



# Bivouac N' Go

PRESENTATION

**Professeur**  
NAUROY Christophe

# Notre Application !

Trouver l'endroit idéal pour camper peut parfois s'avérer compliqué... C'est pourquoi nous avons créé **Bivouac N' Go**, l'Airbnb des campements !

Le concept est simple : les propriétaires de terrains peuvent proposer leur espace en location en tant qu'hôtes, et les campeurs peuvent facilement les réserver pour vivre une expérience unique en pleine nature. Que vous cherchiez un coin isolé ou un emplacement proche de sentiers, Bivouac N' Go vous offre une solution pratique et conviviale.

# Fonctionnalités Implémentées

Les utilisateurs de l'application sont à la fois **hôte** et **voyageur**, ils peuvent :

## Fonctionnalités profil/authentication :

- Créer un compte.
- Modifier les informations de son compte. (nom, téléphone, email, etc.)
- Faire une demande de suppression de son compte et toutes ses données afin de respecter le RGPD.
- Se connecter à l'application avec ce compte.

# Fonctionnalités Implémentées pt.2

## Fonctionnalités Hôte :

- Mettre leur espace (cabane, terrain, etc.) en location pour que les autres utilisateurs puissent effectuer des réservations.
- Préciser les équipements et services disponibles lors de la mise en location de leur espace.
- Gérer des espaces qu'il a mis en location (supprimer etc;).
- Visualiser tous les biens déposés et toutes les demandes de réservation en cours/passées.
- Accepter ou refuser une demande de réservation pour leur propre bien.
- Attribuer une note et rédiger un avis sur le voyageur une fois le séjour terminé.

# Fonctionnalités Implémentées pt.3

## Fonctionnalités Voyageur :

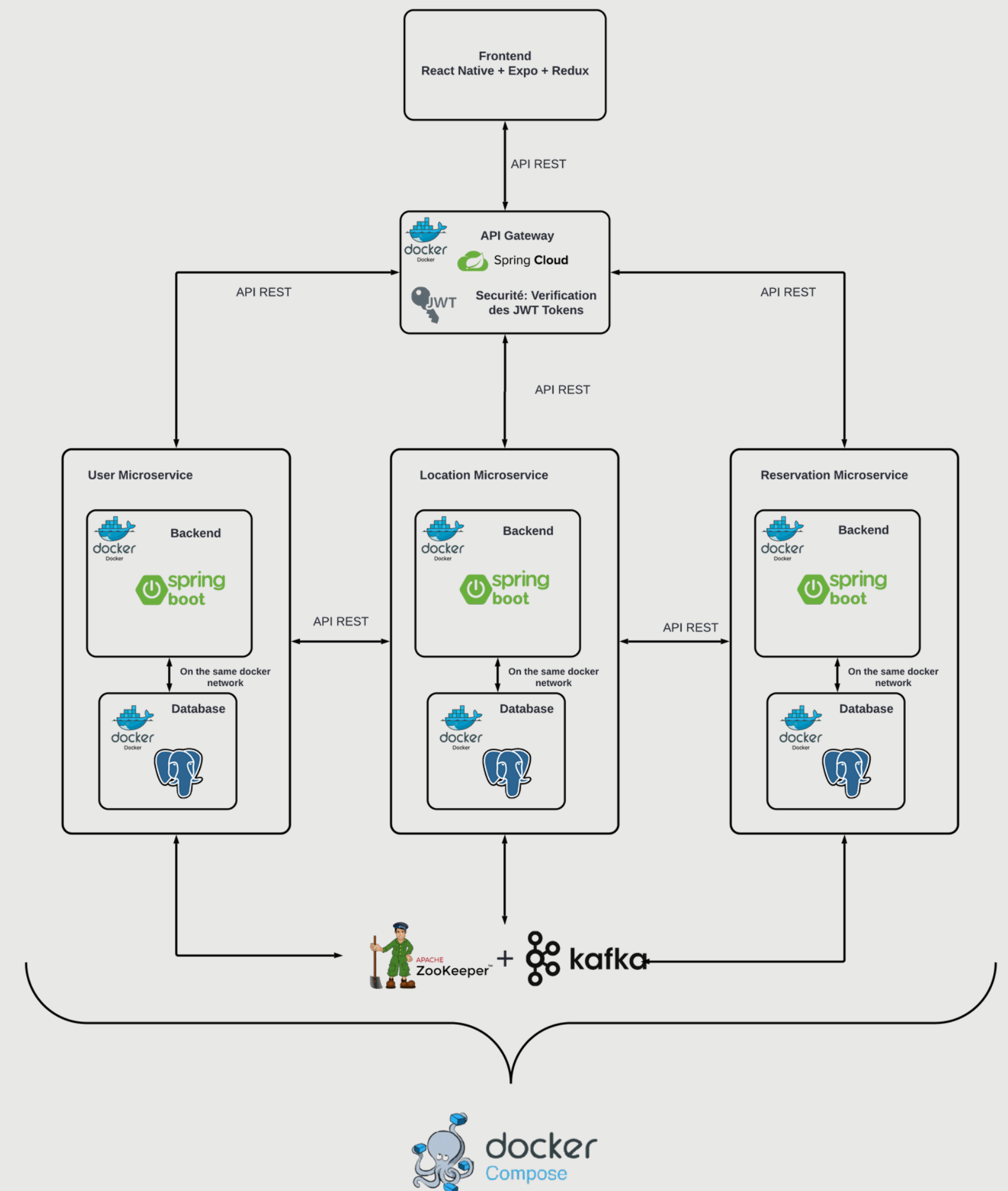
- Faire des demandes de réservation pour les espaces d'autres utilisateurs, en choisissant les dates, le nombre de personnes. Il peut aussi laisser un message de réservation à l'hôte.
- Rechercher un emplacement selon certains critères (Dates, nombre de personnes, emplacement... ).
- Visualiser l'emplacement d'un espace sur une carte.
- Attribuer une note et rédiger un avis sur la réservation une fois le séjour terminé.
- Consulter les avis que l'hôte lui a attribué sur ses réservations.
- Visualiser toutes les réservations et leurs états i.e. en cours/refusées/passées
- Consulter l'historique des ses réservations.

**L'administrateur**, qui dispose de toutes les fonctionnalités précédentes, peut également :

- Valider la suppression d'un compte.
- Définir les services et équipements que chaque hôte peut mentionner lors de la mise en location.

