

Rapport de projet

Mise en œuvre d'une application

De gestion des lieux favoris

ENCADRÉ PAR

Mr. Loiseau Yannick

Mr. Guillon Bruno

Réalisé par :

- OUAZZANI CHAHIDI Anass - LIDOUH Mohamed
- AIT EL KOUCH Iliass - ELKHARRAZ Jamila
- ACHKOUR Mohamed - BOUJRANI Marouane
- AIT ELMKADEM Redouane

2020 /2021

ISIMA-GLIA

REMERCIEMENTS

Nous profitons, par le biais de ce rapport, d'exprimer nos remerciements les plus sincères à toute personne qui nous a aidés de près ou de loin dans l'élaboration de ce travail. Ainsi, nous adressons notre gratitude à :

Mr. Loiseau Yannick, Professeur à l'institut d'informatique d'Auvergne et notre tuteur durant ce projet pour son temps, sa guidance tout au long du travail et pour son aide continue à la réalisation de ce projet.

Mr. Guillon Bruno, professeure à l'institut d'informatique d'Auvergne du module technologie mobile à l'ISIMA, pour le temps et les efforts qu'il a consacrés pour nous enseigner et nous guider tout au long de ce projet.

Enfin, l'ensemble des acteurs de ce projet ainsi que ceux dont il dépend que nous n'avons pas rencontré mais qui auront contribué dans une certaine mesure à nous impliquer dans cette nouvelle expérience.

LISTE DES FIGURES

Figure 1	Diagramme de Gantt réel	16
Figure 2	Diagramme de Gantt réel	16
Figure 3	Création des tickets sur GitHub	17
Figure 4	Schéma de GitFlow Pipeline	18
Figure 5	Création d'un issue sur GitHub	19
Figure 6	Déclenchement du workflow pour un Pull Request	19
Figure 7	Création de release	20
Figure 8	CI en cours de traitement	20
Figure 9	CI réussi	21
Figure 10	Architecture Spring MVC	25
Figure 11	Architecture globale du projet.....	26
Figure 12	Les relations entre les entités dans la base de données	28
Figure 13	Dépendance de jjwt dans pom.xml	30
Figure 14	ContrôleurJwtAuthenticationController	31
Figure 15	Méthode de "user register"	31
Figure 16	Méthode de création du tokenjwt	31
Figure 17	méthode de construction de tokenjwt.....	31
Figure 18	exemple de création d'un compte utilisateur	32
Figure 19	méthode de "user login"	32
Figure 20	méthode de recherche d'un utilisateur	33
Figure 21	la classe JwtRequest	33
Figure 22	la classe JwtResponse	34
Figure 23	la classe JwtAuthenticationEntryPoint.....	34
Figure 24	la classe de configuration WebSecurityConfig.....	34
Figure 25	exemple d'authentification d'un utilisateur	35
Figure 26	Récupérer les lieux d'un tag.....	35
Figure 27	Récupérer les lieux qui dont id_tag est 5	36
Figure 28	Récupérer les lieux d'un utilisateur	36
Figure 29	Récupérer les lieux dont id utilisateur est 1	37
Figure 30	Récupérer toutes les étiquettes de l'utilisateur connecté	37
Figure 31	List des étiquettes pour utilisateur connecté	38
Figure 32	Ajouter des étiquettes pour un lieu	38
Figure 33	étiquette ajoutée pour le lieu dont id est 2	39
Figure 34	Mettre à jour une étiquette	39
Figure 35	Mise à jour de l'étiquette 2	40
Figure 36	Architecure de la partie frontend	40
Figure 37	Interface de login.....	41
Figure 38	interface de création de compte	42
Figure 39	Consultation de menu	43
Figure 40	interfaces de Tags.....	44
Figure 41	partage de tags	44
Figure 42	interfaces de Map	45
Figure 43	modification d'un lieu géographique	46
Figure 44	Recherche d'un lieu par titre	47
Figure 45	Recherche par étiquette	48
Figure 46	Changement du domaine de l'API	49

LISTE DES TABLEAUX

Tableau 1 cas d'utilisation "connecter"	11
Tableau 2 cas d'utilisation "recherche location"	11
Tableau 3 cas d'utilisation "ajouter location"	12
Tableau 4 cas d'utilisation "gérer location"	12
Tableau 1 Equipe du travail & division des tâches	17
Tableau 2 Matrice Des Risques	18

TABLE DES MATIÈRES

Table des matières

INTRODUCTION GÉNÉRALE.....	7
CHAPITRE 1.....	8
CONTEXTE GÉNÉRAL DU PROJET.....	8
I. CONTEXTE GÉNÉRAL DU PROJET.....	9
1. Introduction	9
2. Modélisation.....	9
2.1 CAS D'UTILISATION (USE CASE).....	9
2.2 Description des cas d'utilisation :	11
Voici une description détaillée de quelques cas d'utilisations :	11
3. Conclusion.....	13
CHAPITRE 2.....	14
PLANIFICATION & ORGANISATION DU PROJET	14
II. PLANIFICATION & ORGANISATION DU PROJET.....	15
4. Planning du déroulement du projet.....	15
4.1 Diagramme de GANTT	15
5. Méthodologie	16
6. L'équipe du travail.....	17
7. Division des tâches	17
17	
8. Collaboration Git Flow :	18
9. Intégration continue	20
10. Les risques du projet	21
11. Conclusion.....	22
CHAPITRE 3.....	23
III. Technologies et outils utilisés	24
1. Introduction	24
2. SpringBoot	24
3. Spring MVC.....	25
4. IONIC.....	26
6. CSS	27
7. Type Script.....	27
8. Base de données	28

IV. Réalisation.....	29
1. Backend :.....	29
1.1 Authentification :.....	29
1.2 Les services développés en Backend :	35
2. FrontEnd.....	40
CONCLUSION.....	49
BIBLIOGRAPHIES.....	50

INTRODUCTION GÉNÉRALE

Depuis longtemps, l'humain avait toujours besoin de se localiser. C'est-à-dire de déterminer ses coordonnées sur une surface de la terre. Soit pour avoir une idée sur sa position ou pour aller à un lieu précis. Dans le monde des applications mobiles on trouve nombreuses applications de géo localisation. Cependant, elles ne donnent pas à l'utilisateur un contrôle plus élevé sur la carte. Pour ces raisons-là, on trouve qu'une application qui permet à son utilisateur de gérer un ensemble des lieux et de les partager avec d'autres utilisateurs sera utile.

CHAPITRE 1

CONTEXTE GÉNÉRAL DU PROJET

Chapitre 1

Dans cette section :

- Introduction du sujet.
- Modélisation.

I. CONTEXTE GÉNÉRAL DU PROJET

1. Introduction

Le présent chapitre a pour objectif de situer le projet dans son environnement contextuel et organisationnel avant d'entamer les différentes phases de sa réalisation. Pour faire, nous présentons le contexte général du projet, puis la méthodologie et en fin le planning de déroulement du projet.

2. Modélisation

Cette application a pour but d'enregistrer et organiser des lieux géographiques, représentés par leurs coordonnées GPS. Ces lieux pourront aussi se voir affecter un titre, une description ainsi qu'une série d'étiquettes (tags), qui pourront être utilisées pour l'organisation et la recherche des lieux. Une image ou vignette pourra aussi être associée à un lieu.

2.1 CAS D'UTILISATION (USE CASE)

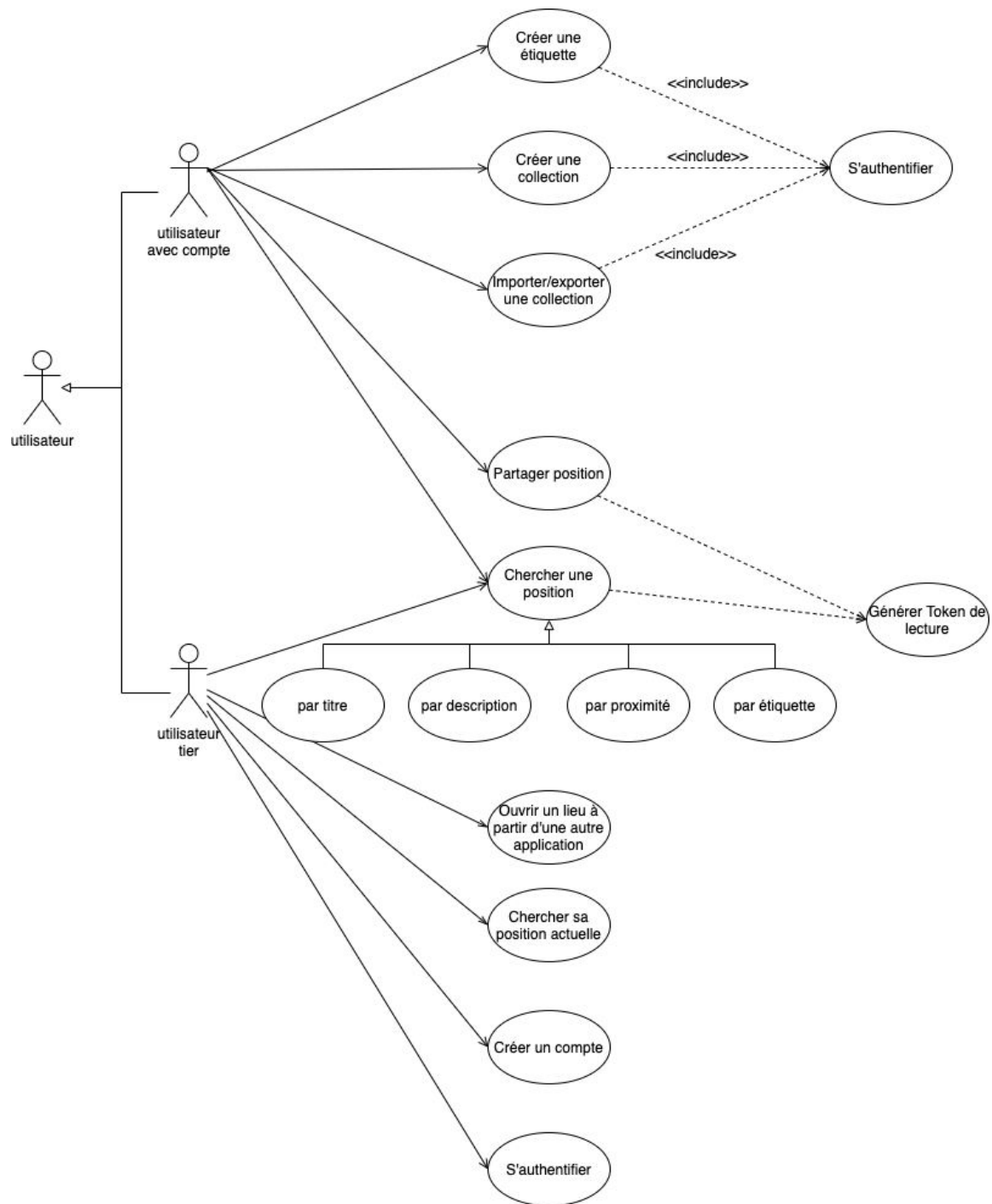
Le diagramme des cas d'utilisation permet de représenter la vision détaillée de l'application du point de vue de l'utilisateur.

Il existe un acteur qui est l'utilisateur de cette application :

- il utilisera cette application afin de consulter des lieux géographiques, ajouter, de supprimer et de modifier ces lieux, aussi les catégoriser à l'aide des tags.

L'analyse des besoins a permis de modéliser l'application MyMAP en plusieurs cas d'utilisation :

- connecter : utilisé par l'utilisateur de l'app.
- chercher sa position actuelle
- rechercher location (une position)
- gérer location
- ajouter location
- ajouter tag
- gérer tag
- partager une liste locations



2.2 Description des cas d'utilisation :

Voici une description détaillée de quelques cas d'utilisations :

Authentification :

Tableau 1 cas d'utilisation "connecter"

Libellés	Description
Nom	Connecter
Objectifs	Ce cas d'utilisation a pour objectif de permettre à des utilisateurs l'habileté d'accéder à l'application
Précondition	Pour accéder à ce cas d'utilisation, l'utilisateur doit accéder à l'application My MAP.
Scénario Nominal	L'utilisateur demande l'accès au système 2. L'application affiche la page de connexion. 3. L'utilisateur saisit son identifiant et son mot de passe 4. L'application autorise l'accès. 5. L'application affiche la page d'accueil.
Scénario Alternatif	4.1 L'application refuse l'accès. 4.2 L'application affiche un message d'erreur. Retour scénario point 2.
Résultat attendu	Accéder à l'application.

Rechercher location :

Tableau 2 cas d'utilisation "recherche location"

Libellés	Description
Nom	Rechercher location
Objectifs	L'objectif de ce cas d'utilisation est de rechercher une ou plusieurs fiches en fonction d'un ou plusieurs critères de recherche : par titre, par description ou par étiquette.
Précondition	Pour accéder à ce cas d'utilisation, l'utilisateur aura préalablement opté pour la recherche d'une location (déjà authentifié).
Scénario Nominal	L'utilisateur opte pour la recherche de location. 2. L'application affiche la page de critère de recherche. 3. L'utilisateur saisit un ou plusieurs critères de recherche. 4. L'application a trouvé là ou la location. 5. L'application affiche le résultat de recherche.
Scénario Alternatif	4.1 l'application n'a pas trouvé là ou la location. 4.2 l'application affiche un message informatif. Retour scénario point 2.
Résultat attendu	Affichage du location recherchée.

