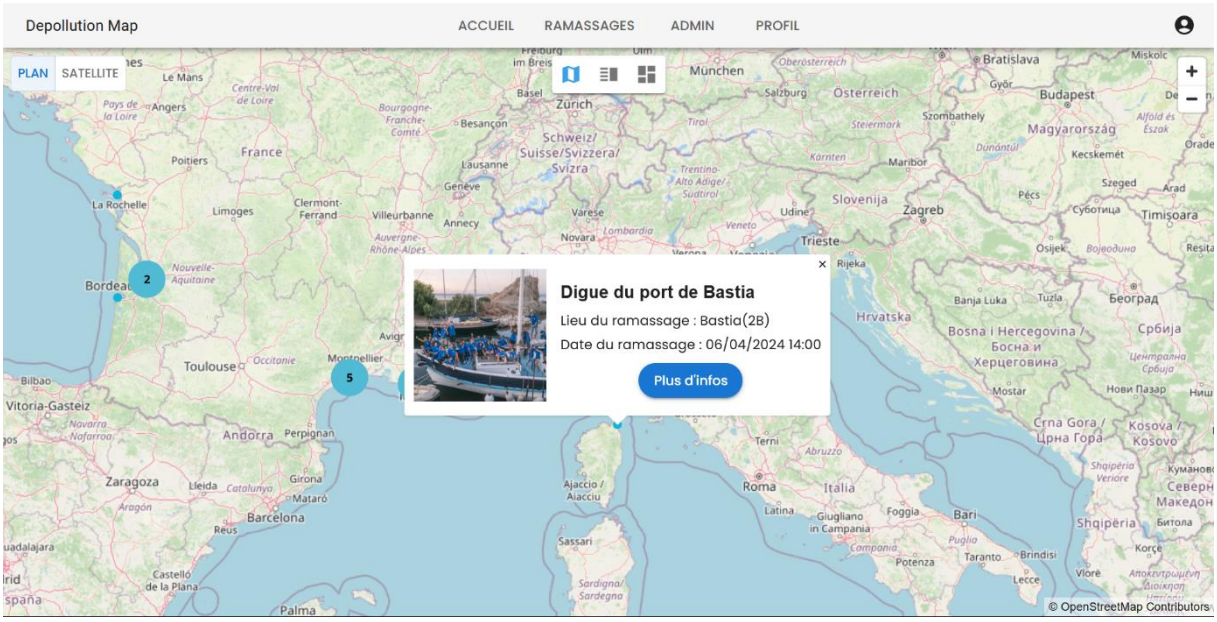


Figure V : Clusterisation des collectes et popup d'un ramassage

4.8.3 Mise en place des filtres

Enfin, j'ai ajouté un bouton « Filtres », sur la page d'accueil, qui déclenche l'ouverture d'une popup. Puisque les filtres étaient destinés à être utilisés dans d'autres pages, j'ai décidé de regrouper le code de cette popup dans un fichier réutilisable par toutes ces pages.

Pour commencer, j'ai élaboré des requêtes pour extraire de la base de données chaque mission, partenaire, ville, département, région et type de ramassages. Ensuite, en utilisant la bibliothèque « Material UI », j'ai formaté cette popup selon les maquettes conçues par la designeuse et j'ai rempli les champs avec les données précédemment extraites.