# Architecture

- Une architecture en microservice pour une meilleure séparation des responsabilités et une organisation claire des fonctionnalités.
- Chaque microservice possède une base de données indépendante PostgreSQL.
- Deux images Docker par microservice, une pour l'application Spring Boot et une pour la base de données
- API Gateway et vérification des JWT Token pour centraliser et sécuriser les requêtes venant du frontend
- Kafka et requêtes API REST pour la communication inter-microservice



# Organisation des Repos

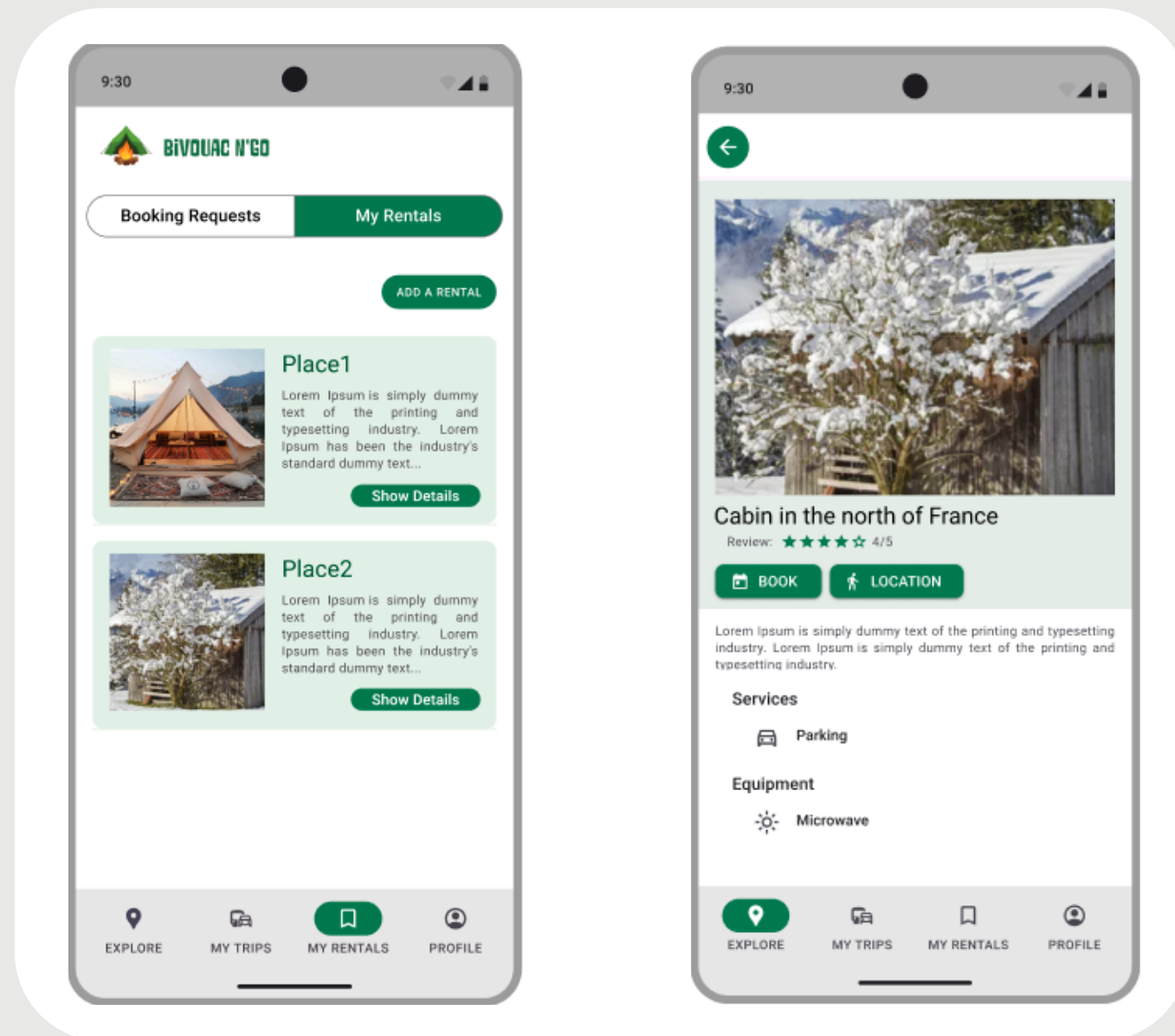
- Le frontend est organisé en plusieurs dossiers (views, components, store)
- Le backend est organisé en projet multi-module Java avec 4 modules:
  - les 3 microservices
    - user\_microservice
    - location\_microservice
    - reservation\_microservice
  - l'API Gateway



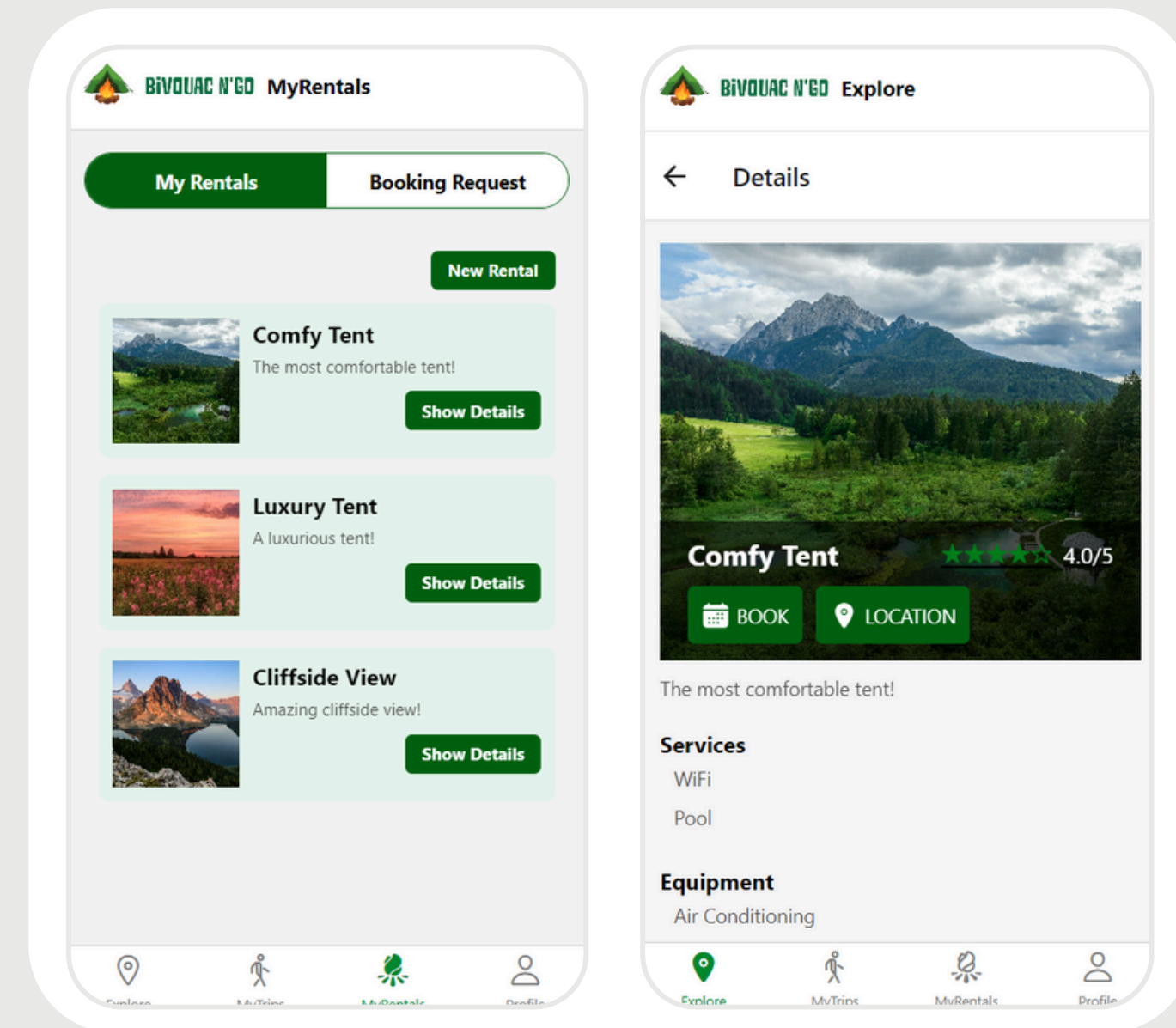


# UI/UX

## Maquettes



## Résultats



Nous avons veillé tout au long du projet à respecter scrupuleusement les maquettes définies avant la phase de développement



