

Dossier de conception

*Projet Dijonons*

***Expert (enseignants) :*** *Abdallah Bratos*

*Luc Chmielowski*

*Gaultier Large*

***Titulaire (équipe Devops) :*** *GARES Sebastien*

*PAYET Damien*

*DESPIERRES Benoît*

*NEDELEC Baptiste*

*GUYON Antoine*

***Date de rédaction******:*** *28/06/2023*

# Historique des révisions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Référence** | **Révision** | **Date** | **Auteur(s)** | **Nature de la révision** |
| Dossier de Conception | 00 | 28/06/2023 | PAYET Damien | *Version projet initiale* |
| Ajout des MCD | 01 | 29/06/2023 | PAYET Damien | *Ajout des MCD* |

# 1. Domaine d’application

## 1.1. Objectifs de l’application

"Dijonons" est une application de rencontres locales conçue spécifiquement pour les résidents de la région dijonnaise qui souhaitent rencontrer de nouvelles personnes partageant des intérêts similaires pour sortir et profiter de diverses activités. Cette application permet aux utilisateurs de découvrir des partenaires de sortie potentiels et de participer à des événements locaux passionnants, tout en favorisant des interactions sociales authentiques.

Les objectifs de Dijonons sont les suivants :

* Rencontres locales basées sur les centres d'intérêt : L'objectif principal de l'application est de faciliter des rencontres entre des individus partageant des centres d'intérêt communs. Les utilisateurs peuvent explorer les profils d'autres personnes dans la région dijonnaise et se connecter avec celles qui ont des affinités pour les activités et les loisirs qu'ils apprécient.
* Sorties et activités locales : L'application propose un large éventail d'événements et d'activités locales, tels que des concerts, des expositions, des randonnées, des dégustations de vin, des rencontres sportives, etc. Les utilisateurs peuvent rechercher des partenaires de sortie pour ces événements et organiser des rencontres en fonction de leurs intérêts communs.
* Sécurité et convivialité : L'application met l'accent sur la sécurité et la convivialité. Les utilisateurs peuvent créer des profils détaillés, spécifiant leurs centres d'intérêt et leurs préférences. Des fonctionnalités de messagerie sécurisée permettent aux utilisateurs d'interagir et de faire connaissance avant de se rencontrer.
* Suggestions de lieux et promotions locales : "Dijonons" propose également un système de suggestions de lieux pour les rencontres. Les utilisateurs peuvent découvrir de nouveaux endroits passionnants pour se rencontrer dans la région dijonnaise. De plus, l'application offre des promotions exclusives dans les établissements partenaires, permettant aux utilisateurs de bénéficier d'offres spéciales lors de leurs sorties.

L'architecture souhaitée pour "Dijonons" est la suivante :

* "Dijonons" est une application mobile développé en React Native disponible sur Android et iOS, offrant une expérience conviviale et intuitive aux utilisateurs.
* Une application web Angular est également disponible pour les administrateurs de l'application, leur permettant de gérer les fonctionnalités et le contenu de "Dijonons".
* Enfin, une troisième application mobile en Kotlin, "Hobbify SMS", est utilisée pour l'envoi de SMS à des fins d'authentification des utilisateurs.
* Un BackEnd Micro service accessible depuis une API Gateway Ocelot, utilisant la technologie C# ASP.NET .Net7.

Cette architecture globale garantit une expérience fluide et sécurisée pour les utilisateurs de "Dijonons", tout en offrant aux administrateurs les outils nécessaires pour gérer efficacement l'application.