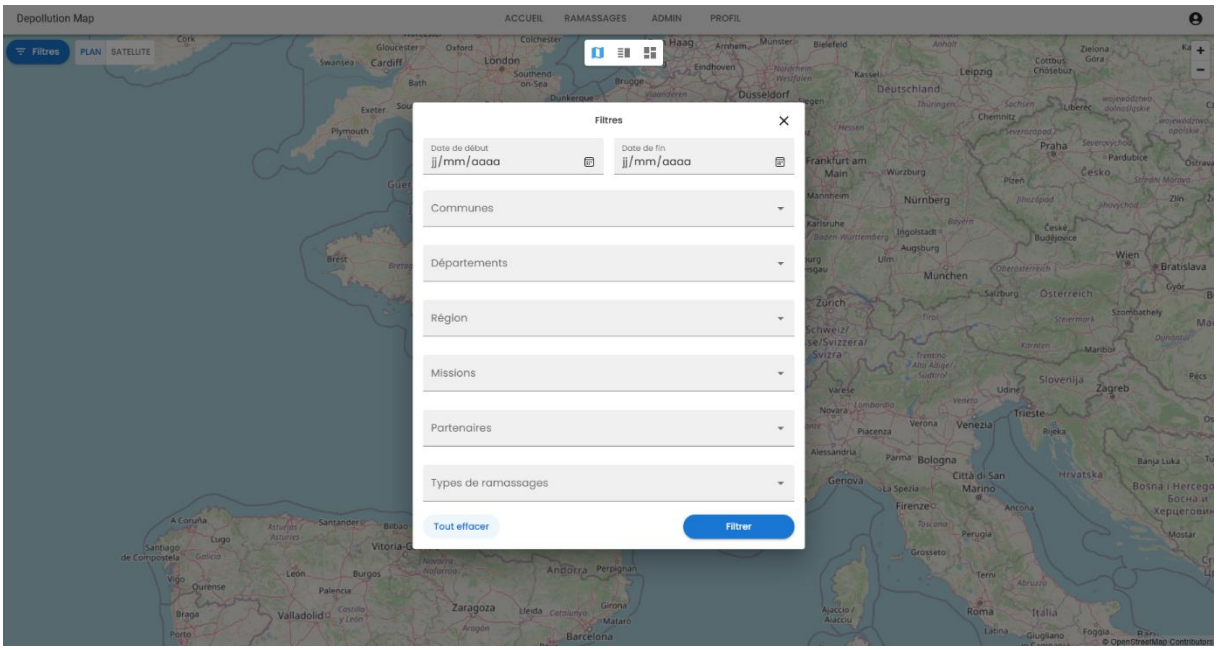


Figure VI : Popup des filtres

Lorsque l'utilisateur valide des critères de recherche (ou filtres), ils sont intégrés à l'URL. Comme dit précédemment, tous les ramassages terminés ont été extraits de la base de données et stockés dans un tableau. L'objectif étant d'accélérer le processus de développement et d'avoir un côté instantané, nous avons choisi d'appliquer les filtres présents dans l'URL à ce tableau. Ce traitement se fait du côté client, c'est-à-dire dans le navigateur. Le résultat est enregistré dans un autre tableau

afin d'afficher les ramassages correspondants sur la carte. La sélection de nouveaux filtres entraîne l'actualisation de ce deuxième tableau. Ce procédé est généralement déconseillé car il peut prendre beaucoup de temps si les données à traiter dans le navigateur sont trop volumineuses. Dans le cas présent, nous pouvions nous le permettre car nous savons que le nombre de ramassages ne va pas augmenter de façon fulgurante.

Mettre les critères de recherche souhaités par l'utilisateur dans l'URL offre l'avantage de conserver le résultat de la recherche même si la page est actualisée ou que l'on navigue entre différentes pages. De plus, elle facilite le partage du lien de la carte filtrée avec un collègue, évitant ainsi à ce dernier de devoir appliquer les mêmes critères de recherche.

4.9 Dashboard

4.9.1 Contexte

Le client nous a demandé de créer un tableau de bord basé sur les ramassages terminés. Celui-ci doit comporter les mêmes filtres que la carte.

Sa seconde demande est que lorsque l'utilisateur arrive sur la page d'accueil il puisse choisir le type de vue à afficher. La sélection se fait entre la carte, le dashboard et une vue mixte. Dans les deux dernières options, le dashboard apparaît sous une forme plus ou moins détaillée.