Ajouter location :*Tableau 3cas d'utilisation "ajouter location"*

Libellés	Description
Nom	Ajouter une nouvelle location
Objectifs	Ce cas d'utilisateur permettra aux utilisateurs de créer une location.
Précondition	Pour accéder à ce cas d'utilisation, l'utilisateur aura préalablement opté pour la création d'une location (déjà authentifié).
Scénario Nominal	1. l'utilisateur opte pour la création d'une location de besoin utilisateur. 3. l'utilisateur sélectionne la création de la location. 4. l'application affiche la page de création de location de besoin utilisateur. 5. l'utilisateur renseigne les données. 6. l'application contrôle les données saisies. 7. l'application enregistre les données. 8. l'application affiche la page de création d'une location de besoin utilisateur.
Scénario Alternatif	7.3 l'application n'enregistre pas les données. 7.4 l'application affiche un message d'erreur. Retour scénario nominal point 4.
Résultat attendu	Création d'une nouvelle location.

Gérer location :*Tableau 4cas d'utilisation "gérer location"*

Libellés	Description
Nom	Gérer location
Objectifs	L'objectif de ce cas d'utilisation est de gérer une location. Gérer une location consiste à consulter, à modifier ou à supprimer une location.
Précondition	Pour accéder à ce cas d'utilisation, l'utilisateur aura préalablement sélectionné une location par suite d'une recherche dans sa liste des locations.
Scénario Nominal	1. l'utilisateur sélectionne une location dans la liste des locations. 2. l'application affiche les détails de la location sélectionnée. 3. l'utilisateur consulte la location.
Scénario Alternatif 1	3.1a l'utilisateur modifie la location. 3.2a l'application contrôle les modifications. 3.3a l'application enregistre les modifications. 3.4a l'application affiche la page de la liste des locations. Ne rejoint pas un point du scénario nominal.
Scénario Alternatif 2	3.3a.1 l'application n'enregistre pas les modifications. 3.3a.2 l'application affiche un message d'erreur Retour scénario alternatif 1 point 3.1a
Scénario Alternatif 3	3.1b l'utilisateur supprime la location. 3.2b l'application demande la confirmation de la suppression. 3.3b l'utilisateur confirme la suppression.

	3.2b l'application supprime la location. 3.3b l'application rafraîchit la page de liste des locations. Ne rejoint pas un point du scénario nominal.
Scénario Alternatif 4	3.3b.1 l'utilisateur ne confirme pas la suppression. 3.3b.2 l'application ne supprime pas la location. Retour scénario nominal point 2.
Résultat attendu	Affichage des données d'une location et prise en compte des opérations effectuées par l'utilisateur sur la location.

3. Conclusion

Ce chapitre a caractérisé le point de départ pour l'élaboration de notre projet, dans la mesure où il décrit en détail son contexte général et l'aspect modélisation de tout le projet.

CHAPITRE 2

PLANIFICATION & ORGANISATION DU PROJET

Dans cette section :

- Planification et organisation du projet.
- Les méthodes de gestion du projet.
- La répartition des tâches.
- Les risques du projet.

II. PLANIFICATION & ORGANISATION DU PROJET

4. Planning du déroulement du projet

La planification du projet est une phase importante d'avant-projet. Elle consiste à prévoir le déroulement du projet tout au long des phases constituant le cycle de développement.

4.1 Diagramme de GANTT

Le diagramme GANTT est un outil de gestion de projet qui permet de visualiser dans le temps les différentes tâches du projet. Cela permet de planifier le projet et par conséquent, de le rendre plus dynamique.

La planification du projet est une étape inéluctable de celui-ci. Elle consiste à pronostiquer le déroulement du projet tout au long des phases constituant le cycle de développement. Ses principaux objectifs sont les suivants :

- Définir les tâches à réaliser.
- Fixer les objectifs.
- Coordonner les actions.
- Suivre les actions en cours.
- Rendre compte de l'état d'avancement du projet.

Nous avons élaboré un diagramme de GANTT pour ordonnancer la planification du projet et modéliser la planification des tâches nécessaires à la réalisation. Afin de constituer une approche globale du projet de type macro-tâche, nous présentons la situation générale de façon synthétique comme indiqué sur la figure ci-dessous :



Name	Begin date	End date
• Faire des recherche sur le sujet pour mieux comprendre les bes...	13/01/21	15/01/21
• Installation de l'environnement de travail	16/01/21	17/01/21
• Se mettre d'accord sur l'architecture à utiliser	18/01/21	19/01/21
• Discuter Conception du projet	20/01/21	30/01/21
• Etablir la base de données plus authentificaion	31/01/21	02/02/21
• Création des services demandés	03/02/21	17/02/21
• Rediscuter la conception	20/02/21	27/02/21
• Discuter avec la partie mobile pour se mettre daccord sur les E...	07/03/21	10/03/21
• Amélioration du code	11/03/21	13/03/21
• Lier le Back-End avec la partie mobile	14/03/21	20/03/21
• Etablir les test unitaires	18/03/21	20/03/21
• Rédaction du rapport	03/02/21	31/03/21

Figure 1 Diagramme de Gantt réel

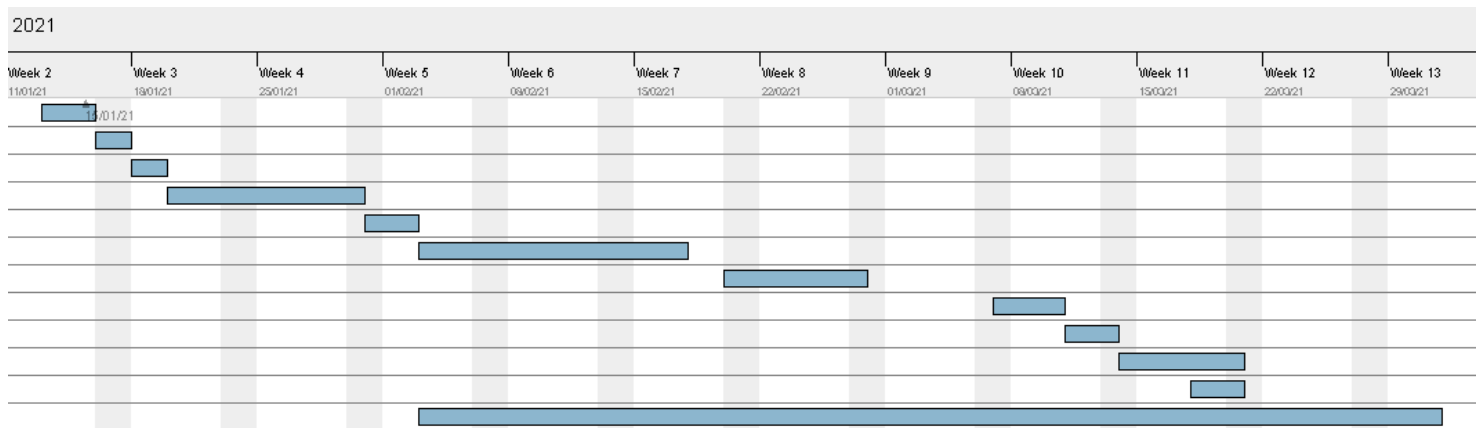


Figure 2 Diagramme de Gantt réel

5. Méthodologie

Scrum est la **méthodologie la plus utilisée parmi les méthodes Agiles existantes**. Le terme Scrum (qui signifie mêlée) apparaît pour la première fois en 1986 dans une publication de Hirotaka Takeuchi et Ikujiro Nonaka qui décrit une nouvelle approche plus rapide et flexible pour le développement de nouveaux produits. Ils comparent alors cette nouvelle méthode au rugby à XV, le principe de base étant que l'équipe avance ensemble et soit toujours prête à réorienter le projet au fur-et-à-mesure de sa progression, tel un ballon de rugby qui doit passer de main en main jusqu'à marquer un essai.

Principe

Evidemment, l'approche Scrum suit les principes de la méthodologie Agile, c'est-à-dire l'implication et la participation active du client tout au long du projet.

Considéré comme un cadre (*framework* en anglais) de gestion de projet, Scrum se compose de plusieurs éléments fondamentaux :

- Des **rôles**,
- Des **événements**,
- Des **artefacts**,
- Des **règles**.

Il s'agit d'une approche empirique (c'est-à-dire qui se base sur l'expérience), dynamique et participative de la conduite du projet. Au rugby, la mêlée est une phase indispensable car elle permet au jeu de repartir sur d'autres bases. Même chose pour Scrum : l'équipe se réunit quotidiennement lors d'une réunion de synchronisation,

appelée mêlée quotidienne, afin de suivre l'avancement du projet.

6. L'équipe du travail

La réussite d'un projet passe impérativement par une organisation rigoureuse et efficace de l'équipe du projet, c'est ainsi le cas pour notre équipe qui est responsable de la mise en place de la démarche de réalisation et la gestion du temps et des tâches allouées. Les rôles de chacun sont définis dans le tableau suivant :

Tableau 1 Equipe du travail & division des tâches

Nom	Rôle
Mohamed LIDOUH	Développeur Backend
Mohamed ACHKOUR	Développeur Backend
Marouane BOUJRANI	Développeur Backend
Ilyass AIT ELKOUCH	DevOps
Anass OUZZANI CHAHDI	Tester
Redouane AIT ELMKDEM	Développeur FrontEnd
Jamila AKHARAZ	Développeur FrontEnd

7. Division des tâches

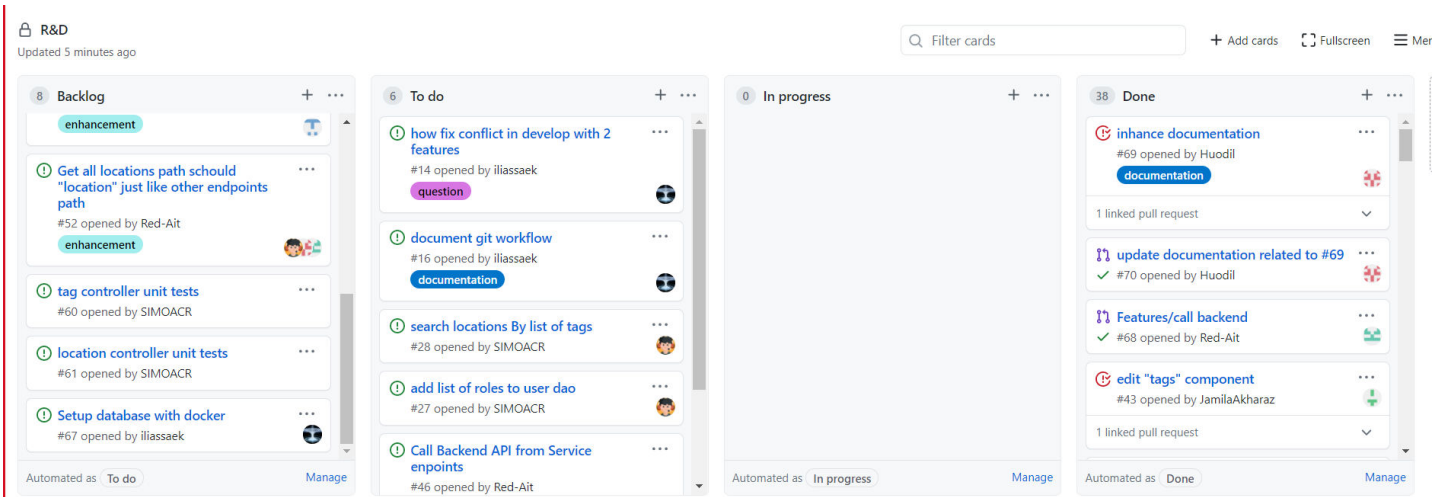


Figure 3 Création des tickets sur GitHub

8. Collaboration Git Flow :

Pour mieux gérer le versionnage du code on a adopté la méthode Gitflow comme le montre le schéma ci-dessous :



Figure 4 Schéma de GitFlow Pipeline

Le principe consiste de créer une branche défaut develop sur laquelle on ajoute les différentes fonctionnalités.

Étape 1 : créer une issue qui décrit la fonctionnalité à implémenter

Étape 2 : Établir un pull local de la branche develop et créer à partir de celle-ci une branche qui s'appelle feature/xxx

Étape3 : Commiter avec une mention des différentes issues par leurs ids (#id_issue)

Delete tag #100

[Edit](#) [New issue](#)
[Open](#) JamilaAkhara opened this issue 8 hours ago · 0 comments

Figure 5 Création d'une issue sur GitHub

Étape4 : une fois la fonctionnalité est développée, on crée un pull request pour merger la branche feature/xxx avec develop. À la fin de la journée, on organise une réunion pour faire le review.

Étape5 : Une fois le review est bon, on fait passer les tests unitaire et d'intégration et on s'assure qu'il n'y a pas de conflit de code. Cette étape est automatique avec le pipeline CI qu'on a ajouté.

Figure 6 Déclenchement du workflow pour un Pull Request

Après un bon nombre de fonctionnalités stables, on crée un release depuis develop et on le merge avec la branche master avec un tag de la version.