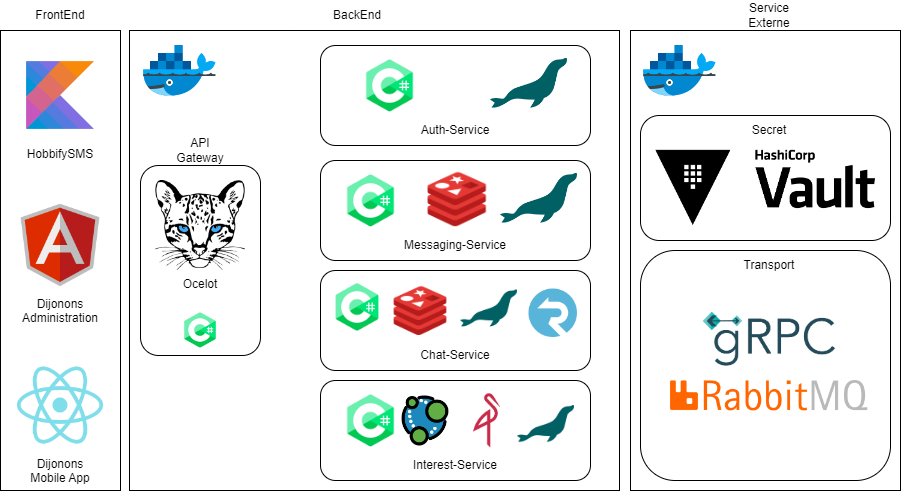


Figure 1: architecture logiciel et technologie

## 1.2. Interfaces

### 1.2.1. Interface Utilisateur

"Dijonons" est une application mobile disponible sur les plateformes Android et iOS, offrant une expérience utilisateur optimale. L'application est entièrement responsive, ce qui signifie qu'elle s'adapte automatiquement aux différents appareils sur lesquels elle est utilisée, et elle tire parti des fonctionnalités natives de ces appareils. Par exemple, sur Android et iOS, l'application peut utiliser l'explorateur de fichiers pour faciliter le téléchargement de fichiers.

L'application comprend plusieurs pages principales, dont voici une liste :

* Connexion / Enregistrement / Mot de passe oublié : Cette partie regroupe les fonctionnalités liées à la connexion à l'application. Les utilisateurs peuvent saisir leur numéro de téléphone pour recevoir un code d'authentification par SMS, qui est valable pendant 60 secondes. S'ils n'ont pas de compte, ils peuvent s'inscrire depuis cette page en créant un compte et en renseignant leurs centres d'intérêt.
* Swipe : La page Swipe est la fonctionnalité clé de l'application "Dijonons". C'est sur cette page que les profils des utilisateurs à rencontrer sont proposés. Les utilisateurs peuvent faire glisser les profils vers la droite pour montrer leur intérêt (like) ou vers la gauche pour montrer leur désintérêt (dislike). Depuis cette page, les utilisateurs peuvent également visualiser les photos des profils proposés, obtenir des informations sur les centres d'intérêt des personnes et lire une brève description si elle est renseignée.
* Profil utilisateur : Cette page permet aux utilisateurs de consulter et de mettre à jour leur propre profil. Ils peuvent ajouter des photos, modifier leurs centres d'intérêt et écrire une description personnelle.
* Messages : La fonctionnalité de messagerie permet aux utilisateurs de communiquer entre eux une fois qu'ils ont établi une correspondance. Les utilisateurs peuvent envoyer des messages, et organiser des sorties directement depuis cette page.
* Suggestions de lieux et promotions : "Dijonons" propose également des suggestions de lieux de sortie en fonction des centres d'intérêt des utilisateurs. Les utilisateurs peuvent découvrir de nouveaux endroits passionnants dans la région dijonnaise et bénéficier de promotions exclusives dans les établissements partenaires.

Cette architecture de l'application "Dijonons" offre une expérience utilisateur fluide et intuitive, en mettant l'accent sur la facilité d'utilisation, la convivialité et la possibilité d'explorer de nouvelles rencontres et activités locales.

### 1.2.2. Interface Administrateur

"Dijonons Administration" est une plateforme web accessible depuis un ordinateur, une tablette ou un mobile. Cette application est dédiée aux personnes autorisées à administrer l'application "Dijonons".