# Redux Toolkit

- Nous avons utilisé Redux Toolkit dans le frontend pour gérer l'authentification, notamment le stockage et la récupération sécurisée des tokens via AsyncStorage, l'envoi des requêtes de connexion et d'inscription, et la gestion des erreurs associées.
- Cela nous a permis de centraliser l'état utilisateur et d'assurer une synchronisation efficace avec l'interface.



# API Gateway : Spring Cloud

- Centralisation des communications entre nos 3 microservices.
- Point d'entrée unique pour toutes les requêtes externes.
- Simplification de la gestion des accès et des politiques de sécurité.
- Optimisation du routage des requêtes vers les microservices appropriés.
- Centralisation des fonctionnalités transverses : authentification, gestion des erreurs, etc.
- Facilite l'évolution et le déploiement indépendant des microservices.

# Sécurité: Gestion JWT Tokens



- Mise en place d'une authentification basée sur des tokens JWT pour sécuriser l'accès aux microservices.
- Chaque requête (excepté celles liées à la connexion) vers l'API Gateway inclut un JWT dans l'en-tête.
- Le JWT est décodé pour récupérer l'identifiant de l'utilisateur et ses autorisations.
- Protection des routes de l'API par des contrôles d'accès basés sur les rôles.
- Les ressources sensibles (ex : routes admin) sont accessibles uniquement par les utilisateurs autorisés.
- Les tokens ont une durée de vie limitée (24h) et sont régénérés en cas de modification des rôles ou des autorisations.

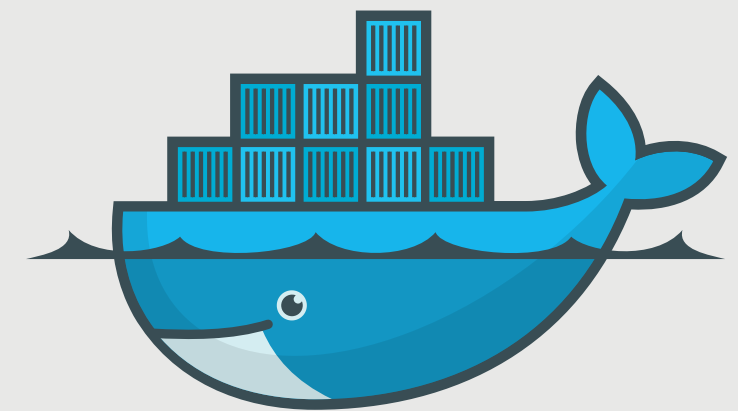
# Apache Kafka et Zookeeper

- Mise en place de Kafka pour la communication inter-microservices.
- Les microservices échangent via des topics Kafka pour une communication asynchrone et scalable.
- **Exemple:** Lorsqu'un utilisateur est supprimé, un message est envoyé sur des topics pour que les autres microservices soient notifiés et puissent, à leur tour, supprimer les informations nécessaires.
- Zookeeper assure la gestion et la coordination des brokers Kafka.

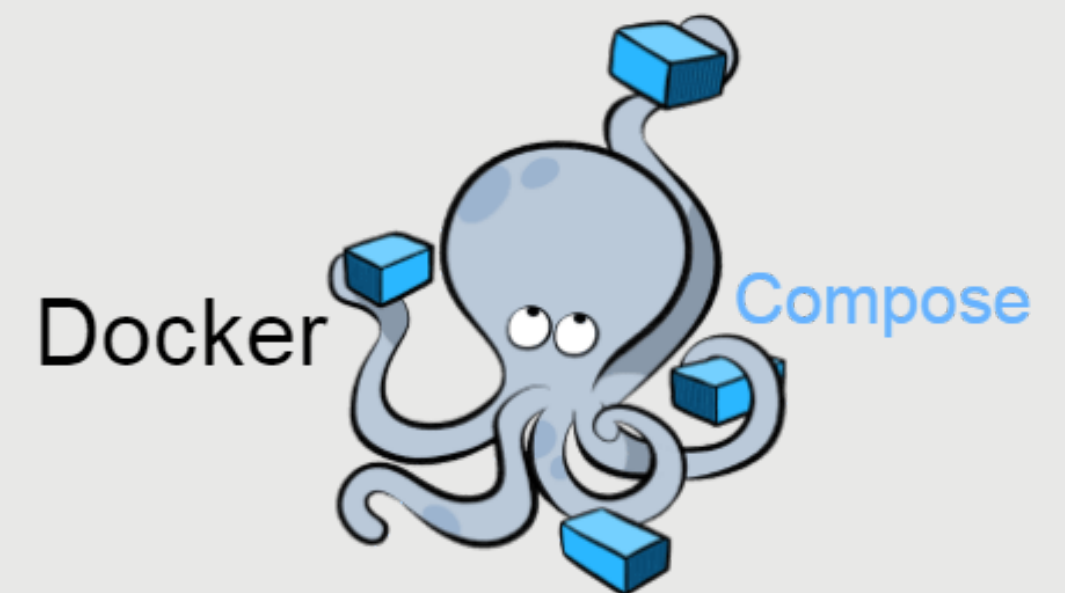


# Docker et Docker Compose

- Chaque microservice, ainsi que l'API Gateway, est encapsulé dans un conteneur Docker afin de garantir leur isolation et leur portabilité.
- Les bases de données, Kafka, et Zookeeper sont également dans des conteneurs Docker distincts.
- L'ensemble des conteneurs Docker est orchestré à l'aide de Docker Compose, ce qui facilite grandement leur démarrage, leur arrêt, ainsi que le déploiement en production de l'application.