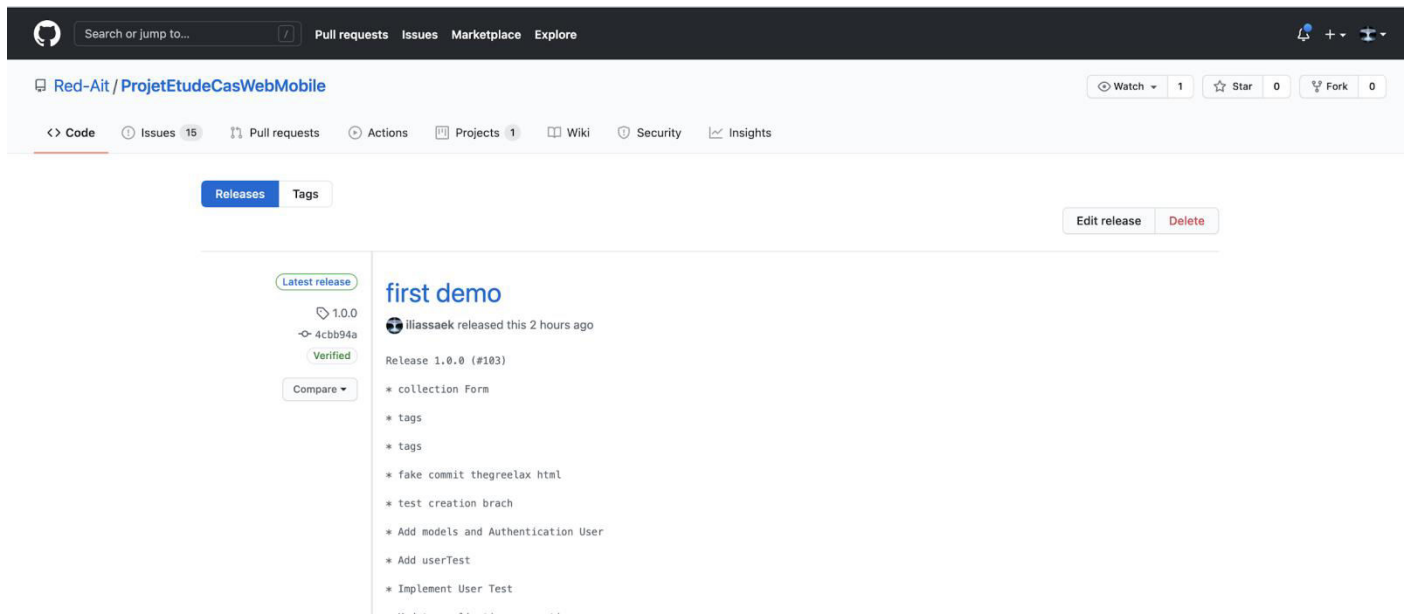


Figure 7 Création de release

9. Intégration continue

Le problème qui se posait avant était de vérifier localement si le pull request fait passer tous les tests unitaires et d'intégration dans la machine autre que celle du propriétaire du pull request. C'était une tâche pénible et répétitive du coup on a créé un pipeline Intégration Continue avec github actions où dans chaque PR les tests sont vérifiés dans les machines de github.

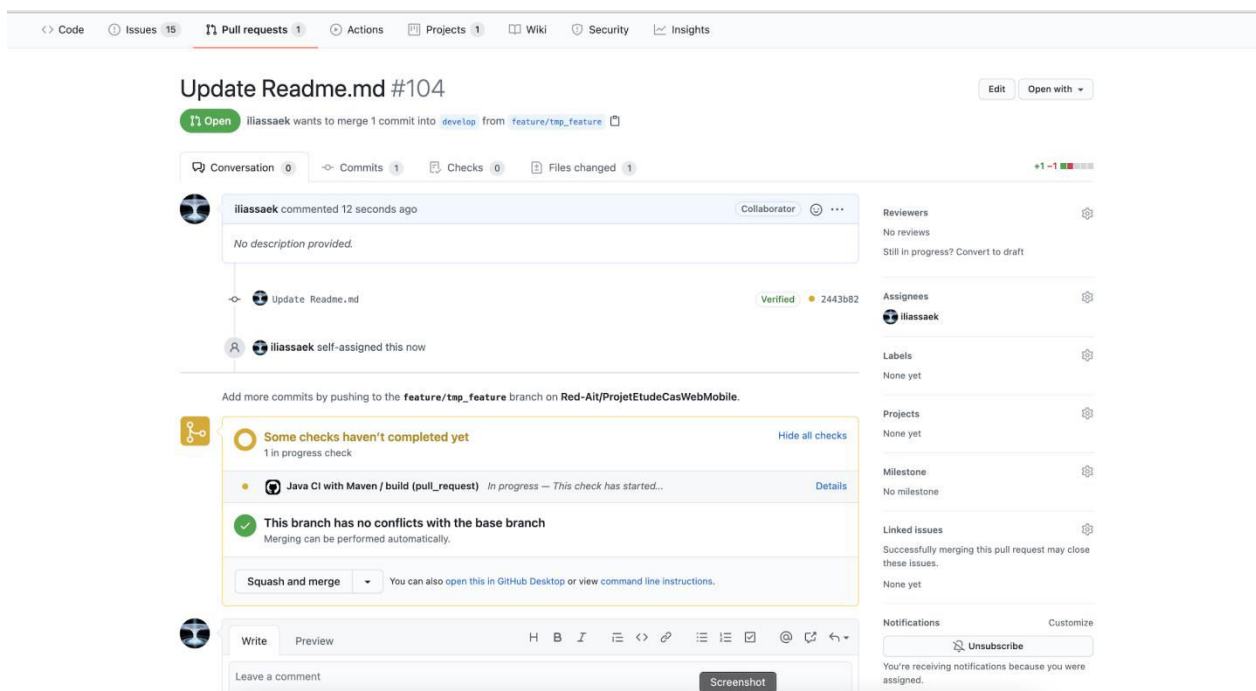
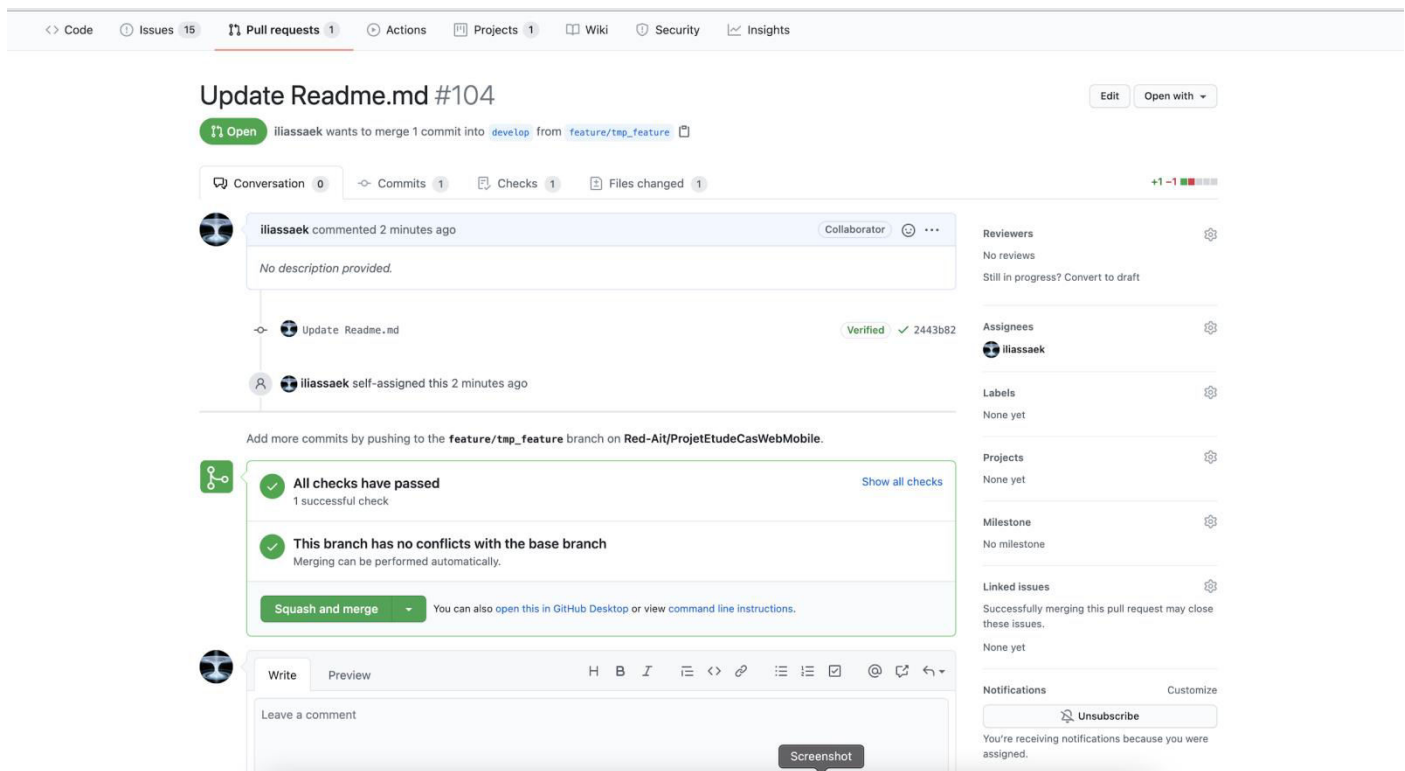


Figure 8 CI en cours de traitement

*Figure 9 CI réussi*

10. Les risques du projet

Cette partie représente la démarche relative à la gestion des risques, qui s'attache à identifier les risques, c'est à dire les pertes potentielles et quantifiables, inhérentes au projet, et associées à l'occurrence d'un événement. Cette prévention des risques tient à établir une grille des risques pouvant survenir au cours de la mise en œuvre du projet. Et pour chaque risque, nous estimons la probabilité, le niveau d'impact, le classement du risque suivi d'un ensemble d'actions préventives ainsi que celles correctives.

Tableau 2Matrice Des Risques

Risque	Type	Importance	Impact	Actions correctives
Mal compréhension du cahier de charge.	Risque bloquant	Moyenne	Création d'une ambiguïté ce qui pourra générer un retard qui influencera probablement la date de livraison	Organiser des réunions avec la responsable afin de détailler le cahier des charges.
Se familiariser avec les technologies utilisées.	Risque bloquant	Moyenne	Pour quelqu'un qui n'a jamais utilisé soit SpringBoot ou bien Ionic ça va être difficile de démarrer le projet sans avoir une connaissance de base sur ces technologies	Mieux se documenter sur ces technologies via des forums .
Installation des outils utilisés.	Risque bloquant	Elevé	Risque de retard de commencement du codage.	Attendre jusqu'à la fin de l'installation. (pas de solution)
Exécution des codes.	Risque bloquant	Elevé	L'exécution prend beaucoup de temps et beaucoup de conflits.	Vérifier les conflits et faire un merge.

11.Conclusion

Ce chapitre a présenté la méthodologie utilisée et le planning du projet à respecter au cours de la réalisation. Le chapitre suivant sera réservé à l'étude des besoins fonctionnels et organisationnels.

CHAPITRE 3

REALISATION

Dans cette section :

- Présentation des technologies & outils utilisés.
- Présentation des résultats.

III. Technologies et outils utilisés

1. Introduction

Dans le cadre de notre formation à l'ISIMA, nous sommes amenés à effectuer un projet depuis sa conception jusqu'à sa réalisation, notre groupe a choisi d'effectuer cette mission en utilisant comme technologie **SpringBoot** en Back-End et **ionic** en partie Mobile. Ce projet nous a permis de découvrir de nouvelles technologies et nous avons eu l'expérience de travail en équipe en utilisant la méthode **Agile/Scrum** pour mieux gérer l'équipe.

2. SpringBoot

Spring Boot fournit une bonne plateforme aux développeurs Java pour développer une application Spring autonome que vous pouvez simplement exécuter. Vous pouvez commencer avec des configurations minimales sans avoir besoin d'une configuration complète de Spring.

Avantages

- Spring Boot offre à ses développeurs les avantages suivants
- Facilité de compréhension et de développement des applications Spring
- Augmente la productivité
- Réduit le temps de développement

Objectifs

Spring Boot a été conçu avec les objectifs suivants

- Éviter une configuration XML complexe dans Spring
- Développer plus facilement des applications Spring prêtes pour la production.
- Réduire le temps de développement et exécuter l'application de manière indépendante
- Offrir un moyen plus facile de démarrer l'application.

3. Spring MVC

Spring MVC est un framework Java qui est utilisé pour construire des applications web. Il suit le patron de conception Modèle-Vue-Contrôleur. Il implémente toutes les fonctionnalités de base d'un framework Spring comme l'inversion de contrôle, l'injection de dépendances.

Spring MVC fournit une solution élégante pour utiliser MVC dans le cadre de Spring grâce à DispatcherServlet. Ici, DispatcherServlet est une classe qui reçoit la demande entrante et la met en correspondance avec la bonne ressource comme les contrôleurs, les modèles et les vues.

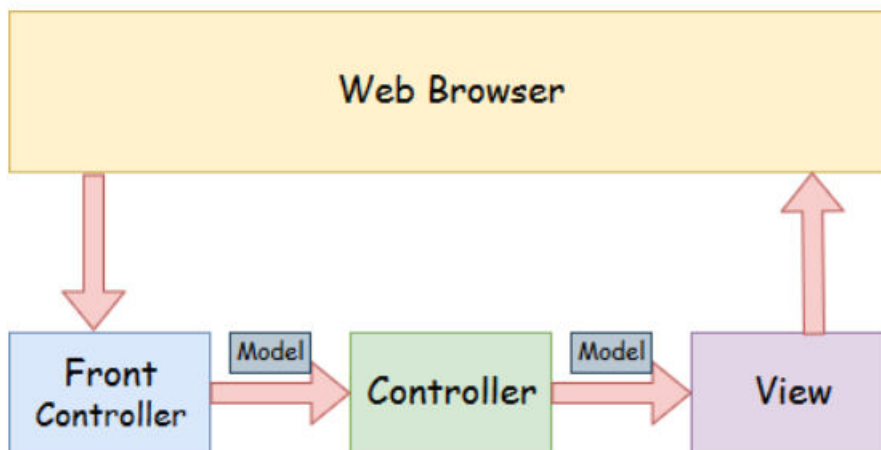


Figure 10 Architecture Spring MVC

- **Modèle** - Un modèle contient les données de l'application. Une donnée peut être un objet unique ou une collection d'objets.
- **Un contrôleur** - contient la logique métier d'une application. Ici, l'annotation `@Controller` est utilisée pour marquer la classe comme contrôleur.