L'interface de "Dijonons Administration" offre différentes fonctionnalités de gestion, notamment :

* Accès au reporting : Cette fonctionnalité permet de consulter des rapports détaillés sur l'utilisation de l'application "Dijonons". Les administrateurs peuvent accéder à des statistiques sur le nombre d'utilisateurs actifs, les rencontres organisées, les centres d'intérêt les plus populaires, etc. Ces informations peuvent être utilisées pour analyser les tendances et prendre des décisions éclairées pour améliorer l'expérience des utilisateurs.
* Gestion des lieux de sortie : Les administrateurs de "Dijonons" peuvent gérer la liste des lieux de sortie proposés aux utilisateurs. Ils peuvent ajouter de nouveaux lieux, mettre à jour les informations existantes (adresse, horaires, description, etc.) et supprimer des lieux qui ne sont plus pertinents. Cette fonctionnalité garantit que les utilisateurs ont accès à une liste actualisée et variée de lieux pour leurs rencontres.
* Gestion de l'outil d'envoi de SMS : "Dijonons" utilise un outil d'envoi de SMS pour l'authentification des utilisateurs. Les administrateurs peuvent gérer les paramètres de cet outil, tels que les appareils autorisés à envoyer des SMS, les messages personnalisés, etc. Ils peuvent également effectuer des tests d'envoi de SMS pour s'assurer que le système fonctionne correctement.

### 1.2.3. Interface ClientSMS

Pour garantir une sécurité renforcée sans l'utilisation de mots de passe, "Dijonons" a mis en place un système d'authentification par envoi de SMS. Ce système utilise une solution développée en interne qui transforme un périphérique Android équipé d'une carte SIM et d'un forfait mobile en un client SMS capable d'envoyer des messages à la demande et d'être géré. L'application "HobbifySMS" est une application Android disponible sur le Play Store. Elle propose une interface simple avec un bouton de connexion/déconnexion et une section de configuration permettant de renseigner les informations relatives au serveur auquel l'application doit se connecter.Cette solution d'envoi de SMS offre plusieurs avantages en termes de sécurité et de commodité pour les utilisateurs de "Dijonons". En évitant l'utilisation de mots de passe, elle réduit les risques de piratage et d'accès non autorisé. De plus, elle simplifie le processus d'authentification en envoyant un code unique par SMS, permettant ainsi aux utilisateurs de se connecter rapidement et en toute sécurité à l'application. L'application "HobbifySMS " constitue un élément essentiel de l'architecture de "Dijonons", permettant la gestion efficace et sécurisée de l'authentification des utilisateurs sans recourir aux mots de passe traditionnels.

Cette solution utilise la technologie WebSocket pour la communication en temps réel. Cependant, par défaut, WebSocket ne prend pas en charge l'équilibrage de charge, ce qui a nécessité une solution adaptée. Un serveur de base de données Redis a été mis en place pour permettre aux différents serveurs de communiquer entre eux et de transmettre les messages via le paradigme Publish/Subscribe.

Grâce à l'intégration de Redis, les serveurs de "Dijonons" peuvent partager des informations sur les connexions WebSocket disponibles. Lorsqu'un utilisateur envoie un message via l'application, le serveur qui reçoit ce message peut le publier sur Redis. Les autres serveurs connectés à Redis, qui gèrent également des connexions WebSocket, sont informés de la disponibilité du message et peuvent le transmettre aux destinataires concernés.

Ce système basé sur le paradigme Publish/Subscribe permet une communication efficace et synchronisée entre les différents serveurs de "Dijonons". Il garantit que les messages sont acheminés de manière fiable aux destinataires appropriés, quel que soit le serveur auquel ils sont connectés.