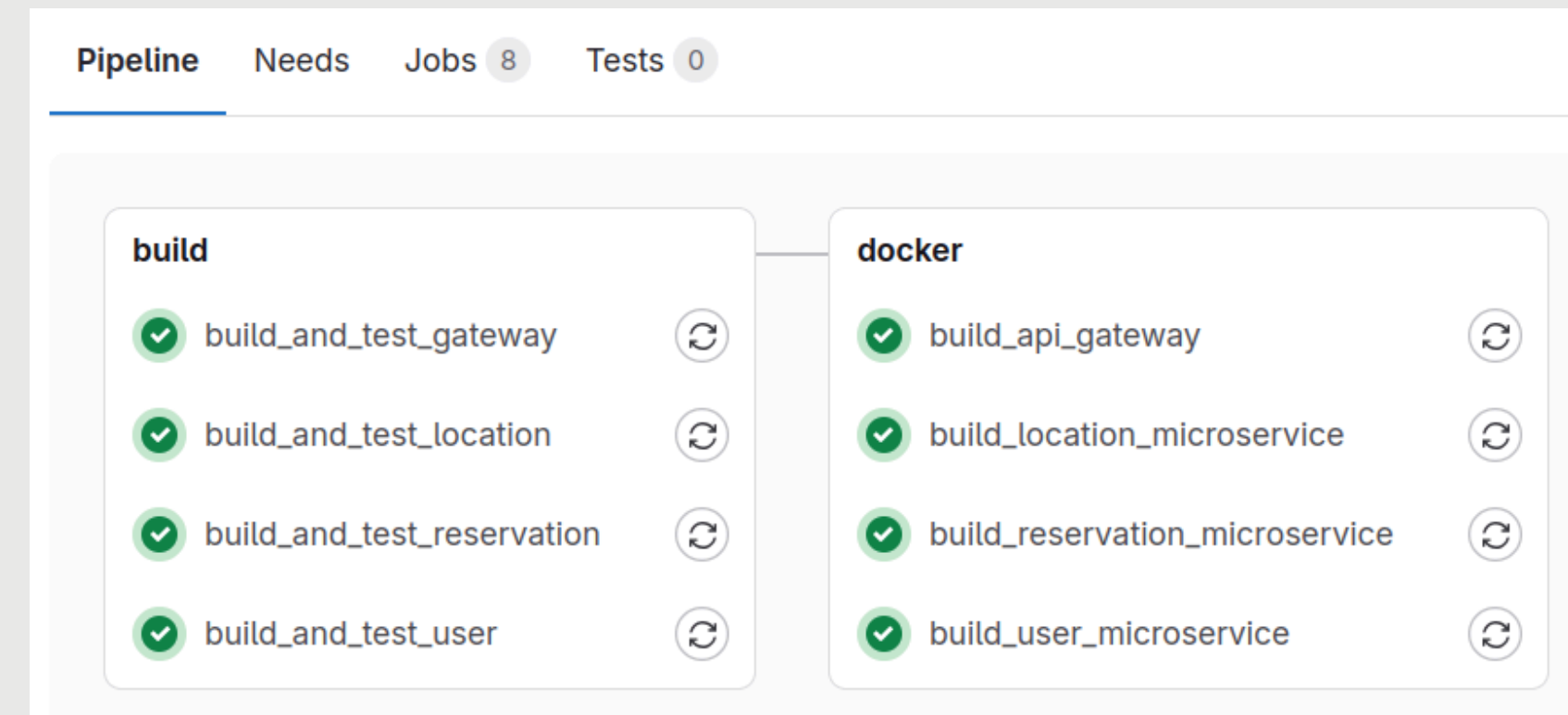
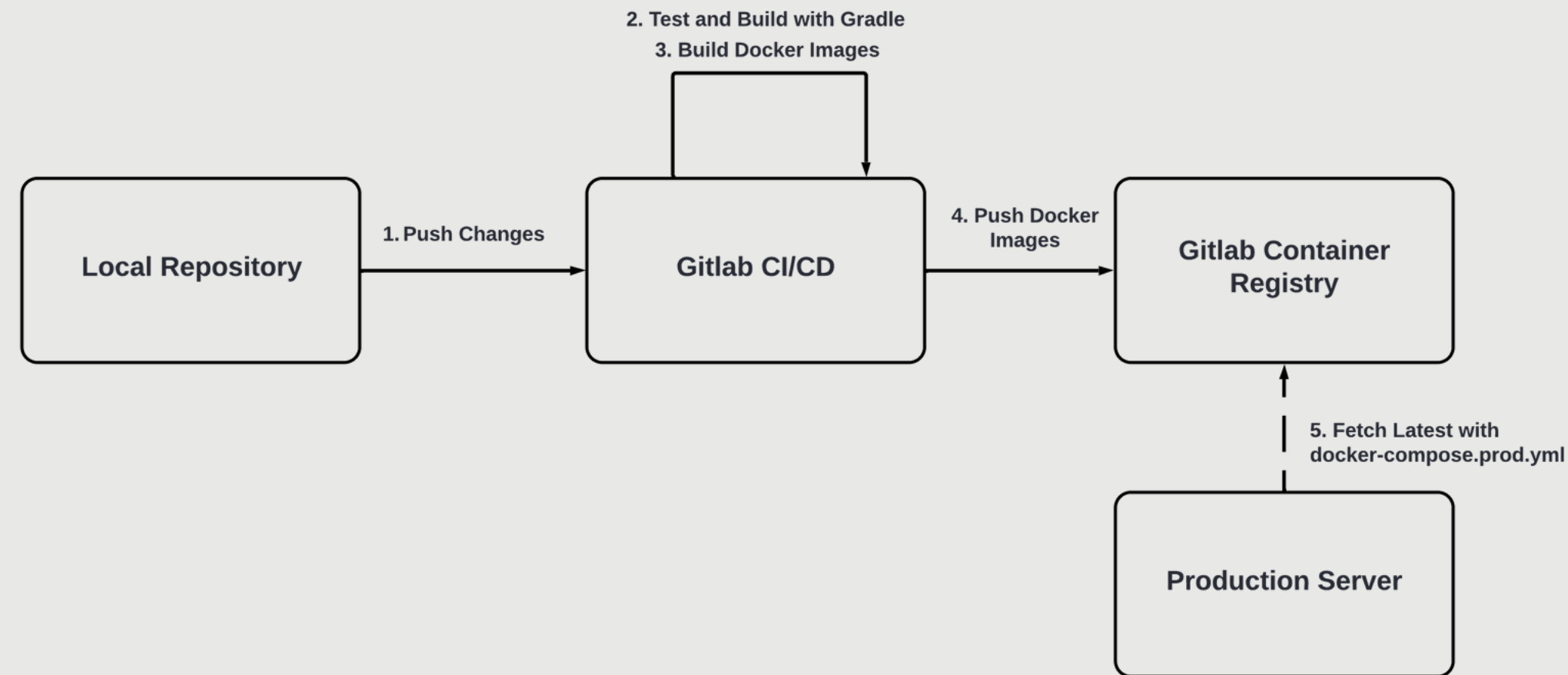