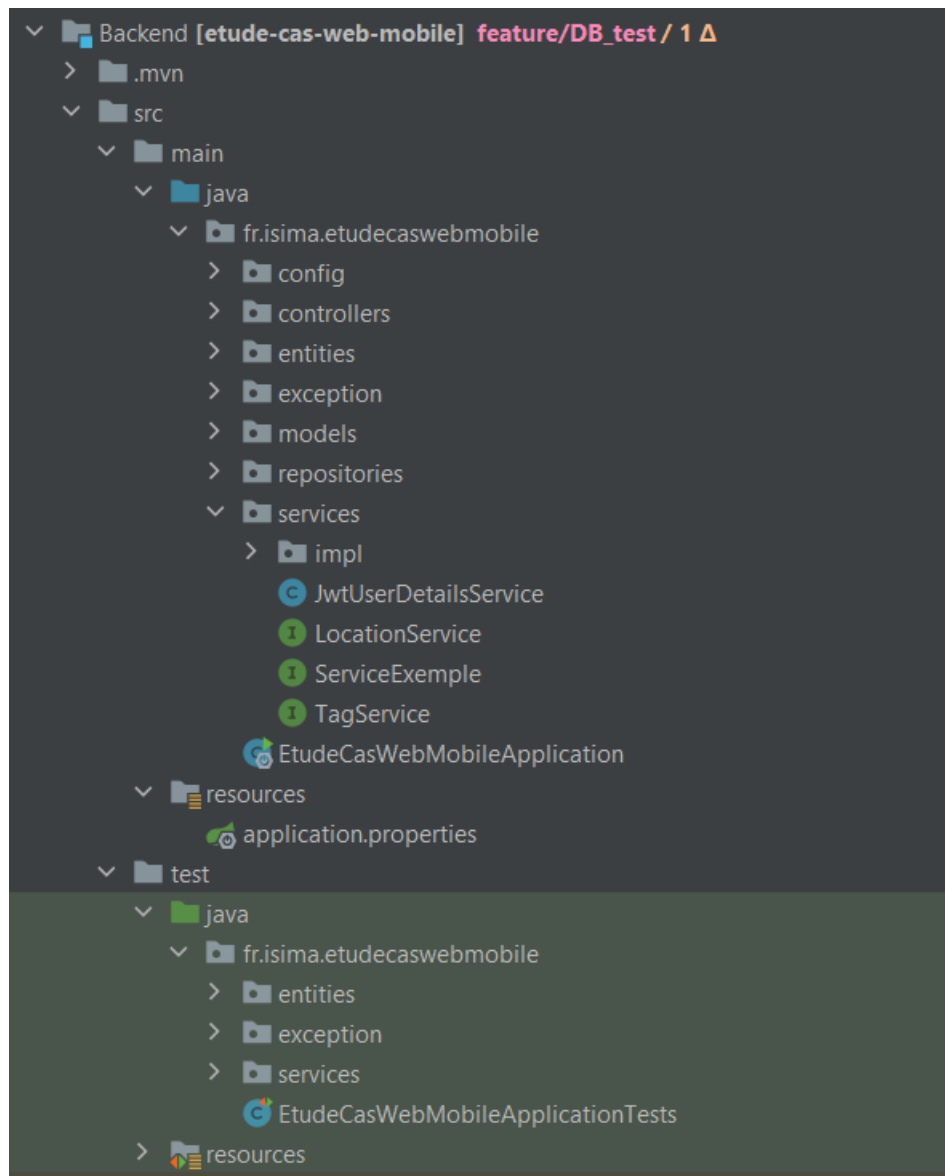


Figure 11 Architecture globale du projet

4. IONIC

Le framework Ionic est une boîte à outils UI open-source pour la création d'applications mobiles, d'applications de bureau et d'applications Web progressives performantes et de haute qualité à l'aide de technologies Web telles que HTML, CSS et JavaScript. Il permet aux développeurs de construire une fois et de l'exécuter partout. Elle a été créée par Max Lynch, Ben Sperry et Adam Bradley de Drifty Co. en 2013. La première version bêta du framework Ionic a été publiée en mars 2014.

Le framework Ionic se concentre principalement sur l'expérience utilisateur ou interaction UI qui gère toute l'apparence de votre application. Il est facile à apprendre et peut s'intégrer à d'autres bibliothèques ou frameworks tels qu'Angular, Cordova, etc.

5. HTML

Le HyperText Markup Language, généralement abrégé HTML ou dans sa dernière version HTML5, est le langage de balisage conçu pour représenter les pages web.

Ce langage permet :

- D'écrire de l'hypertexte, d'où son nom,
- De structurer sémantiquement la page,
- De mettre en forme le contenu,
- De créer des formulaires de saisie,
- d'inclure des ressources multimédias dont des images, des vidéos, et des programmes informatiques,
- De créer des documents interopérables avec des équipements très variés de manière conforme aux exigences de l'accessibilité du web.

Il est souvent utilisé conjointement avec le langage de programmation JavaScript et des feuilles de style en cascade (CSS)

6. CSS

Les feuilles de style en cascade¹, généralement appelées CSS de l'anglais Cascading Style Sheets, forment un langage informatique qui décrit la présentation des documents HTML et XML. Les standards définissant CSS sont publiés par le World Wide Web Consortium (W3C). Introduit au milieu des années 1990, CSS devient couramment utilisé dans la conception de sites web et bien pris en charge par les navigateurs web dans les années 2000.

7. Type Script

Les feuilles de style en cascade¹, généralement appelées CSS de l'anglais Cascading Style Sheets, forment un langage informatique qui décrit la présentation des documents HTML et XML. Les standards définissant CSS sont publiés par le World Wide Web Consortium (W3C). Introduit au milieu des années 1990, CSS devient couramment utilisé dans la conception de sites web et bien pris en charge par les navigateurs web dans les années 2000.

8. Base de données

MySQL est un système de gestion de bases de données relationnelles (SGBDR). Il est distribué sous une double licence GPL et propriétaire. Il fait partie des logiciels de gestion de base de données les plus utilisés au monde³, autant par le grand public (applications web principalement) que par des professionnels, en concurrence avec Oracle, PostgreSQL et Microsoft SQL Server.

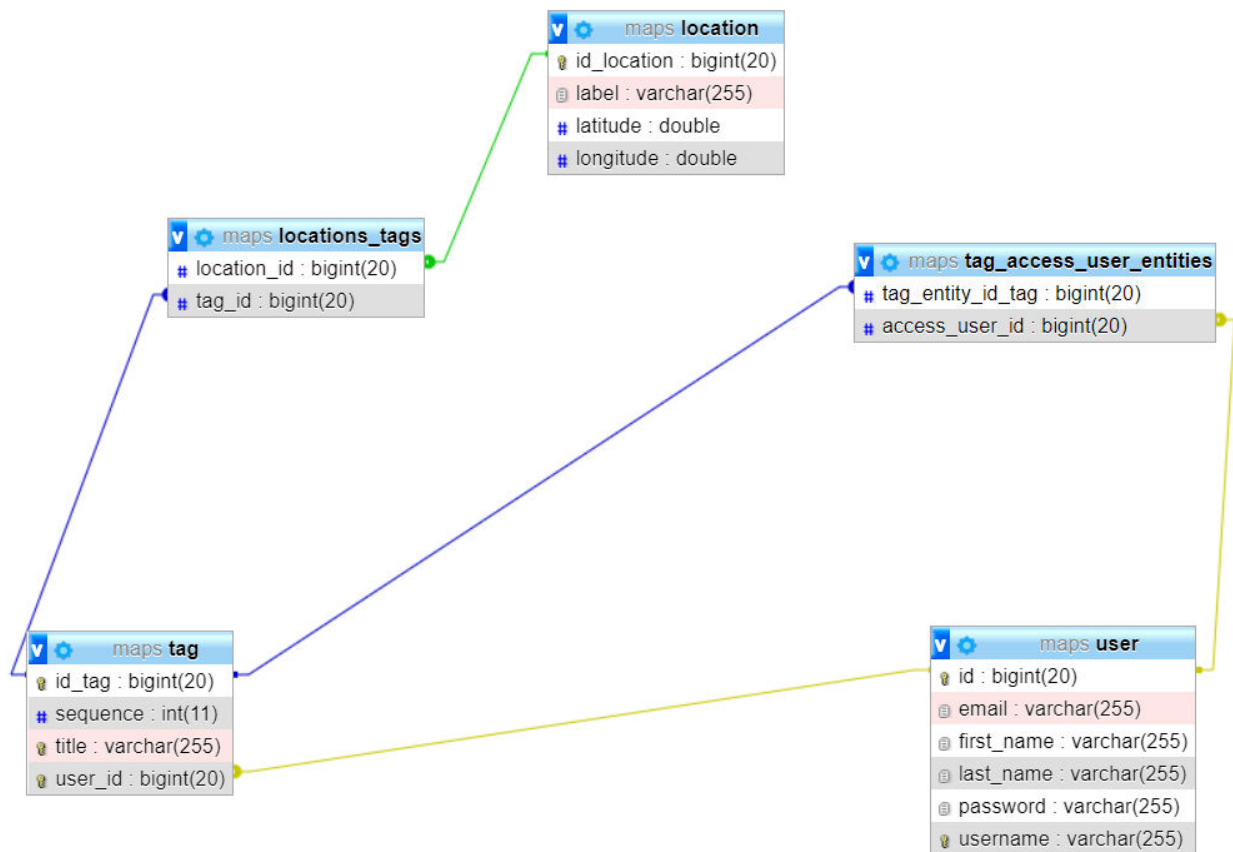


Figure 12 Les relations entre les entités dans la base de données

IV. Réalisation

1. Backend :

Dans un premier temps on a créé les entités de la base de données qui sont (user, location, tag).

L'entité **user** composée de :

- **Id user** : qui permet d'identifier l'utilisateur dans la base de données.
- **First name** : c'est le prénom de l'utilisateur.
- **Last name** : c'est le nom de l'utilisateur.
- **Email** : le mail de l'utilisateur.
- **Username** : c'est le nom d'utilisateur dans l'application.
- **Password** : c'est le mot de passe de l'utilisateur.

L'entité **location** composée de :

- **Id location** : qui permet d'identifier le lieu dans la base de données.
- **Label** : c'est en quelque sorte le nom du lieu.
- **Longitude et latitude** : coordonnées géographiques du lieu.

L'entité **tag** composée de :

- **Id tag** : qui permet d'identifier l'étiquette dans la base de données.
- **Title** : c'est le titre ou le nom que l'utilisateur peut donner à un lieu.
- **User id** : c'est le tag d'un utilisateur connecté à l'application.

Après avoir créé la base de données on a établi l'authentification dans un premier temps qui va permettre à l'utilisateur d'exploiter l'ensemble des services qui lui sont permis après authentification.

1.1 Authentification :

Pour authentifier l'utilisateur on a implémenté un système d'authentification de type BearerAuthentication utilisant le Json Web Token (JWT).

Un JSON Web Token est un accesstoken (jeton d'accès) aux normes RFC 7519 qui permet un échange sécurisé de donnée entre deux parties. Il contient toutes les informations importantes sur une entité, ce qui rend la consultation d'une base de données superflue et la session n'a pas besoin d'être stockée sur le serveur (stateless session).

Un JWT signé se compose de trois parties codées en base64 et séparées par un point :

HEADER.PAYLOAD.SIGNATURE

Il est très aisé de comprendre le fonctionnement d'un JSON Web Token en utilisant l'exemple d'une connexion utilisateur. Une clé secrète est déterminée avant l'utilisation du JWT. Dès qu'un utilisateur a entré avec succès ses données de connexion, le JWT est renvoyé avec la clé et stocké localement. Le transfert se fait par HTTPS afin de mieux protéger les données.

Lorsque l'utilisateur veut accéder à des ressources protégées comme une API ou un chemin d'accès protégé, le JWT sera envoyé par l'agent utilisateur comme paramètre (par exemple « jwt » pour les GET-Requests) ou comme en-tête d'autorisation (pour POST, PUT, OPTIONS, DELETE). L'interlocuteur peut déchiffrer le JSON Web Token et si le contrôle est réussi, exécuter la demande.

Les JSON Web Token offrent un certain nombre d'avantages comparés aux méthodes traditionnelles d'authentification et d'autorisation avec des cookies et sont pour cela utilisés dans les scénarios suivants :

Applications REST : dans les applications REST, le JWT sécurise le protocole sans état en envoyant les informations pour l'authentification directement lors de la requête.

Cross originresource sharing : le JSON Web Token envoie les informations lors du Cross Origin Resource Sharing. Cela présente un énorme avantage par rapport aux cookies, qui ne sont généralement pas envoyés dans cette procédure.

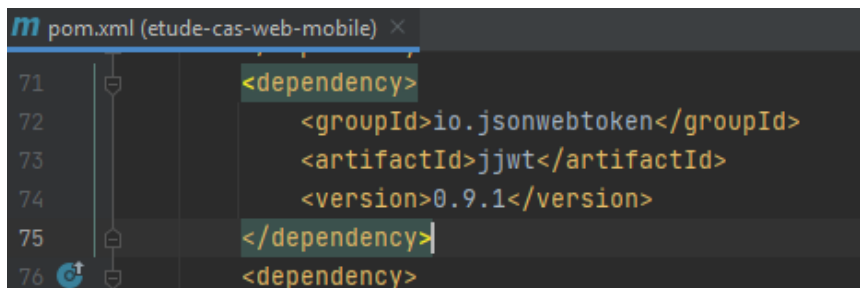
Utilisation de plusieurs frameworks : les JSON Web Token sont standardisés et donc polyvalents. Lors de l'utilisation de plusieurs Frameworks, ils permettent de partager facilement les données d'authentification.

Implémentation de JWT authentication

Pour implémenter l'authentification JWT on a utilisé la dépendance jjwt.