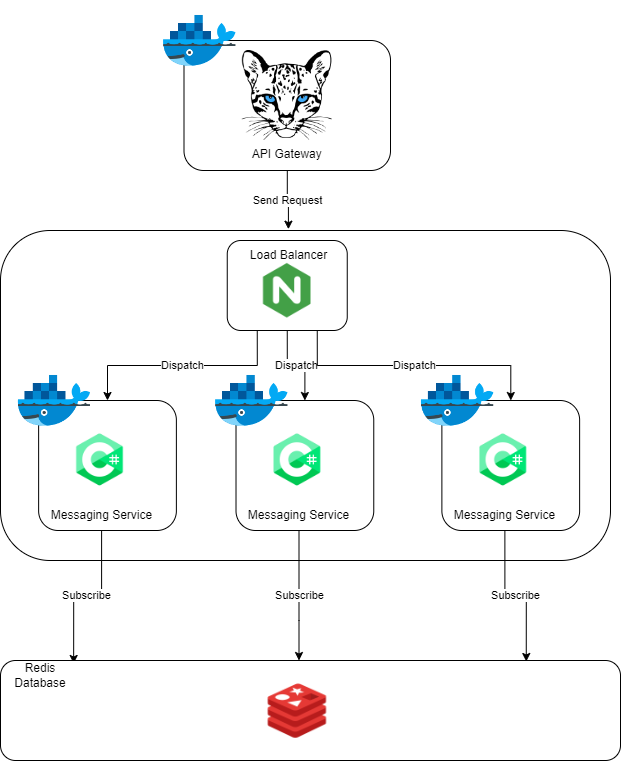
L'utilisation de Redis dans cette architecture renforce la scalabilité et la robustesse de l'application "Dijonons", en permettant une gestion optimale des connexions WebSocket et de la transmission des messages en temps réel.

Figure 2: Représentation de l'architecture mise en place

## 1.3. Contrainte générale de conception

"Dijonons" met en œuvre un ensemble de contraintes et de choix technologiques pour assurer sa fiabilité et sa sécurité, ainsi que pour offrir une expérience utilisateur optimale. Voici les éléments clés de ces contraintes et choix techniques :

Architecture microservices : "Dijonons" est basé sur une architecture microservices, ce qui permet de découper l'application en plusieurs services indépendants et modulaires. Chaque service est responsable d'une fonctionnalité spécifique, ce qui facilite la maintenance, le déploiement et la scalabilité de l'application.

Clean Architecture : "Dijonons" suit les principes de la Clean Architecture, une approche de développement logiciel qui favorise la séparation des responsabilités et la facilité de test. Cela permet de maintenir un code clair, modulaire et évolutif.

Utilisation d'une base de données de graph pour les recommandations : "Dijonons" utilise une base de données de graph pour les fonctionnalités de recommandation. Cette technologie permet de modéliser les relations entre les utilisateurs et les centres d'intérêt, facilitant ainsi les recommandations précises et personnalisées.

Utilisation de Minio pour le stockage de fichiers : "Dijonons" utilise Minio, une solution open-source et auto-hébergée, pour le stockage des fichiers. Cela offre un contrôle total sur les données des utilisateurs et permet de garantir leur confidentialité.

Déploiement sur du Proxmox : "Dijonons" est déployé sur une infrastructure basée sur Proxmox, une plateforme de virtualisation open-source. Cela permet de gérer efficacement les ressources et d'assurer une haute disponibilité de l'application.

Déploiement avec Docker : "Dijonons" utilise Docker pour le déploiement de ses services. Cette technologie de conteneurisation facilite la portabilité, l'isolation et la gestion des différentes composantes de l'application.

CI/CD : "Dijonons" a mis en place un processus d'intégration continue et de déploiement continu pour garantir des mises à jour rapides et fiables de l'application. Cela permet d'automatiser les tests, les builds et les déploiements, assurant ainsi une meilleure qualité et une plus grande réactivité.

Utilisation de WebSockets : "Dijonons" utilise la technologie des WebSockets pour permettre une communication en temps réel entre les utilisateurs. Cela facilite les échanges instantanés, tels que les chats en direct, et améliore l'expérience de l'utilisateur.

Développement d'une solution personnalisée d'envoi de SMS : "Dijonons" a développé sa propre solution d'envoi de SMS, adaptée à ses besoins spécifiques en matière d'authentification et de communication avec les utilisateurs.