+



# CI/CD

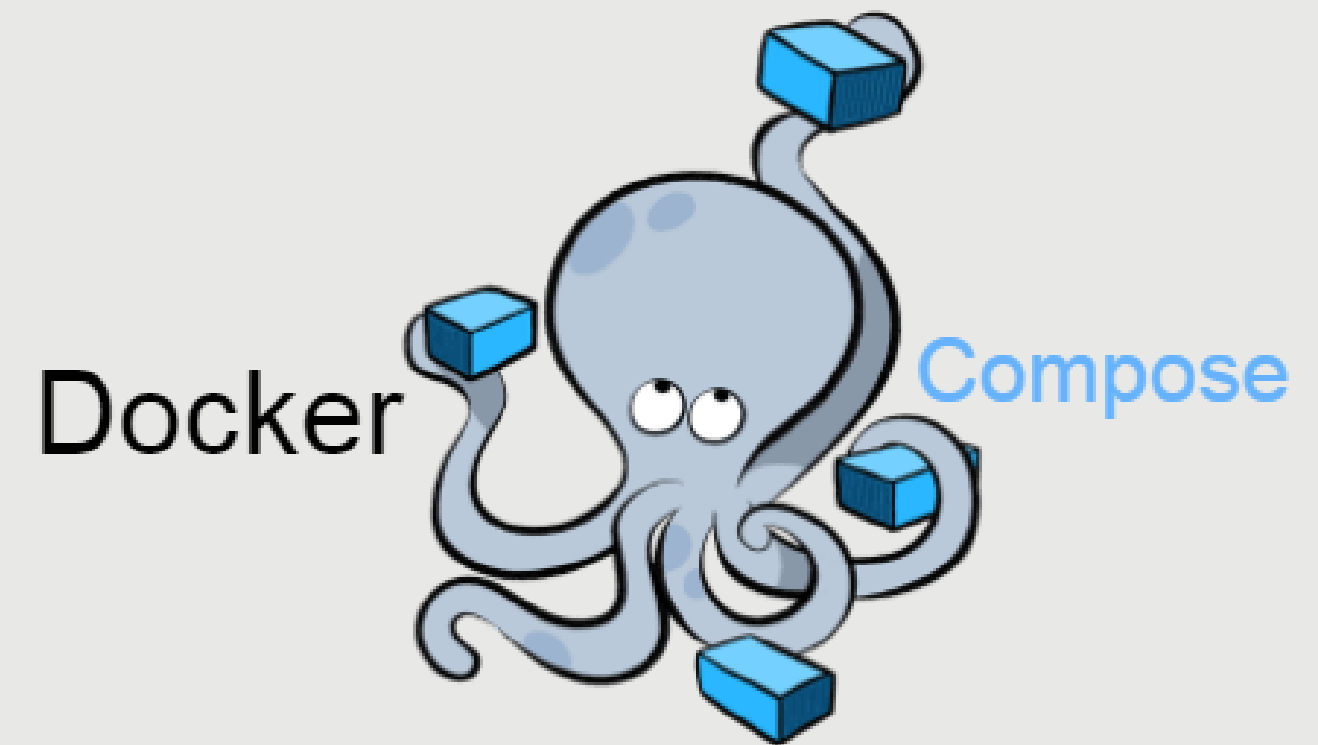
- Mise en place d'une CI pour le backend.
- Plusieurs actions sont effectués:
  - Tests avec Gradle
  - Build avec Gradle
  - Build des images Docker
  - Push sur la Container Registry de Gitlab
- La CI permet également, à travers le tag de commits Gits, d'automatiquement construire des images Docker taggées, ce qui permet de sortir des versions de l'application.





# Déploiement en Production

- Un fichier `docker-compose.prod.yml` a été mis en place pour le déploiement sur un serveur de production.
- Il suffit de lancer Docker Compose avec ce fichier, et celui-ci téléchargera automatiquement la dernière version des images depuis la Container Registry de GitLab.
- Des scripts Bash ont également été créés pour configurer et remplir les bases de données.



# Gestion de Projet

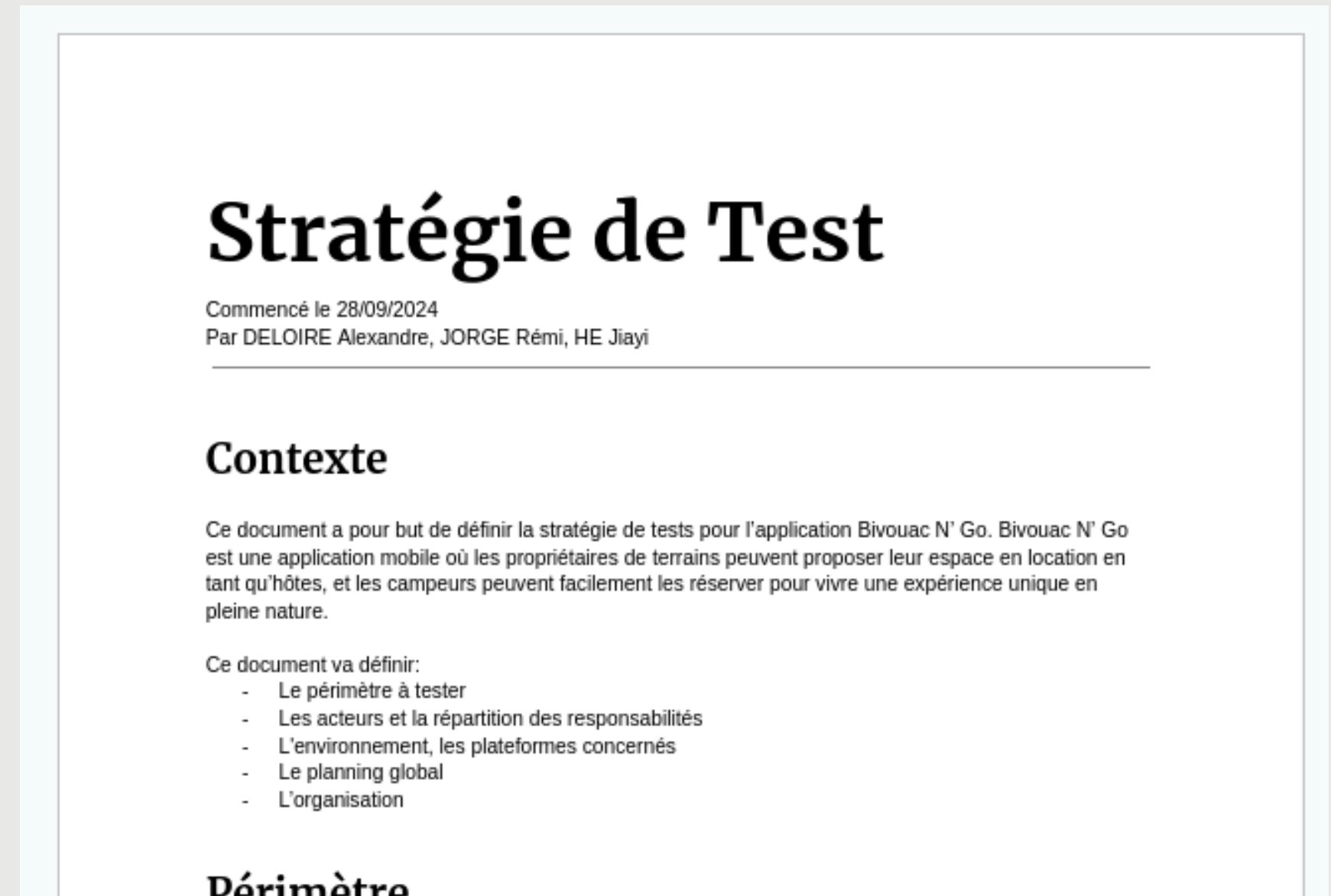
- Répartition du travail : Chaque membre de l'équipe a été responsable d'un microservice (User, Location ou Reservation) ainsi que des pages frontend associées.
- Communication : Utilisation d'un serveur Discord avec différents salons pour chaque phase du projet.
- Organisation : Réunions quotidiennes pour suivre l'avancement du projet.
- Résolution des conflits : Les conflits ont été résolus par des revues de code régulières.
- Ambiance : Très agréable. En tant que trois amis, nous avons l'habitude de travailler ensemble, ce qui nous a permis d'être efficaces.