Pour plus d'information : <https://github.com/jwtkt/jjwt>

Dans le fichier pom.xml on ajoute la description de jjwt pour que maven puisse l'importer.



```
71 <dependency>
72 <groupId>io.jsonwebtoken</groupId>
73 <artifactId>jjwt</artifactId>
74 <version>0.9.1</version>
75 </dependency>
76 <dependency>
```

Figure 13 Dépendance de jjwt dans pom.xml

On a un contrôleur général pour l'authentification JWT :

```
@RestController
@CrossOrigin
public class JwtAuthenticationController {
|
```

Figure 14 Contrôleur JwtAuthenticationController

Créer un compte (Registre)

Il y a dans le contrôleur une méthode pour la création du compte utilisateur :

Cette méthode enregistre les informations entrées par l'utilisateur à travers la méthode http Post et les enregistre dans la base de données. Après elle appelle à la fonction generateToken de la classe jwtTokenUtil pour générer un jeton jwt à l'utilisateur.

```
@RequestMapping(value = "/register", method = RequestMethod.POST)
public ResponseEntity<?> saveUser(@RequestBody UserDto user) throws Exception {
    UserDao savedUser = userDetailsService.save(user);

    final UserDetails userDetails = userDetailsService.loadUserByUsername(savedUser.getUsername());

    final String token = jwtTokenUtil.generateToken(userDetails);

    return ResponseEntity.ok(new JwtResponse(token));
}
```

Figure 15 Méthode de "user register"

La méthode de generateToken de jwtTokenUtil retourne le résultat d'une autre méthode doGenerateToken qui se trouve au sein de la même classe

```
public String generateToken(UserDetails userDetails) {
    Map<String, Object> claims = new HashMap<>();
    return doGenerateToken(claims, userDetails.getUsername());
}
```

Figure 16 Méthode de création du tokenjwt

La méthode doGenerateToken utilise la classe Jwts pour générer le jeton. La classe Jwts se trouve dans la dépendance jjwt.

```
private String doGenerateToken(Map<String, Object> claims, String subject) {
    return Jwts.builder().setClaims(claims).setSubject(subject).setIssuedAt(new Date(System.currentTimeMillis()))
        .setExpiration(new Date(System.currentTimeMillis() + JWT_TOKEN_VALIDITY*1000)).signWith(SignatureAlgorithm.HS512, secret).compact();
}
```

Figure 17 méthode de construction de tokenjwt

Exemple :

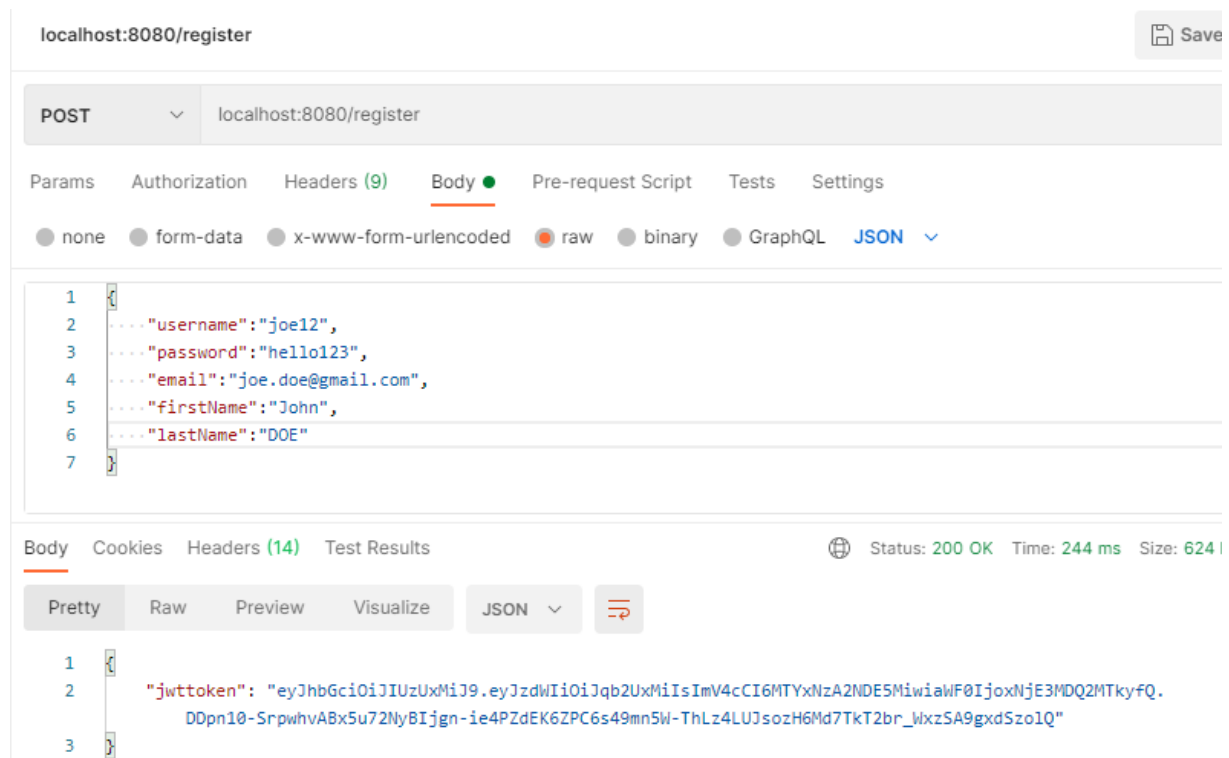


Figure 18 exemple de création d'un compte utilisateur

[Se connecter \(authenticate\)](#)

Au sein de même contrôleur d'authentification on trouve une méthode pour s'authentifier

```

@RequestMapping(value = "/authenticate", method = RequestMethod.POST)
public ResponseEntity<?> createAuthenticationToken(@RequestBody JwtRequest authenticationRequest) throws Exception {
    authenticate(authenticationRequest.getUsername(), authenticationRequest.getPassword());

    final UserDetails userDetails = userDetailsService.loadUserByUsername(authenticationRequest.getUsername());

    final String token = jwtTokenUtil.generateToken(userDetails);

    return ResponseEntity.ok(new JwtResponse(token));
}

```

Figure 19 méthode de "user login"

Cette méthode appelle la méthode loadUserByUsername de la classe JwtUserDetailsService qui cherche l'utilisateur envoyé par la méthode http Post.

```

@Override
public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
    UserDao user = userDao.findByUsername(username);
    if (user == null) {
        throw new UsernameNotFoundException("User not found with username: " + username);
    }
    return new org.springframework.security.core.userdetails.User(user.getUsername(), user.getPassword(),
        new ArrayList<>());
}

```

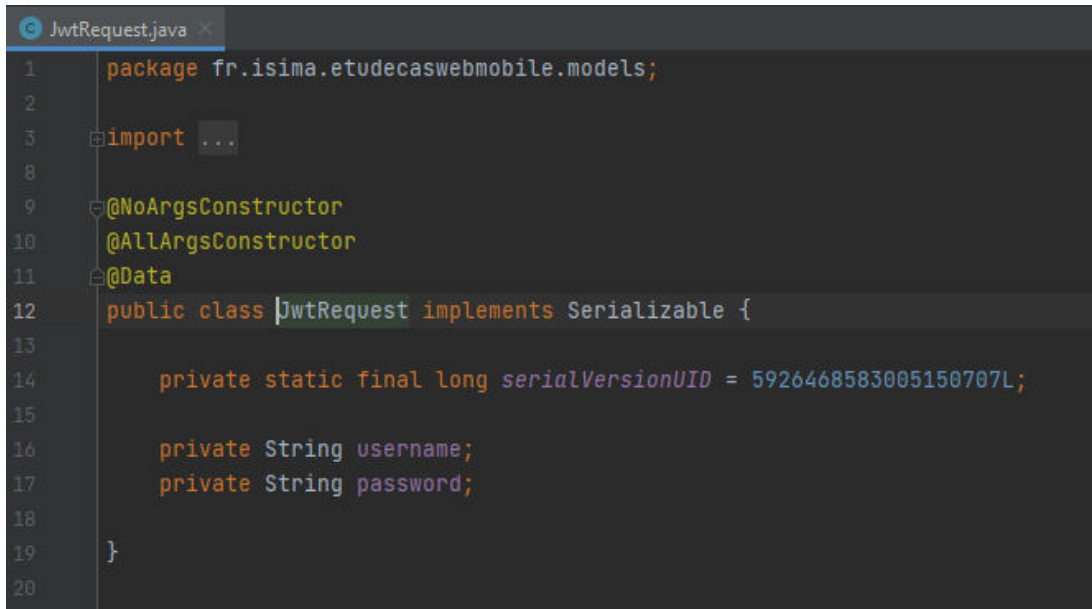
Figure 20 méthode de recherche d'un utilisateur

Si l'utilisateur est trouvé on lui génère un jeton JWT sinon on retourne un message d'erreur.

Pour interagir avec les requêtes de l'utilisateur on a deux classe JwtRequest et JwtResponse

JwtRequest

Cette classe est un modèle d'une requête d'authentification envoyée par l'utilisateur. Elle contient les deux attributs qui sont les identifiants de son login : username et password

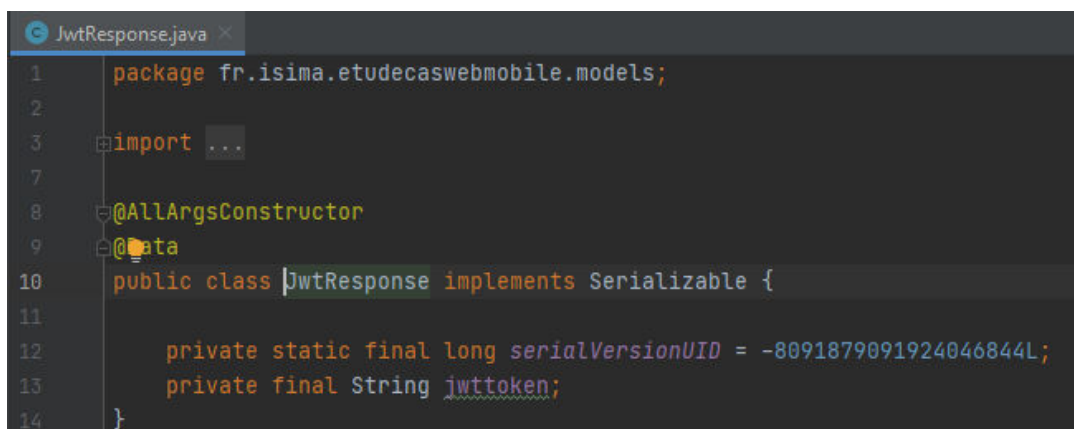


```
1 package fr.isima.etudecaswebmobile.models;
2
3 import ...
4
5
6
7
8
9 @NoArgsConstructor
10 @AllArgsConstructor
11 @Data
12 public class JwtRequest implements Serializable {
13
14     private static final long serialVersionUID = 5926468583005150707L;
15
16     private String username;
17     private String password;
18
19 }
20
```

Figure 21 la classe JwtRequest

JwtResponse

Cette classe contient l'attribut jwttoken qui prendra le token généré par le système d'authentification.



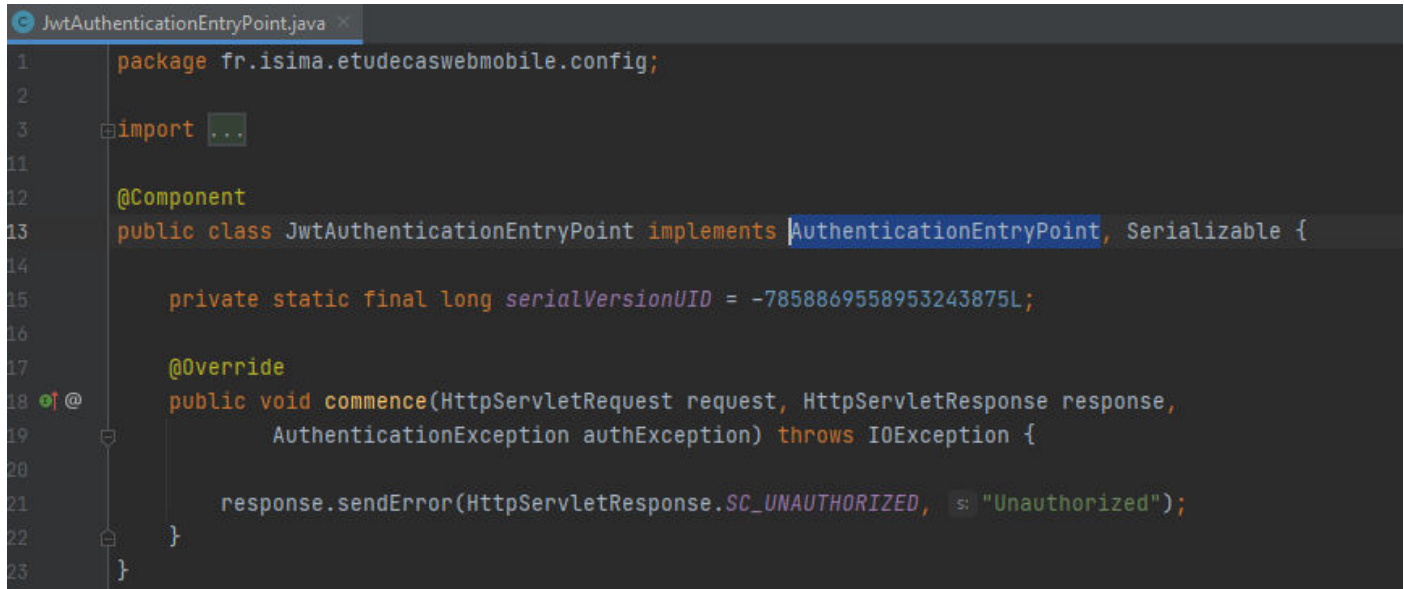
```
1 package fr.isima.etudecaswebmobile.models;
2
3 import ...
4
5
6
7
8 @AllArgsConstructor
9 @Data
10 public class JwtResponse implements Serializable {
11
12     private static final long serialVersionUID = -8091879091924046844L;
13     private final String jwttoken;
14 }

```

Figure 22 la classe JwtResponse

Remarque : l'attribut serialVersionUID est utilisé pour la serialization en java. Pour plus d'information : <https://stackoverflow.com/questions/285793/what-is-a-serialversionuid-and-why-should-i-use-it>

On trouve aussi une autre classe `JwtAuthenticationEntryPoint` qui implement l'interface `AuthenticationEntryPoint` et `Serializable`