Utilisation de SignalR pour le chat instantané : SignalR est utilisé pour faciliter le chat instantané dans "Dijonons". Cette technologie permet une communication bidirectionnelle et en temps réel entre les utilisateurs, offrant ainsi une expérience de chat fluide et réactive.

Utilisation de Redis pour le stockage temporaire : "Dijonons" utilise Redis comme base de données clé-valeur pour le stockage temporaire de données. Cela permet une récupération rapide des informations et une meilleure performance de l'application.

Utilisation du paradigme Publish/Subscribe : "Dijonons" utilise le paradigme Publish/Subscribe pour la transmission de messages entre les différents services de l'application. Cela permet une communication efficace et asynchrone entre les composantes de l'application.

Utilisation de gRPC : "Dijonons" utilise gRPC pour la communication entre les différents services de l'application. Ce framework de communication à distance permet d'optimiser les performances et la fiabilité des échanges de données.

Utilisation de RabbitMQ : "Dijonons" utilise RabbitMQ comme système de messagerie pour la gestion des messages asynchrones entre les différents services. Cela permet une communication fiable et évite les pertes de données en cas de défaillance temporaire d'un service.

Hébergement sur GitHub : "Dijonons" utilise GitHub comme plateforme d'hébergement pour le code source de l'application. Cela facilite la collaboration entre les développeurs et permet un suivi efficace des versions et des modifications apportées au projet.

Gestion de projet SCRUM : "Dijonons" suit une approche de gestion de projet SCRUM, basée sur des itérations itératives et des cycles de développement courts. Cela favorise la flexibilité, la transparence et l'adaptabilité aux besoins changeants du projet.

Système d'authentification inspiré d'Identity Server : "Dijonons" a développé un système d'authentification inspiré d'Identity Server, une solution open-source pour la gestion de l'authentification et de l'autorisation. Cela permet une gestion sécurisée et centralisée de l'authentification des utilisateurs.

Développement de l'application avec la technologie React Native : L'application "Dijonons" est développée en utilisant la technologie React Native, qui permet de créer des applications mobiles pour Android et iOS à partir d'un seul codebase. Cela garantit une expérience utilisateur cohérente et optimisée sur différentes plateformes.

Développement du backoffice avec Angular : Le backoffice de "Dijonons" est développé en utilisant le framework Angular, qui permet de créer des interfaces administratives interactives et réactives. Cela facilite la gestion et l'administration de l'application.

Développement du client SMS en Kotlin : Le client SMS de "Dijonons" est développé en utilisant le langage de programmation Kotlin. Cela permet de créer une application Android native optimisée pour l'envoi de SMS et intégrée de manière transparente à l'architecture globale de l'application.

Développement du backend en C# : Le backend de "Dijonons" est développé en utilisant le langage de programmation C#. Cela offre une performance élevée, une compatibilité avec les outils Microsoft et une facilité d'intégration avec les autres composantes de l'application.

Ces choix et contraintes technologiques démontrent l'approche avancée et rigoureuse adoptée par "Dijonons" pour fournir une application performante, sécurisée et scalable, tout en offrant une expérience utilisateur fluide et riche en fonctionnalités.

# 2. Normes, standards et outils

## 2.1. Convention de nommage

"Dijonons" respecte des conventions de nommage spécifiques pour ses différentes applications et langages de programmation. Voici comment les conventions de nommage ont été appliquées dans chaque cas :

### Pour C# (backend) :

* **Classes** : Les classes en C# suivent généralement la convention PascalCase, où chaque mot dans le nom de la classe commence par une majuscule. Par exemple, "UserService", "EventController", etc.
* **Propriétés** : Les propriétés en C# suivent la convention PascalCase. Par exemple, "FirstName", "LastName", etc.
* **Méthodes** : Les méthodes en C# suivent également la convention PascalCase. Par exemple, "GetUsers", "CreateEvent", etc.