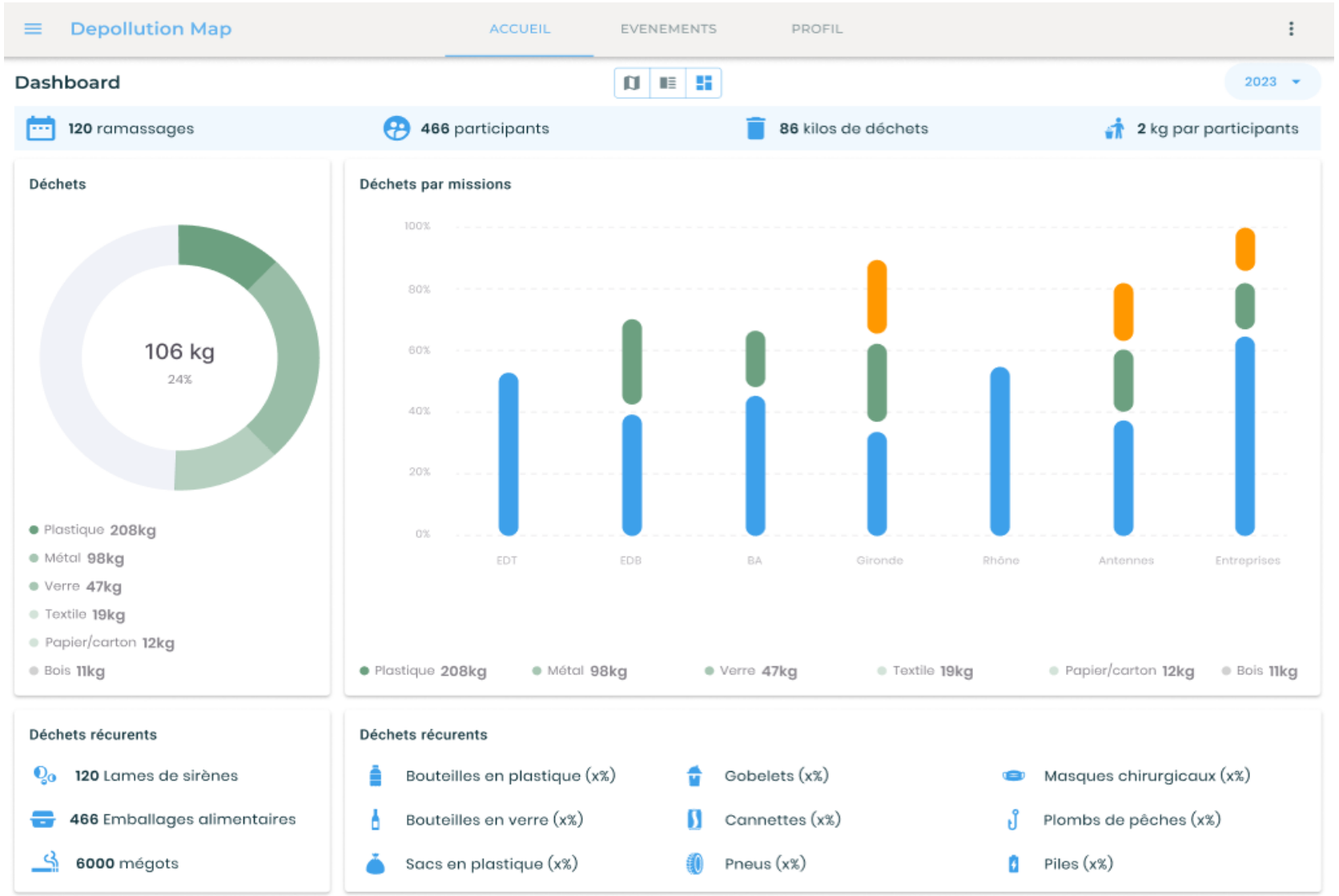


Figure VII : Maquette de la vue du dashboard

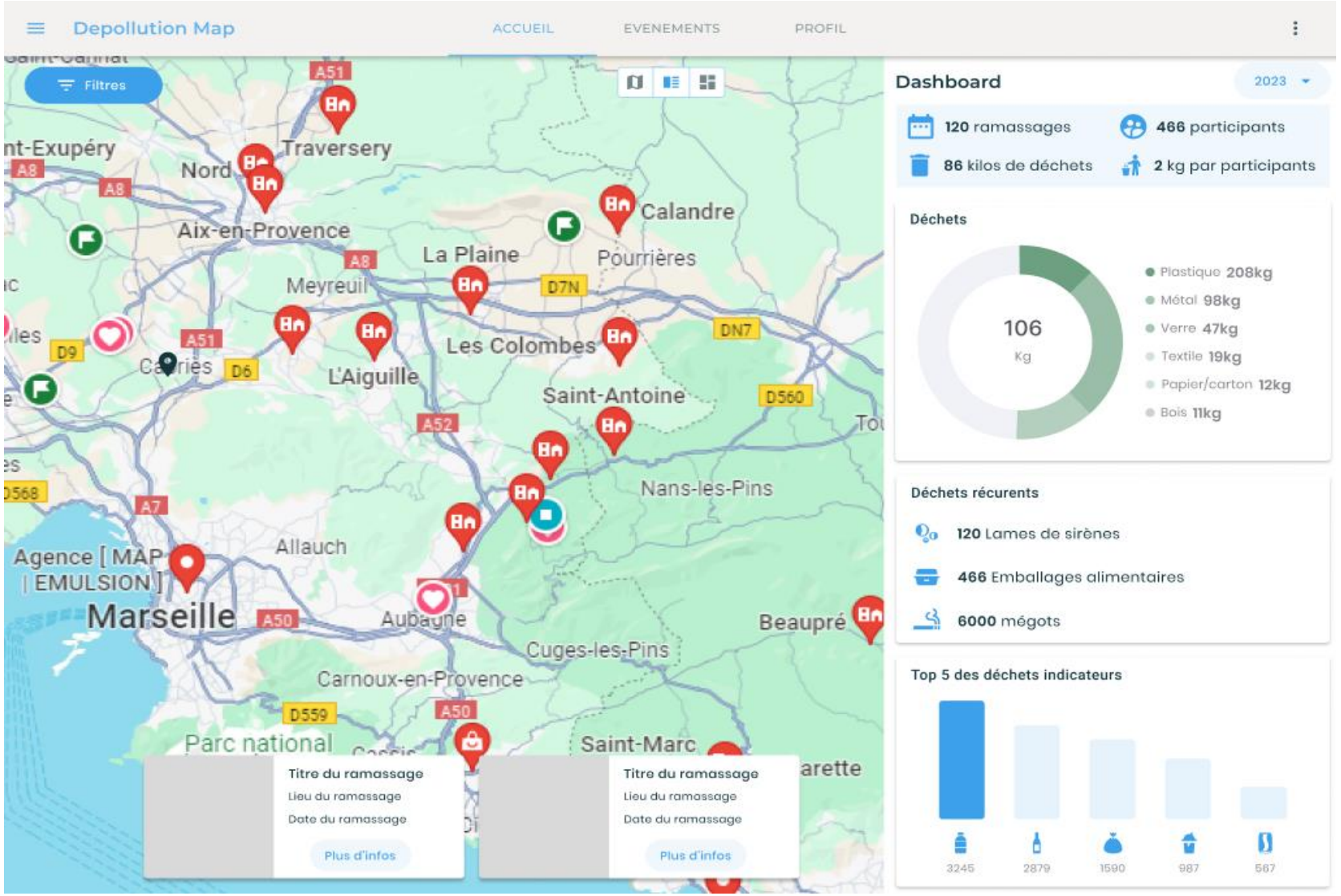


Figure VIII : Maquette de la vue mixte

4.9.2 Réalisation de la tâche

Dans cette partie, je vais expliquer comment j'ai mis en place ces deux versions du tableau de bord, chacune devant réagir aux filtres.

Comme mentionné précédemment, tout ramassage dont la caractérisation est validée est considéré comme terminé. Ses données sont alors utilisées pour générer des statistiques.

Dans un premier temps, j'ai écrit des requêtes GraphQL permettant de réaliser les statistiques des ramassages souhaités. J'ai testé ces requêtes dans l'interface graphique proposée par Hasura. Lors de cette étape, j'ai rencontré des problèmes pour les données du graphique des déchets par mission (cf. Figure VII). Au lieu, de récupérer les données par mission, je les récupérais par ramassage. Avant d'afficher ce résultat, j'aurais dû parcourir chaque ramassage obtenu, extraire sa mission et incrémenter pour celle-ci la quantité de déchets récoltée pour les catégories « plastique », « métal », « verre », « textile », « papier et carton » et « bois ». Comme cette façon de procéder n'était pas optimale et que je ne pouvais pas contourner ce problème avec le langage GraphQL, j'ai choisi de créer une vue[15] dans la base de données puis de l'interroger avec une requête GraphQL.

J'ai ensuite ajouté une route spécifique au dashboard dans le backend. Cette route prend en paramètre les identifiants des ramassages souhaités afin de les transmettre aux requêtes GraphQL précédemment écrites.

Côté frontend, au chargement initial de la page, j'ai réutilisé la requête écrite dans la carte afin de récupérer tous les ramassages terminés et je les ai stockés dans un tableau. Ensuite, j'applique les filtres définis dans l'URL à ce tableau et je range le

[15] Une vue en SQL est une représentation d'une ou plusieurs tables dans une base de données. Elle est créée à partir d'une requête SQL qui extrait et transforme les données de ces tables selon les besoins spécifiques. Les vues permettent de simplifier les requêtes complexes, de masquer la complexité de la structure des données et de limiter l'accès aux données sensibles en fournissant une interface de consultation restreinte.

résultat dans un second tableau. Enfin, j’ai fait appel à la route du backend en lui transmettant les identifiants des ramassages filtrés. Comme pour la carte, lorsque de nouveaux filtres sont sélectionnés (que ce soit via la vue mixte ou la vue du dashboard), le tableau de ramassages filtrés se met à jour, déclenchant ainsi un nouvel appel au backend.