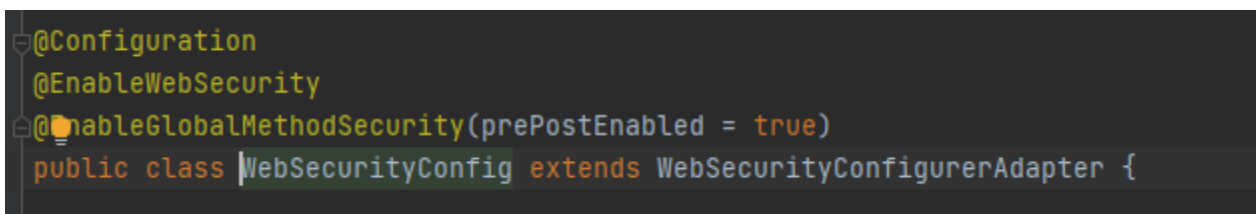
```

1 package fr.isima.etudecaswebmobile.config;
2
3 import ...
4
11
12 @Component
13 public class JwtAuthenticationEntryPoint implements AuthenticationEntryPoint, Serializable {
14
15     private static final long serialVersionUID = -7858869558953243875L;
16
17     @Override
18     public void commence(HttpServletRequest request, HttpServletResponse response,
19         AuthenticationException authException) throws IOException {
20
21         response.sendError(HttpServletResponse.SC_UNAUTHORIZED, "Unauthorized");
22     }
23 }

```

Figure 23 la classe `JwtAuthenticationEntryPoint`

Et on configure notre système de sécurité à l'aide de la classe qui hérite de la classe `WebSecurityConfigurerAdapter`



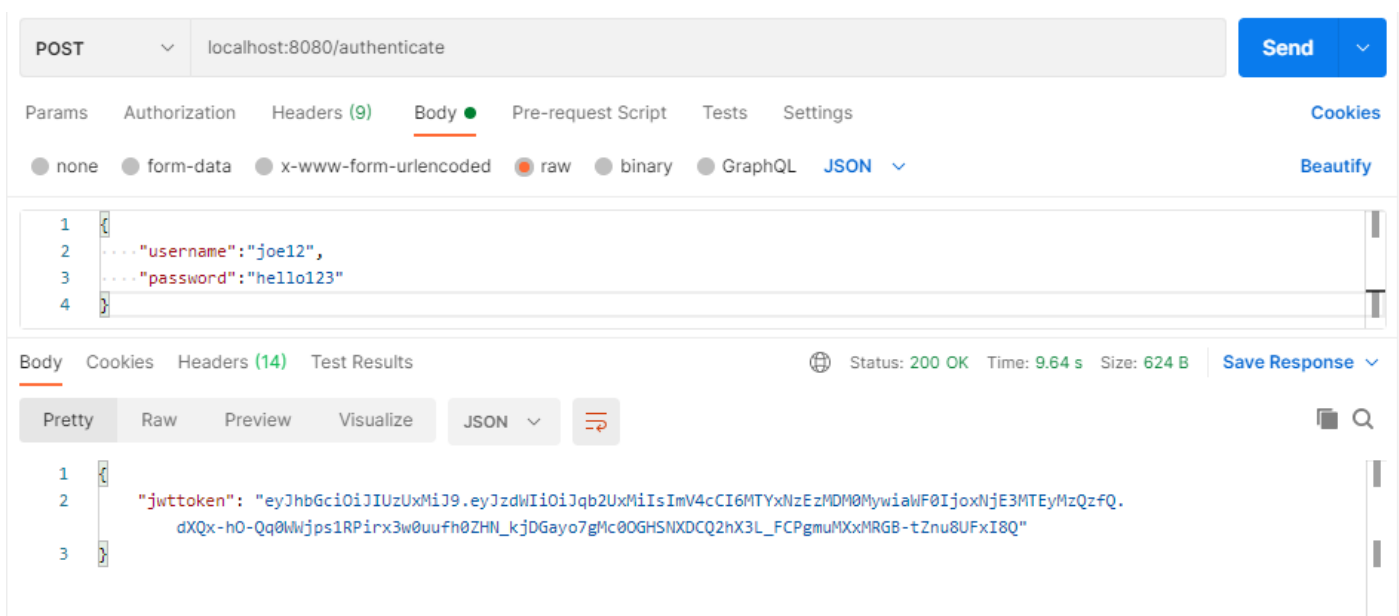
```

1 @Configuration
2 @EnableWebSecurity
3 @EnableGlobalMethodSecurity(prePostEnabled = true)
4 public class WebSecurityConfig extends WebSecurityConfigurerAdapter {
5
6 }

```

Figure 24 la classe de configuration `WebSecurityConfig`

Exemple de l'authentification :



POST localhost:8080/authenticate

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```

1 {
2   "username": "joe12",
3   "password": "hello123"
4 }

```

Body Cookies Headers (14) Test Results

Status: 200 OK Time: 9.64 s Size: 624 B Save Response

Pretty Raw Preview Visualize JSON

```

1 {
2   "jwttoken": "eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJqb2UxMiIsImV4cCI6MTYxNzEzMDM0MywiaWF0IjoxNjE3MTEyMzQzFQ.
3   dXQx-hO-Qq0Wljps1RPirx3w0uufh0ZHN_kjDGayo7gMc0OGHSNXDCQ2hX3L_FCPgmuMXxMRGB-tZnu8UFxI8Q"
4 }

```

Figure 25 exemple d'authentification d'un utilisateur

1.2 Les services développés en Backend :

Récupérer les lieux qui sont stockés dans une étiquette(tag) :

```
@Query("select l from LocationEntity l join l.tagEntities t where t.id_tag= :tag_id")
Optional<List<LocationEntity>> getLocationsByTag(@Param("tag_id") long tag_id);
```

Requête SQL qui permet de faire la jointure entre la table Location et la table Tag, et qui permet de récupérer les lieux d'un tag.

```
@Override
public List<Location> getLocationsByTag(@PathVariable long tag_id)
{
    return locationRepository.getLocationsByTag(tag_id) Optional<List<LocationEntity>>
        .orElseThrow(() -> new NoContentException("Locations Not Found")) List<LocationEntity>
        .stream() Stream<LocationEntity>
        .map(locationMapper::toModel).collect(Collectors.toList());
}
```

Figure 26 Récupérer les lieux d'un tag

C'est un service qui prend en paramètre l'id_tag et qui retourne une liste des lieux qui sont affectés à ce tag, et si le tag ne comporte aucun lieu il va générer une exception « NoContentException ».

GET http://localhost:8080/api/locations/tag/5

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Type Bearer Token Token eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJsaWRvc...

Body Cookies Headers (11) Test Results Status: 200 OK Time: 22 ms Size: 492 B Save Response

Pretty Raw Preview Visualize JSON

```
[
  {
    "id": 2,
    "label": "LOC2",
    "latitude": 55.0,
    "longitude": 55.0,
    "tags": [
      {
        "id": 1,
        "label": "default tag"
      }
    ]
  },
  {
    "id": 4,
    "label": "TAG2"
  },
  {
    "id": 5,
    "label": "TAG4"
  }
]
```

Figure 27 Récupérer les lieux qui dont id_tag est 5

Récupérer les lieux d'un utilisateur :

```
@Query(value = "SELECT l.id_location, l.label, l.longitude, l.latitude " +
    "from location l " +
    "INNER JOIN locations_tags tl on l.id_location = tl.location_id " +
    "INNER JOIN tag t on tl.tag_id = t.id_tag where t.user_id = ?1", nativeQuery = true)
Optional<List<LocationEntity>> findAllLocationsByUserId(Long id);
```

Requête SQL qui permet de faire la jointure entre la table Location et la table Tag qui est en relation avec la table User, et qui permet de récupérer les lieux avec leurs tags pour un utilisateur identifiée par son id.

```
@Override
public List<Location> findAllLocationsByUserId(Long id) {

    return locationRepository.findAllLocationsByUserId(id) Optional<List<LocationEntity>>
        .orElseThrow(() -> new NoContentException("There are no Location for this user")) List<LocationEntity>
        .stream() Stream<LocationEntity>
        .map(locationMapper::toModel) Stream<Location>
        .collect(Collectors.toList());
}
```

Figure 28 Récupérer les lieux d'un utilisateur

C'est un service qui prend en paramètre l'id de l'utilisateur et qui retourne une liste des lieux qui de l'utilisateur demandé avec leurs tag, et si l'utilisateur n'existe pas il va générer une exception « NoContentException » annonçant qu'il n'y a pas de lieu pour cet utilisateur.

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** http://localhost:8080/api/location/user/1
- Authorization:** Bearer Token (Token: eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJsaWRvc...)
- Status:** 200 OK, Time: 119 ms, Size: 1.19 KB
- Response Body (JSON):**

```
[
  {
    "id": 1,
    "label": "LOC1",
    "latitude": 45.0,
    "longitude": 45.0,
    "tags": [
      {
        "id": 1,
        "label": "default tag"
      },
      {
        "id": 2,
        "label": "TAG"
      },
      {
        "id": 3,
        "label": "TAG1"
      }
    ]
  }
]
```

Figure 29 Récupérer les lieux dont id utilisateur est 1

Récupérer les étiquettes d'un utilisateur :

```
@Query("select t from TagEntity t where t.userDao.id= :user_id")
List<TagEntity> getUserTags(@Param("user_id") long user_id);
```

Requête SQL qui permet de faire la jointure entre la table Tag et la table User, qui permet de récupérer les étiquettes pour un utilisateur identifiée par son id.

```
@Override
public List<Tag> getAllTags() {
    //List<TagEntity> tagEntities = tagRepository.findAll();
    UserDao user = userDetailsService.getCurrentUser();
    List<TagEntity> tagEntitiesUser = tagRepository.getUserTags(user.getId());

    if (!tagEntitiesUser.isEmpty())
        return tagEntitiesUser.stream()
            .map(tagMapper::toModel)
            .collect(Collectors.toList());
    else
        throw new NoContentException("Tags Not Found");
}
```

Figure 30 Récupérer toutes les étiquettes de l'utilisateur connecté

C'est un qui retourne une liste des étiquettes d'un utilisateur, et si les étiquettes n'existent pas il va générer une exception « NoContentException » annonçant qu'il n'y a pas trouvé d'étiquettes.

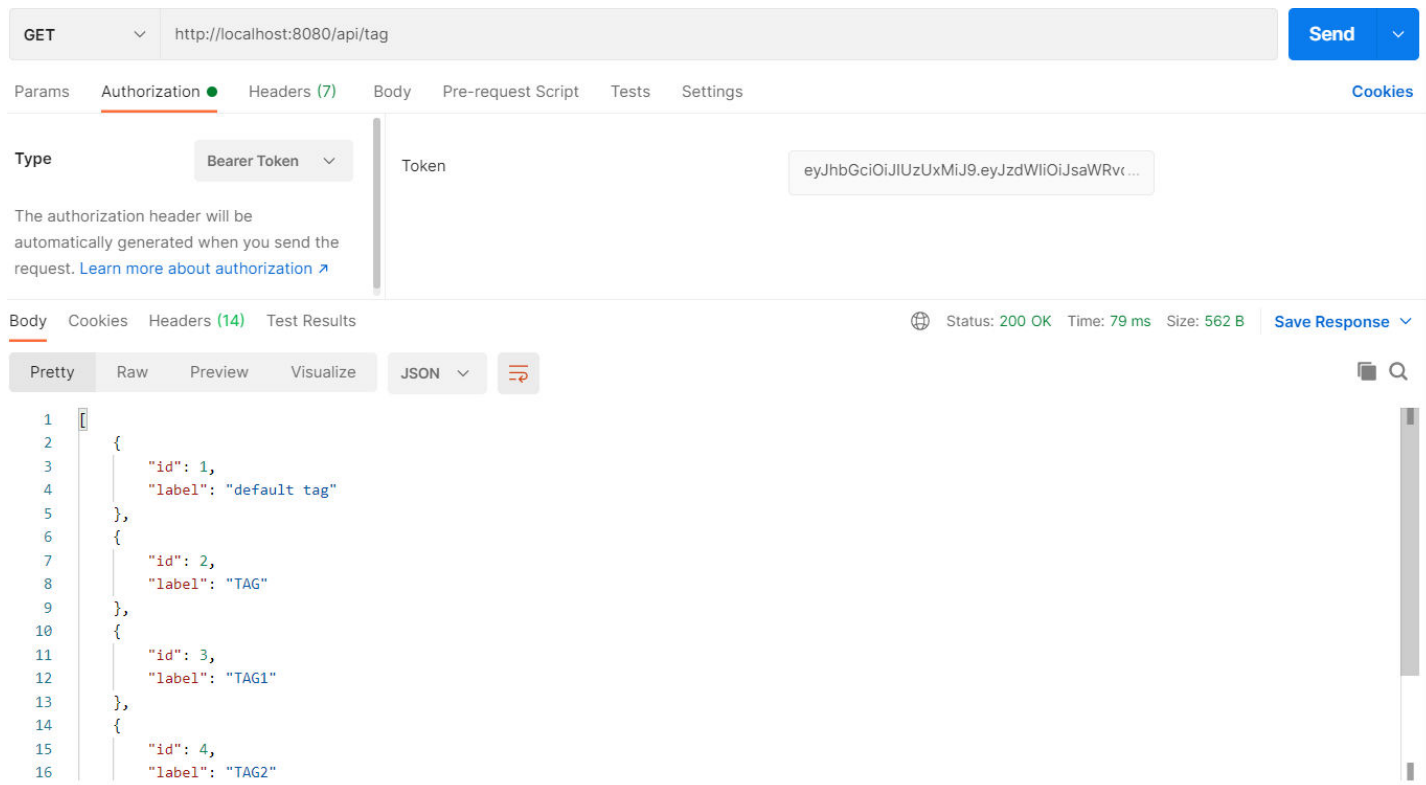


Figure 31 List des étiquettes pour utilisateur connecté

Ajouter une étiquette a un lieu :