### Pour Angular (backoffice) :

* **Classes** : En Angular, les classes suivent la convention PascalCase. Par exemple, "UserService", "EventComponent", etc.
* **Fichiers de composants** : Les fichiers de composants sont généralement nommés en kebab-case, où chaque mot est séparé par des tirets. Par exemple, "user-list.component.ts", "event-details.component.html", etc.
* **Sélecteurs de composants** : Les sélecteurs de composants suivent la convention kebab-case dans les fichiers HTML. Par exemple, "app-user-list", "app-event-details", etc.
* **Fichiers de modèles** : Les fichiers de modèles utilisés par les composants peuvent être nommés en camelCase ou en kebab-case, en fonction de la préférence de l'équipe de développement.
* **Propriétés** : Les propriétés en Angular utilisent généralement la convention camelCase, où le premier mot commence par une minuscule et les mots suivants commencent par une majuscule. Par exemple, "firstName", "lastName", etc.
* **Méthodes** : Les méthodes en Angular suivent également la convention camelCase. Par exemple, "getUsers", "createEvent", etc.

### Pour Kotlin (client SMS) :

* **Classes** : En Kotlin, les classes suivent généralement la convention PascalCase. Par exemple, "SmsService", "SmsClient", etc.
* **Propriétés** : Les propriétés en Kotlin utilisent généralement la convention camelCase. Par exemple, "messageText", "phoneNumber", etc.
* **Méthodes** : Les méthodes en Kotlin suivent également la convention camelCase. Par exemple, "sendMessage", "formatPhoneNumber", etc.

Ces conventions de nommage permettent de maintenir une cohérence et une lisibilité dans le code des applications "Dijonons", facilitant ainsi la compréhension et la collaboration entre les développeurs travaillant sur différentes parties de l'application.

## 2.2. Normes et standards de développement

Dans le développement de "Dijonons", plusieurs normes et standards sont utilisés pour assurer la qualité, la cohérence et la fiabilité de l'application. Voici quelques-uns des standards et pratiques couramment utilisés dans le développement de "Dijonons" :

### Convention de codage :

Le respect des conventions de codage spécifiques à chaque langage utilisé, tels que les conventions de codage en C#, Angular, Kotlin et React Native, afin de maintenir un code clair, lisible et cohérent.

### Architecture logicielle :

L'adoption de l'architecture microservices pour découper l'application en modules indépendants, favorisant la maintenabilité et la scalabilité de "Dijonons".

L'utilisation de la Clean Architecture pour définir des couches clairement séparées, facilitant la testabilité et la compréhension du code.

### Gestion de versions :

L'utilisation de Git comme système de contrôle de versions pour suivre et gérer les modifications du code source de "Dijonons".

L'application des bonnes pratiques de gestion des branches, telles que Git Flow, pour permettre un développement parallèle, une intégration continue et une gestion efficace des versions.

### Tests et qualité du code :

L'écriture de tests unitaires pour valider le bon fonctionnement des différentes fonctionnalités de "Dijonons".

L'utilisation de frameworks de tests appropriés pour les langages et les technologies utilisées, tels que NUnit pour les tests unitaires en C# et Jest pour les tests unitaires en JavaScript/TypeScript.

L'intégration de processus d'analyse statique du code, tels que l'utilisation de SonarQube, pour détecter les problèmes de qualité du code et appliquer des bonnes pratiques de développement.

### Sécurité :

La mise en œuvre de bonnes pratiques de sécurité, telles que l'authentification et l'autorisation robustes, la protection contre les attaques de sécurité courantes (injections SQL, cross-site scripting, etc.) et la protection des données sensibles des utilisateurs.

L'utilisation de protocoles de communication sécurisés, tels que HTTPS, pour chiffrer les données échangées entre les utilisateurs et les serveurs.

Ces standards et pratiques dans le développement de "Dijonons" visent à garantir une application de haute qualité, sécurisée et maintenable, tout en favorisant la collaboration au sein de l'équipe de développement.

## 2.3. Outils de développement

Dans le développement de "Dijonons", plusieurs outils et technologies sont utilisés pour faciliter le processus de développement, la collaboration et la gestion du projet. Voici quelques-uns des outils couramment utilisés dans le développement de "Dijonons" :

### Environnement de développement intégré (IDE) :

La suite JetBrains : Les outils de développement JetBrains, tels que Rider, WebStorm et Android Studio, sont utilisés comme environnement de développement intégré pour différentes langues et technologies, notamment pour le développement en Kotlin, JavaScript, TypeScript et C#.