Pour l’interface utilisateur, j'ai utilisé la bibliothèque Material UI ainsi que le template Materio. Il s’agit d’un modèle d'interface sous forme de projet, offrant des pages Web prêtes à l'emploi, que l'on peut intégrer et adapter à nos besoins. Pour utiliser ces composants, j'ai extrait le code du template et l'ai intégré à DepollutionMap. Je l’ai ensuite légèrement modifié pour afficher les résultats renvoyés par le backend de façon dynamique.

5 Bilan

5.1 Bilan humain

Dès mon arrivée dans l'équipe, j'ai été chaleureusement accueillie. Chacun de mes collègues a été bienveillant et a pris le temps de m'expliquer les différentes tâches et responsabilités qui m'étaient confiées.

Les autres membres de l'entreprise ont également été très présents lorsque j'en avais besoin. Cependant, j'ai réalisé que j'aurais pu interagir davantage avec eux afin d'apprendre à mieux les connaître et à m'informer sur leur rôle au sein de Natural Solutions.

5.2 Bilan concernant mon travail

Chaque fois que j'ai rencontré des problèmes au cours de mon travail, j'ai envisagé des solutions que j'ai pu exposer à mon maître de stage. Cette démarche m'a permis d'exploiter ses explications et commentaires et de m'améliorer sur certaines notions. Toutefois, pour éviter de déranger mon tuteur, il m'est arrivé de consacrer trop de temps à mes recherches, ce qui s'est avéré peu productif.

En ce qui concerne les outils utilisés, je pense avoir su m'adapter et me familiariser avec ceux que je ne connaissais pas, tels que React Admin et Hasura.

5.3 Bilan de compétences

Durant ces trois mois consacrés au développement de DepollutionMap, j'ai eu la possibilité d'approfondir mes connaissances en SQL, d'acquérir des connaissances sur le langage de requêtes GraphQL et de découvrir le framework Hasura. De plus, j'ai pu développer une meilleure maîtrise de Git en l'utilisant de manière avancée pour la

gestion des versions du code, ce qui a renforcé ma compréhension des bonnes pratiques de développement collaboratif.

Grâce à cette expérience enrichissante, j'ai consolidé mes compétences techniques et élargi mes connaissances dans le domaine du développement Web. J'ai également découvert les différents acteurs impliqués dans un projet informatique et mesuré l'importance de la communication entre eux. Les échanges avec le client m'ont aussi permis d’apprendre à identifier ses besoins spécifiques et à les intégrer dans le processus de développement, contribuant ainsi à une vision plus globale du projet dans son ensemble.

Conclusion

Lors de mon stage, j'ai eu l'opportunité de travailler sur une plateforme de gestion de déchets, DepollutionMap. Mon objectif était de contribuer activement au développement de celle-ci. Je suis satisfaite d'avoir appréhendé cette mission sous un angle approprié et d'avoir su répondre aux besoins exprimés. J'ai démontré ma capacité à envisager tous les cas de figures possibles et à mettre en œuvre des solutions concrètes.

Toutefois, étant donné que le projet devrait se terminer un mois après mon départ, des ajustements seront apportés par mes collègues. De plus, compte tenu des besoins évolutifs des utilisateurs de cette plateforme, de nouvelles fonctionnalités pourraient être ajoutées dans les années à venir.

Ce stage a été une excellente complémentarité à mes trois années de formation en BUT Informatique. Non seulement j'ai pu mettre en pratique les connaissances théoriques que j'avais acquises jusqu'à présent, mais j'ai également eu la possibilité d'acquérir des compétences essentielles en milieu professionnel. J'ai travaillé en collaboration avec d'autres développeurs, ce qui m'a permis d'améliorer mes compétences en travail d'équipe et de comprendre l'importance de la communication dans un projet de développement logiciel.

Je garderai un excellent souvenir de cette expérience, que ce soit pour les enseignements humains et professionnels qu'elle m'a apportés.