# Stratégie de Tests

Elaboration d'un document formel pour définir la stratégie qu'on doit employer tout au long du projet.

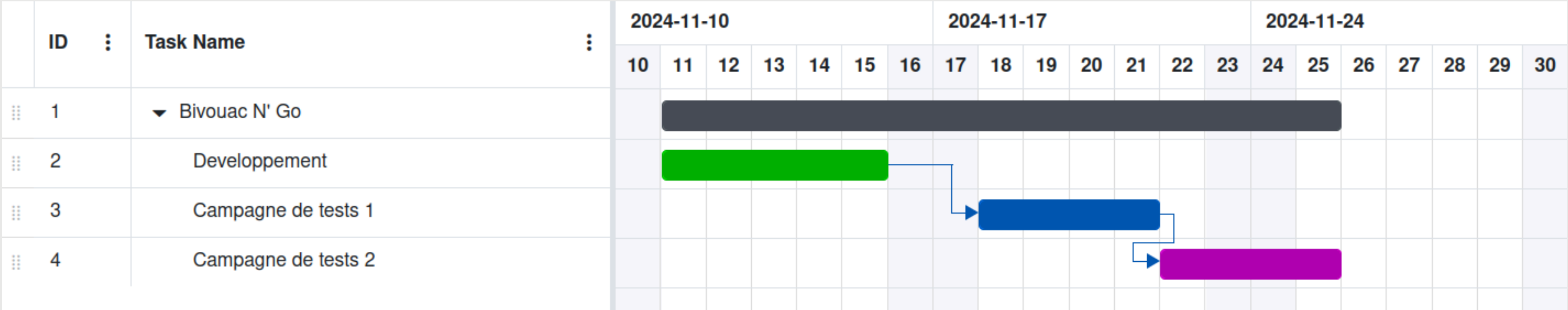
Définition de:

- Périmètre de test
- Environnement de test
- Acteurs
- Planning
- Méthodologie



<https://docs.google.com/document/d/1wOuxwoMmKC9myeUgDpTerM0Tffvf4Fsz3gocmcxGB2w/edit?usp=sharing>