```

@Override
public Tag addTagToLocation(Long location_id, Tag tag)
{
    TagEntity newTag = tagMapper.fromModel(tag);

    Optional<LocationEntity> locationOptional = locationRepository.findById(location_id);
    if (locationOptional.isPresent()) {
        LocationEntity location = locationOptional.get();

        List<LocationEntity> locationEntities = newTag.getLocationEntities();
        locationEntities.add(location);
        newTag.setLocationEntities(locationEntities);

        List<TagEntity> tagEntities = location.getTagEntities();
        tagEntities.add(newTag);
        location.setTagEntities(tagEntities);

        locationRepository.save(location);

        return tagMapper.toModel(tagRepository.save(newTag));
    } else {
        throw new NotFoundException("Location Not Found");
    }
}

```

Figure 32 Ajouter des étiquettes pour un lieu

Ce service permet d'ajouter une étiquette a un lieu et qui prend en paramètre location_id et tag et qui retourne le tag ajouté au lieu.

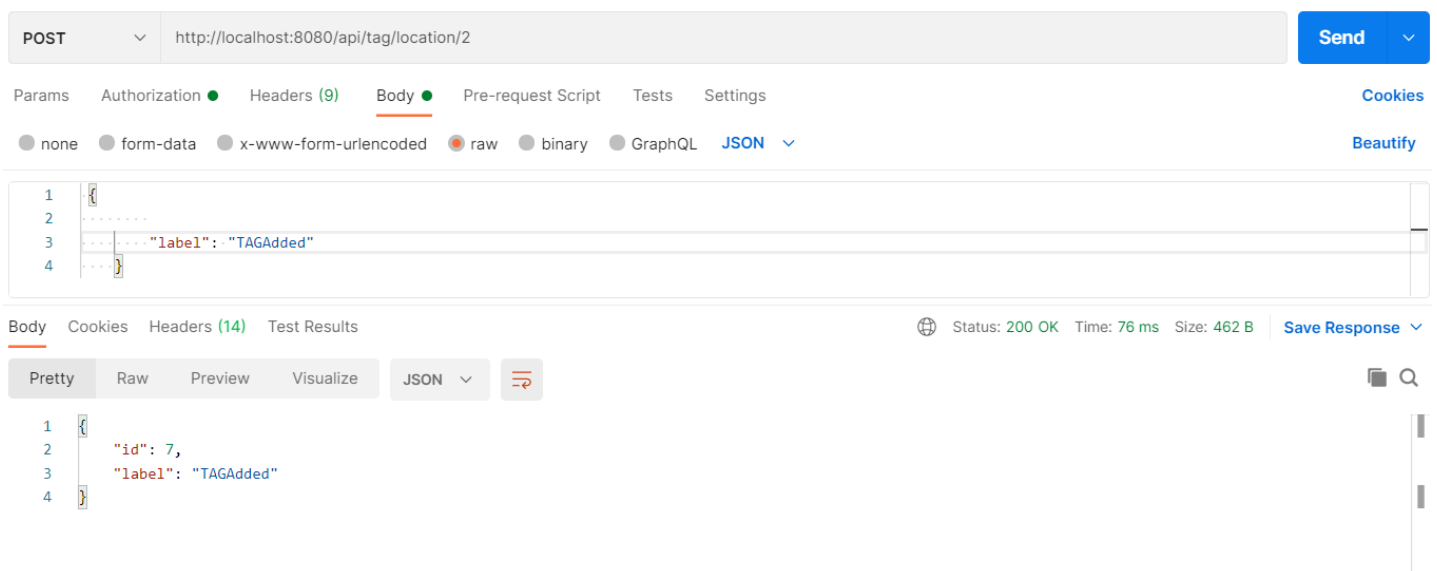


Figure 33 étiquette ajoutée pour le lieu dont id est 2

Mettre à jour une étiquette :