Sitographie

Natural Solutions, site officiel. Disponible sur : https://www.natural-solutions.eu/produits

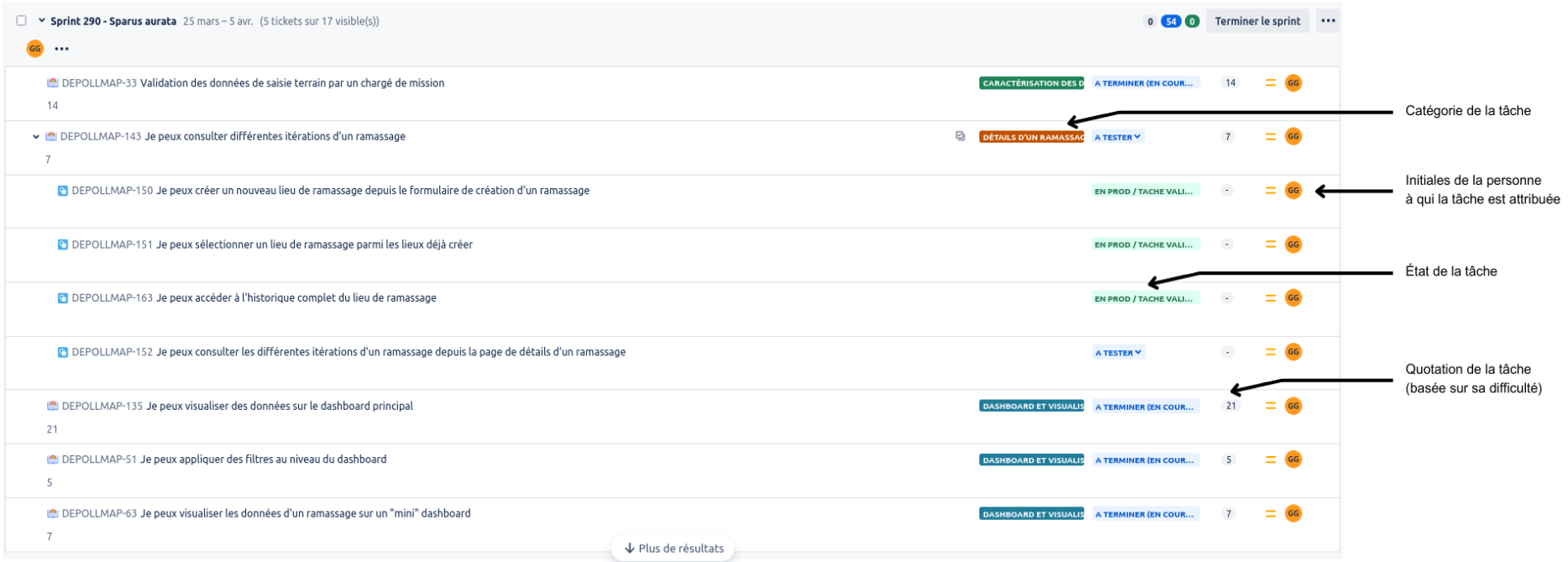
Documentation embarquement Natural Solutions. Disponible sur : https://natural-solutions.gitlab.io/nsdoc/

« Holacratie : définition et exemples » - Journal du Net. Disponible sur : https://www.journaldunet.fr/management/guide-du-management/1204818-holacratie-definition-exemples/

« Méthodologie Scrum » - Chef de Projet. Disponible sur : https://urlz.fr/m4Lr

Table des illustrations

Tables des annexes



Annexe I : Planification des tâches d'un sprint avec Jira

RAPPORT MENSUEL

Résultats des ramassages

Lieu	Date
Mole Saint Louis	8/7/2023
Marché Oriental île de thau	6/7/2023
Quai de Sète (Mégots)	7/7/2023
Quai de Sète (Mégots)	07/07/2023
NB ramassages	3