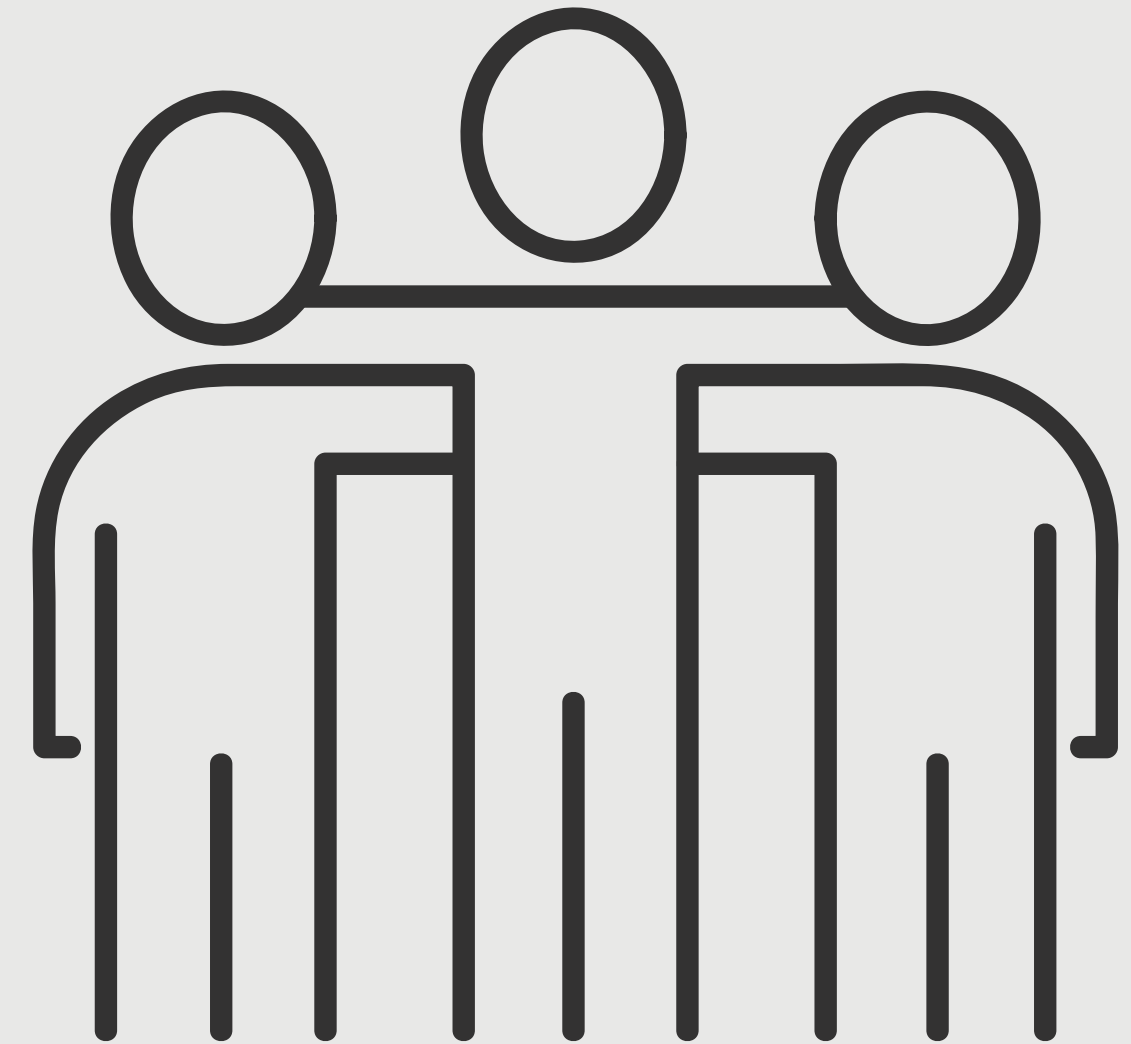
# Organisation Campagnes de Tests



# Tests Statiques

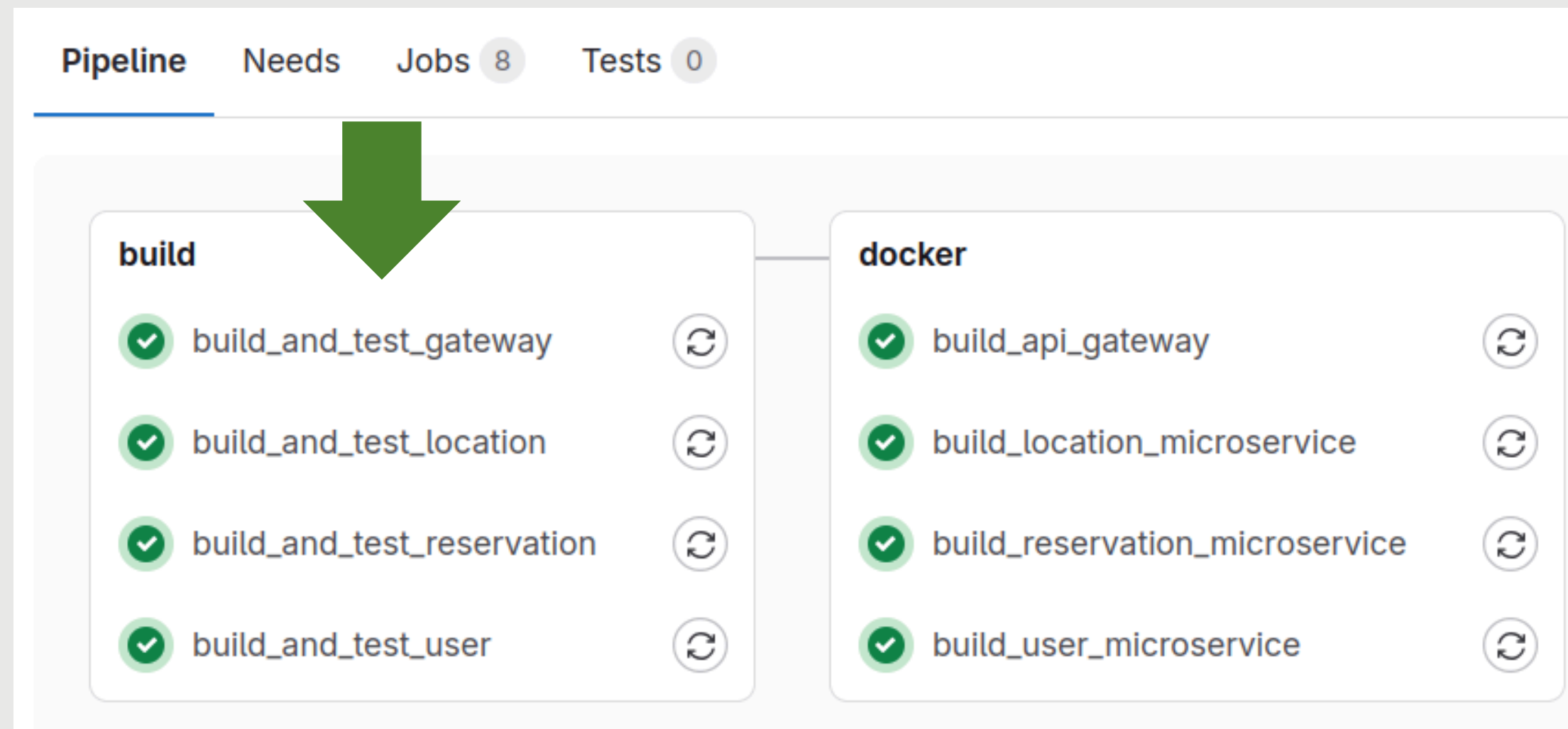
Mesures prises:

- Relecture des commits sur Gitlab
- Relecture de code
- Peer programming



# Tests Dynamiques avec la CI

Exécution automatique des tests unitaires avec gradle lors de la CI





# Versioning dans la CI

On doit pouvoir fixer des versions pour les campagnes de tests

```
build_location_microservice:
  stage: docker
  image: docker
  script:
    - cd location_microservice
    - |
      if [[ -n "$CI_COMMIT_TAG" ]]; then
        DOCKER_TAG="$CI_COMMIT_TAG"
      else
        DOCKER_TAG=latest
      fi
    - echo "Building and pushing Docker image with tag: $DOCKER_TAG"

    - docker build -t $IMAGE_TAG/location_microservice:$DOCKER_TAG .
    - docker login -u $CI_REGISTRY_USER -p $CI_REGISTRY_PASSWORD $REGISTRY
    - docker push $IMAGE_TAG/location_microservice:$DOCKER_TAG
  only:
    - main
    - tags
```

# Tests Systèmes

Mise en place d'un workflow avec une structure spécifique pour les tests et leur exécution

|                                   |  |  |          |             |   |                     |
|-----------------------------------|--|--|----------|-------------|---|---------------------|
| Nom du test                       | Recherche d'un Bien  |  |          |             |   |                     |
| Executé par                       |  |  |          |             |   |                     |
| Date                              |  |  |          |             |   |                     |
| Nom de l'application              | Bivouac N' Go  |  |          |             |   |                     |
| Version                           |  |  |          |             |   |                     |
| Environement                      | Android Samsung S11 Expo Go  |  |          |             |   |                     |
|                                   |  |  |          |             |   |                     |
|                                   |  |  |          |             |   |                     |
| Action                            | Vérification   | Infos Extra  | Resultat | Commentaire | PRE Pièces jointes  | POST Pièces jointes |
| Cliquer sur la section Explore    | La section Explore doit s'ouvrir   |  |          |             |   |                     |
|                                   | Une barre de recherche doit apparaitre en haut de l'écran                      |  |          |             |   |                     |
|                                   | Une liste de proposition de biens doit s'afficher                              |  |          |             |   |                     |
| Cliquer sur la barre de recherche | Un formulaire pour choisir les dates et le nombre de personnes doit s'afficher | Il faut que l'interface ressemble a celui de la pièce jointe |          |             | <a href="https://drive.google.com/file/d/1a1dWoV30DIQqYMjoODFq0koEEtHoUnj4/view?usp=drive_link">https://drive.google.com/file/d/1a1dWoV30DIQqYMjoODFq0koEEtHoUnj4/view?usp=drive_link</a> |                     |
| Cliquer sur la premiere date      | Un calendrier doit s'ouvrir  |  |          |             |   |                     |
| Selectionner la date d'arrivé     | La date doit devenir selectionnée en vert                                      | Prendre le premier janvier 2025                              |          |             |   |                     |
| Selectionner la date de retour    | La date doit devenir selectionnée en vert                                      | Prendre le 15 janvier 2025                                   |          |             |   |                     |
|                                   | Les dates entre celle de départ et celle d'arrivée doivent etre selectionnées  |  |          |             |   |                     |
| Renseigner le nombre de personnes | On doit pouvoir indiquer le nombre de personnes                                | Remplissez 2   |          |             |   |                     |
| Appuyer sur le bouton rechercher  | Un liste de biens qui réponde au critères doivent s'afficher                   |  |          |             |   |                     |
|                                   | Les critères doivent s'affihcer en haut avec un croix a coté                   |  |          |             |   |                     |
| Appuyer sur la croix              | Le filtre doit disparaitre   |  |          |             |   |                     |