```
@Override
public Tag updateTagById(Tag newTag, Long id) {
    Optional<TagEntity> optionalTagEntity = tagRepository.findById(id);
    if (optionalTagEntity.isPresent()) {
        TagEntity oldTagEntity = optionalTagEntity.get();
        oldTagEntity.setTitle(newTag.getLabel());
        return tagMapper.toModel(tagRepository.save(oldTagEntity));
    } else {
        throw new NotFoundException("Tag not found");
    }
}
```

Figure 34 Mettre à jour une étiquette

C'est le service qui permet de mettre à jour une étiquette déjà existante en utilisant son id, qui prend en paramètre la nouvelle étiquette et l'id puis retourne l'étiquette mise à jour.

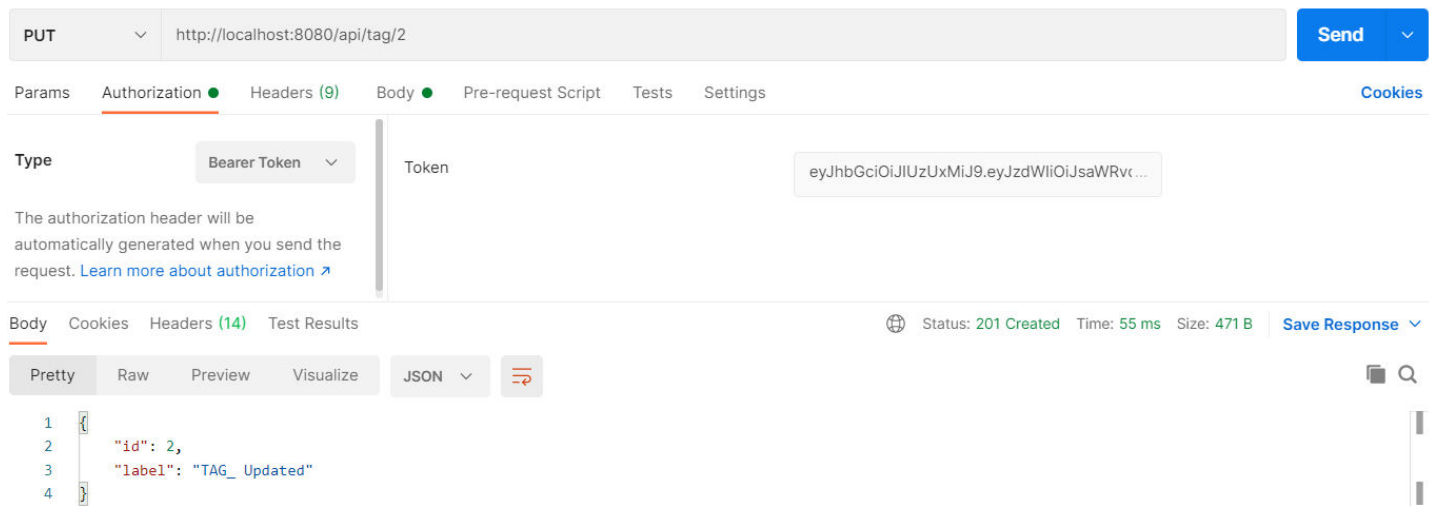


Figure 35 Mise à jour de l'étiquette 2

2. FrontEnd

La partie Front-end est développée par le framework ionic v6 basé sur Angular v11, elle se compose de deux modules principaux, Authentication et Location, et en se basant sur la bibliothèque NGXS pour gérer son état. L'architecture suivante montre les différentes interactions entre les composants de l'application frontend :

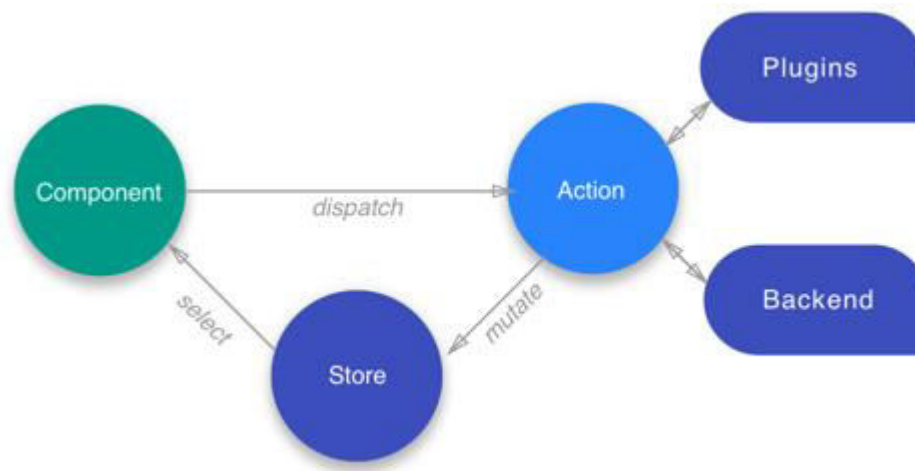


Figure 36 Architecture de la partie frontend

- Les composants ne se connectent pas directement aux services, mais ils se contentent seulement de lancer des actions et lire les données de store.
- **Store** : Stockage et gestion des données.
- **Action** : Définit un comportement particulier.
- **Plugins** : Différents plugins ajoutés (Exemple : Leaflet.js pour l'affichage de la carte)
- **Backend** : API de l'application.

Se connecter (Login)

Commençons tout d'abord par l'interface de l'authentification.

La figure ci-dessous représente l'interface principale de l'authentification qui demande un identifiant et un mot de passe afin d'accéder à l'application.

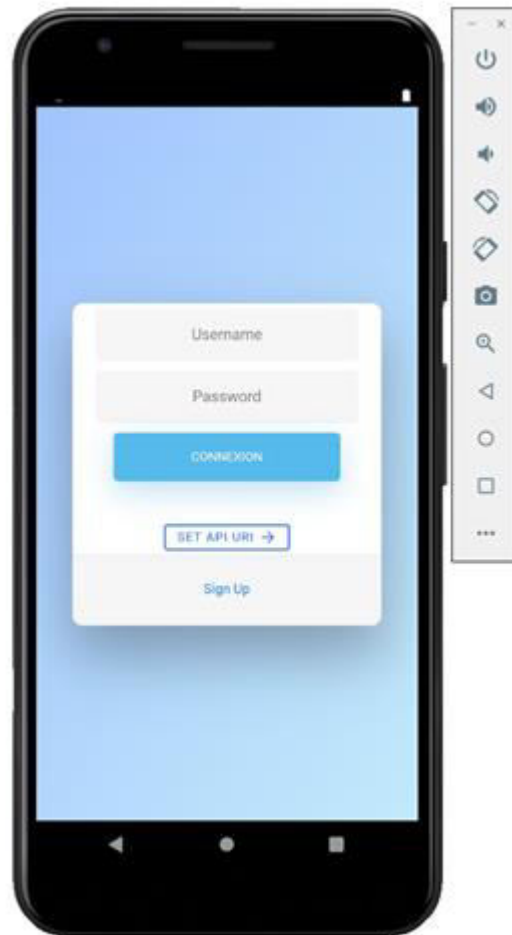


Figure 37 Interface de login

Création de compte

Pour un nouvel utilisateur, il aura la possibilité de créer son propre compte ainsi accéder à l'application. La figure 2 suivante montre l'interface de création de comptes.

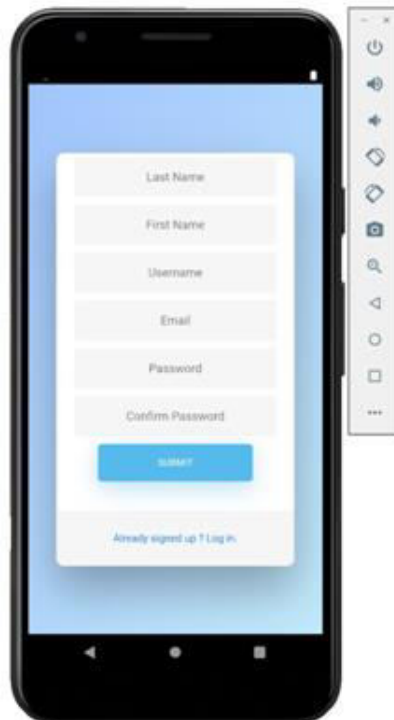


Figure 38 interface de création de compte

Consultation de menu

On donne à l'utilisateur la possibilité de consulter le menu de l'application et une simple navigation entre les différentes interfaces, comme le montre la figure suivante.

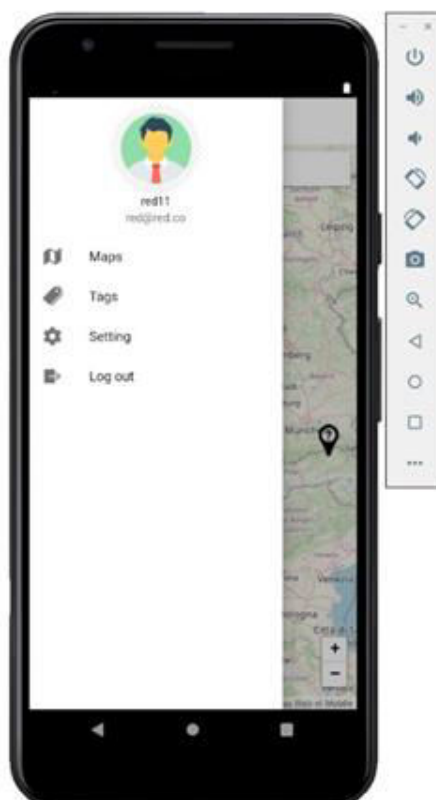


Figure 39 Consultation de menu

Interfaces des étiquettes

L'utilisateur peut accéder à la liste des tags (voir figure 4) et gérer ses propres tags, ainsi que la possibilité de partager le tag avec un autre utilisateur de l'application

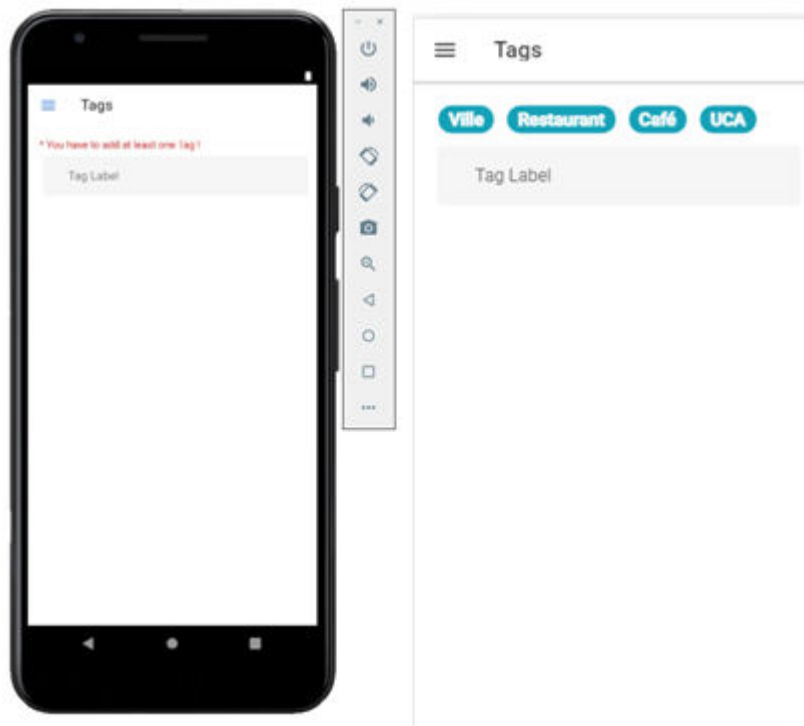


Figure 40 interfaces de Tags



Figure 41 partage de tags

Interface des localisations

Notre application permet à l'utilisateur authentifié d'accéder à l'interface Map, ainsi on lui donne la possibilité d'enregistrer un lieu géographique en lui affectant une liste d'étiquettes (étiquettes déjà créées ou même l'ajout de nouvelles étiquettes lors de l'enregistrement de lieu).

Les interfaces suivantes montrent ce cas.

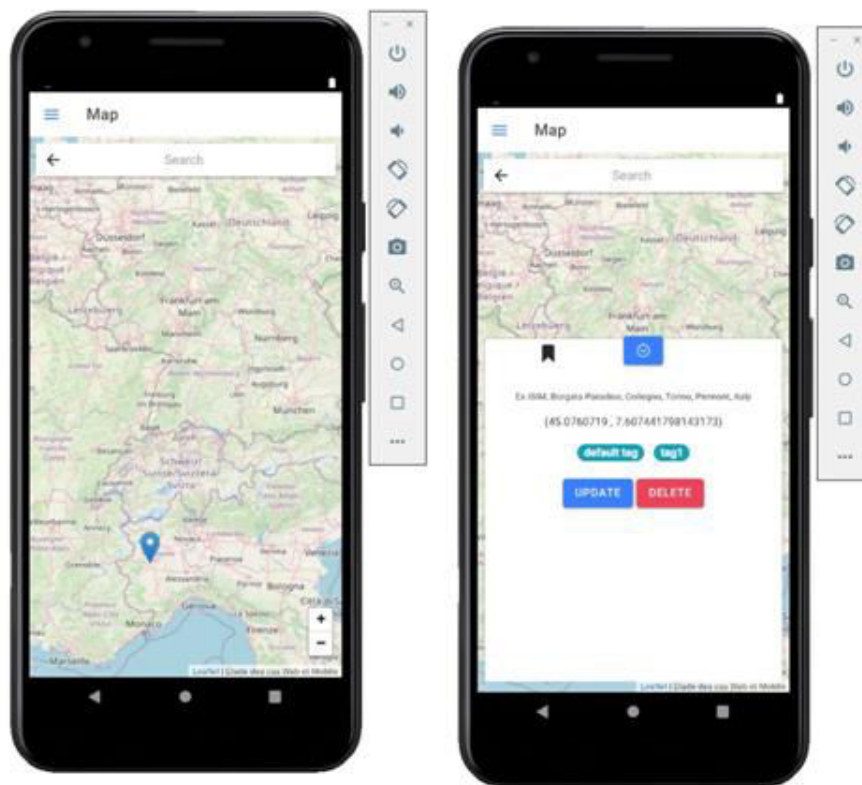


Figure 42 interfaces de Map

Modification d'un lieu géographique

L'interface suivante montre le cas de modification de lieu enregistré, cette modification concerne l'ajout ou la suppression des tags affectés au lieu, comme indiqué sur la figure suivante.

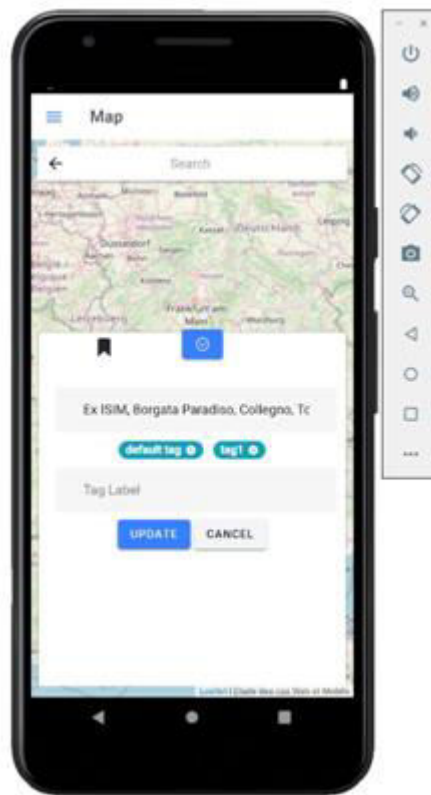


Figure 43 modification d'un lieu géographique

Recherche d'un lieu précis par mot-clé (titre)

L'utilisateur peut effectuer une recherche d'un certain lieu par son titre ou bien par étiquette.

Pour cette raison deux modes de recherche sont disponibles sur notre application afin de satisfaire le besoin de l'utilisateur.

Mode de recherche par titre :

Les interfaces suivantes montrent la recherche par titre : par exemple la recherche de l'emplacement exacte de « Paris ».

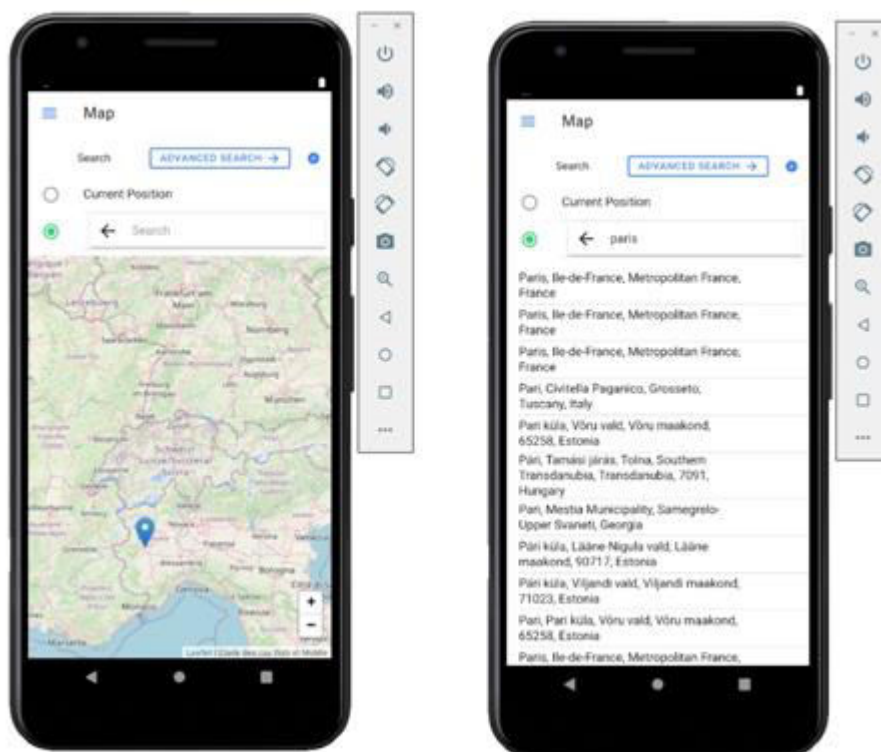


Figure 44 Recherche d'un lieu par titre

Mode de recherche par étiquette

Les interfaces suivantes montrent la recherche par étiquette avec deux possibilités « Join mode OR » ou « Join mode AND » : la recherche de lieux avec l'ensemble des tags sélectionnés, ou seulement avec une ou plusieurs tags.

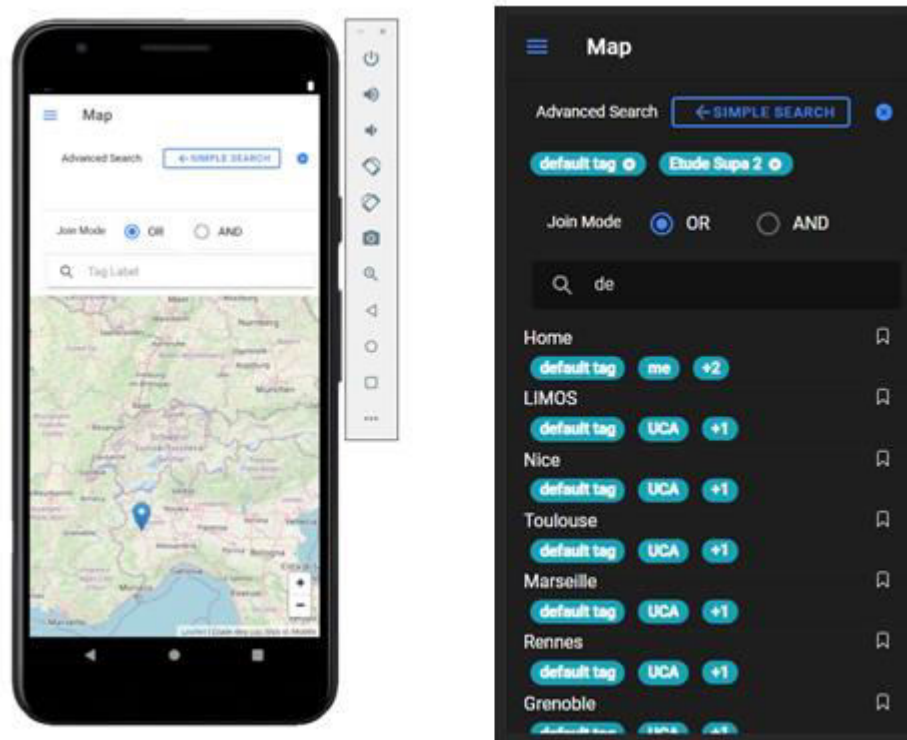


Figure 45 Recherche par étiquette

Interface de paramètres

Cette interface permet de changer le domaine de l'API comme la montre les figures suivantes.

Ainsi permettre à l'utilisateur de se connecter sur différents domaines.

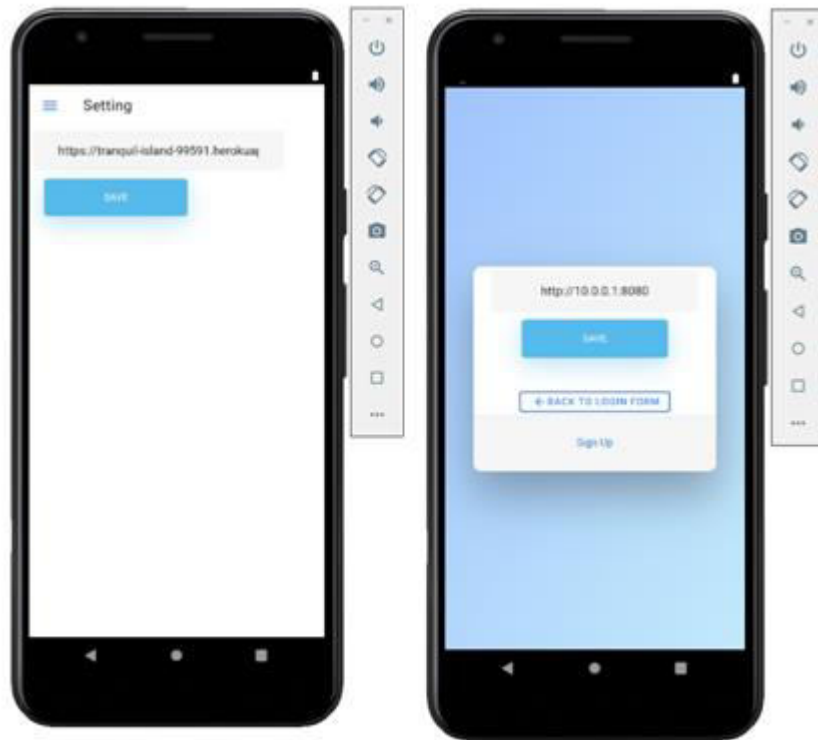


Figure 46 Changement du domaine de l'API

CONCLUSION

Dans le cadre de notre projet, nous devions réaliser un projet d'innovation et de conception qui a pour vocation de mettre en œuvre une application de gestion des lieux favoris, cette expérience a été très enrichissante pour l'ensemble des membres de l'équipe, d'une part, sur le plan personnel, elle nous a appris que le respect des idées et le choix des autres membres de l'équipe est primordial dans le travail en équipe.

L'objectif principal de ce projet est la mise en œuvre de la solution technique qui a permis d'automatiser et de simplifier le processus de l'application « MyMAP ».

Ce projet nous a permis de progresser sur différents points et améliorer la qualité de conception et de développement d'application Web.

L'expérience acquise lors de la gestion de ce projet sera appliquée sur la gestion de nos futurs projets afin de toujours m'améliorer aussi bien au niveau humain qu'en termes de méthodologie et de conception.

Dans le cadre de la réalisation de ce, nous étions totalement autonomes et nous réalisons l'intégralité des tâches du processus du développement, c'est-à-dire la réalisation des spécifications fonctionnels et techniques, le développement et l'intégration de l'app.