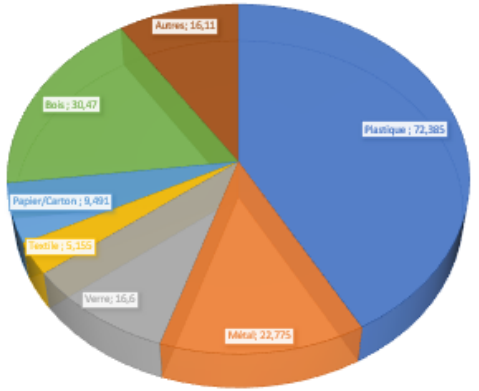
Mole Saint Louis	POIDS	QUANTITÉ	VOLUME
PLASTIQUE	7,115	327	115
METAL	3,085	148	150
VERRE	11,5	57	10
TEXTILE	0,355	0	5
PAPIER CARTON	4,191	0	120
BOIS	11,6	0	40
AUTRES	0,54	603	5
MEGOTS		600	

Marché Oriental île de thau	POIDS	QUANTITÉ	VOLUME
PLASTIQUE	65,27	1113	1005
METAL	19,69	83	85
VERRE	5,1	23	40
TEXTILE	4,8	0	50
PAPIER CARTON	5,3	0	150
BOIS	18,87	0	50
AUTRES	15,57	779	55
MEGOTS		750	

Résultats Mois de Juillet

POIDS TOTAL	Poids	Quantité	Volume
Plastique	72,385	1440	1120
Métal	22,775	211	235
Verre	16,6	80	50
Textile	5,155	0	55
Papier/Carton	9,491	0	270
Bois	30,47	0	90
			0
Autres	16,11	1382	60
POIDS TOTAL	**172,986**	**3113**	**1880**

MÉGOTS RAMASSÉS	1350
Ramassages mégots	
07/07/23 Quai de Sète	4500
15/07/23 Quai de Sète	4600
TOTAL mégots	10450



DÉCHETS INDICATEURS

	Mole Saint Louis	Marché Oriental île de thau
BOUTEILLE VERRE	57	23
CANETTE ALU	108	59
Pneu		4
CAPRISUN	1	15
Gobelet plastique	26	65
Coton tige plastique		
Bouchon plastique	55	102
Paille en plastique	8	3
Fil de pêche		
Boite d'appat	3	1
Emballage Alimentaire	97	576
Briquet	2	9
Média filtrant		

Déchets indicateurs	Quantité
Bouteilles en verre	80
Canettes en aluminium	167
Pneu	4
Caprisun	16
Gobelets en plastique	91
Cotons tiges en plastique	0
Bouchons en plastique	157
Pailles en plastique	11
Fils de pêche	0
Boîtes d'appâts	4
Emballages alimentaire	673
Briquets	11
Médias filtrants	0

Annexe II : Réalisation des statistiques avant la mise en place du dashboard

id	label	rubbish_category_id	is_support_brand	is_support_doi	QTE1	QTE2	QTE3	WEIGHT1	WEIGHT2	WEIGHT3
oyster_equipment	Matériels ostréicoles	oyster_farming				x				
storage_tubs	Bacs de stockage	oyster_farming					x			
oyster_cups	Coupelles ostréicoles	oyster_farming					x			
plastic_bottles	Bouteilles en plastique	plastic	x	x	x	x	x	x	x	x
surgical_masks	Masques chirurgicaux	plastic			x	x	x			
plastic_straws	Pailles en plastique	plastic			x	x	x			
plastic_bags	Sacs en plastique	plastic			x	x	x			
mermaid_tears	Larmes de sirène	plastic			x	x	x	x	x	x
plastic_miscellaneous	Divers plastique (Morceaux, déchets indicateurs etc..)	plastic						x	x	x
e-cigarettes_and_vaping_products	E-cigarettes & produits de vapotage	smoking_vaping	x	x	x	x	x			
butts	Mégots	smoking_vaping	x		x	x	x			
smoking_items	Tabagisme (Paquets, feuilles, filtres)	smoking_vaping	x	x	x	x	x			
fishing_gear	Matériel de pêche (Fil, bouée, canne, épuisette)	fishing				x				
bait_boxes	Boîtes d'appâts	fishing	x	x		x	x			
buoys	Bouées	fishing					x			
fishing_line	Fils de pêche (nylon)	fishing					x			
large_fishing_gear	Gros matériel de pêche (Canne à pêche, épuisette ...)	fishing					x			
disposable_dishes	Vaisselles à usage unique	aperitif			x	x	x			
plastic_cups	Gobelets en plastique	aperitif					x			
hulls_boat_ends	Coques, bouts de bateaux	water_sport					x			
life_jackets	Gilets de sauvetage	water_sport					x			
fenders	Pare-battages	water_sport					x			

Annexe III : Fichier CSV à insérer dans la base