[https://drive.google.com/drive/folders/1dZYpZ11Rum3F4gMXSM\\_komdqS527e-q6?usp=drive\\_link](https://drive.google.com/drive/folders/1dZYpZ11Rum3F4gMXSM_komdqS527e-q6?usp=drive_link)

# Première Campagne de Tests

- Sortie de la verison v1.0 avec tag sur la CI
- Execution de la première campagne de tests sur la version v1.0

| Action                                      | Vérification   | Infos Extra                            | Resultat | Commentaire                               |
|---|--|--|----------|---|
| Connectez vous avec un utilisateur admin    | Vous devez pouvoir vous connecter  | username: "user1" password: "password" | OK       |   |
| Cliquer sur le bouton profil                | La page profil doit s'afficher   |  | OK       |   |
|   | Le bouton ADMIN PAGE doit etre visible   |  | OK       |   |
| Cliquer sur le bouton ADMIN PAGE            | La page admin doit s'afficher  |  | OK       |   |
|   | Il doit y avoir deux tabs, une pour user et une pour rentals en haut de la page              | Voir screen en piece jointe            | OK       |   |
|   | On doit etre sur le tab gestion user   |  | OK       |   |
|   | Deux listes distinctes doivent s'afficher, une pour user et une pour les demandes            |  | OK       |   |
| Appuyer sur Show All de la liste user       | Tous les users doivent s'afficher  |  | OK       |   |
| Appuyer sur Hide de la liste user           | Au plus 3 users doivent etre affichés  |  | OK       |   |
| Appuyer sur Show All de la liste demandes   | Tous les demandes doivent s'afficher   |  | OK       |   |
| Appuyer sur Hide de la liste demandes       | Au plus 3 demandes doivent etre affichées  |  | OK       |   |
| Cliquer sur le tab Rentals                  | Deux listes distinctes doivent s'afficher, une pour les equipements et une pour les services |  | OK       |   |
| Appuyer sur Show All de la liste equipement | Tous les equipements doivent s'afficher  |  | NOK      | Le bouton ne marche pas, rien ne se passe |
| Appuyer sur Hide de la liste equipement     | Au plus 3 equipements doivent etre affichés  |  | SKIP     | A cause de l'étape précédente             |
| Appuyer sur Show All de la liste services   | Tous les services doivent s'afficher   |  | OK       |   |
| Appuyer sur Hide de la liste services       | Au plus 3 servicess doivent etre affichées   |  | OK       |   |

[https://drive.google.com/drive/folders/17VGtCLX7a\\_Z\\_5-a7kAEmKDQtQ7hHxu7q?usp=drive\\_link](https://drive.google.com/drive/folders/17VGtCLX7a_Z_5-a7kAEmKDQtQ7hHxu7q?usp=drive_link)











# Rapports et Meeting Tests

- Faire un point entre nous pour discuter de la première campagne et mettre en place un plan pour régler les problèmes.
- Dans la vraie vie: rédaction d'un rapport
- Nous: Channel Discord



# Deuxième Campagne de Tests

- Après les mesures correctives, sortie de la version v1.1 avec tag Cl.
- Exécution de la deuxième campagne de tests pour vérifier que les problèmes sont résolus.
- Cela permet également de vérifier qu'il n'y a pas eu de régressions.

| Name  | ↓  |
|---|--|
|    | IWA_Test_Rental     |
|    | IWA_Test_Profil     |
|    | IWA_Test_MyTrips    |
|  | IWA_Test_Explore  |
|  | IWA_Test_Admin    |

[https://drive.google.com/drive/folders/18k0Hv5ZCJDsdsTeeXZDptsbPfDrKp6mv?usp=drive\\_link](https://drive.google.com/drive/folders/18k0Hv5ZCJDsdsTeeXZDptsbPfDrKp6mv?usp=drive_link)

# Difficultés Rencontrées

- Difficultés techniques:
  - Faire marcher un repo multi-modules
  - API Gateway était compliqué à configurer
  - La CI a dû être adaptée à cause des limitations de ressources des serveurs Polytech
  - Le choix des librairies pour que le développement de l'application marche à la fois pour le web et sur téléphone (certains membres du groupe n'ont pas de téléphone Android)
  - Le build du frontend de l'application (manque de ressources CPU et RAM)
- Difficulté organisationnelle:
  - Manque de temps



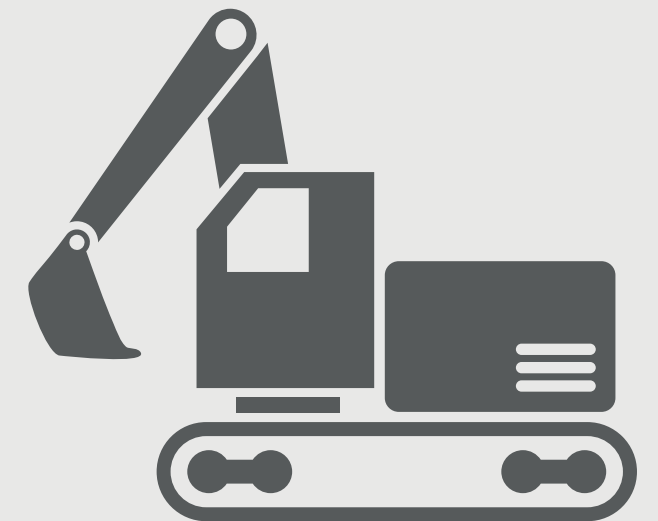
# Perspectives d'Amélioration

- Déployer l'application sur le Play Store: Prendre un serveur pour build l'application.
- Approfondir les vérifications des entrées utilisateur afin de pallier toute malveillance possible (faire des recherches sur la sécurité spécifique aux applications mobile).
- S'outiller en logiciel de tests et embaucher une équipe à part pour faire de la QA (quality analysis).



# Installation sur votre poste

- Toutes les instructions pour l'installation sur votre poste se trouvent dans les fichiers README.md des projets.
- Il est recommandé de commencer par le backend.
- Des scripts ont été mis en place pour que l'installation soit **rapide** et **agréable**.



# Liens GitLab

- Frontend :

<https://gitlab.polytech.umontpellier.fr/alexandre.deloire01/camping-android-app>

- Backend :

<https://gitlab.polytech.umontpellier.fr/alexandre.deloire01/camping-microservices-app>

# L'EQUIPE DE BIVOUAC N' GO VOUS REMERCIE !



*Bivouac N' Go*