### Contrôle de versions :

Git : Le système de contrôle de versions le plus populaire, utilisé pour suivre et gérer les modifications du code source de "Dijonons".

### Outils de gestion de projets et de collaboration :

Azure DevOps : Une plateforme de gestion de projets et de développement collaboratif utilisée pour planifier, suivre et gérer les tâches, les problèmes et les versions de "Dijonons".

Teams : Pour communiquer

### Outils de tests et d'intégration continue :

NUnit : Un framework de tests unitaires pour le développement en C#, utilisé pour écrire et exécuter des tests unitaires pour valider le bon fonctionnement des fonctionnalités de "Dijonons".

Jest : Un framework de tests JavaScript/TypeScript utilisé pour écrire des tests unitaires pour les composants Angular et React Native de "Dijonons".

### Outils de déploiement et d'infrastructure :

Docker : Une plateforme de conteneurisation qui facilite le déploiement et la gestion des différentes composantes de "Dijonons" de manière isolée et portable.

Proxmox : Une plateforme de virtualisation open-source utilisée pour déployer et gérer l'infrastructure de serveurs de "Dijonons".

Docker Compose : Utilisé pour gérer les conteneurs Docker et l'orchestration des services de "Dijonons" en local ou dans des environnements de développement.

# 3. Conception générale

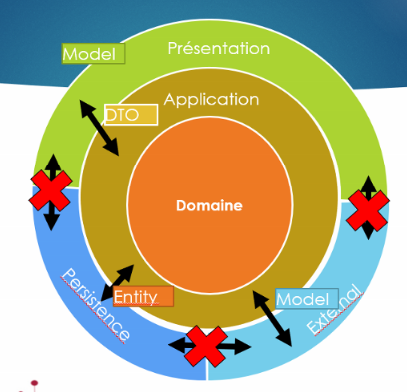
## 3.1. Langages de programmation utilisé

* Application Mobile - Dijonons Mobile App :
  + Framework : React Native
  + Langage : TypeScript
* Application Web - Dijonons Administration :
  + Framework : Angular
  + Langages : TypeScript, HTML, SCSS
* Application Mobile - HobbifySMS :
  + Langage : Kotlin
* Base de données de graph (Neo4j) :
  + Langage de requête : Cypher
* Base de données relationnelle :
  + Langage de requête : SQL (Structured Query Language)
* Application Backend :
  + Langage : C#

## 3.2. Diagramme de déploiement

## 3.3. Architecture logicielle

### 3.3.1. Clean Architecture



Application : La couche Application reste inchangée, elle représente la partie la plus externe de l'architecture et coordonne les interactions entre les différentes couches.

Domain Layer : Cette couche contient le cœur métier de l'application, incluant les entités, les valeurs objets, les agrégats, les services et les règles métier. Elle est indépendante des détails techniques et des frameworks.

Persistence Layer : Cette couche gère la persistance des données et l'accès à la base de données. Elle contient les interfaces et les classes définissant les opérations de lecture, écriture et requête des données. Les implémentations concrètes de ces interfaces se trouvent dans la couche Infrastructure.

Figure 3: Clean Architecture

Infrastructure Layer : Cette couche gère les détails techniques et les dépendances externes de l'application. Elle comprend les implémentations concrètes des interfaces définies dans le Domain Layer et la Persistence Layer. Elle peut inclure les bases de données, les services externes, les frameworks et les outils utilisés par l'application.

Presentation : Cette couche gère l'interface utilisateur et l'interaction avec les utilisateurs. Elle communique avec la couche Application pour obtenir les données et les actions nécessaires à l'affichage et à la manipulation de l'interface utilisateur.

### 3.3.2. Micro-service

Figure 4: Architecture Micro-Service simplifié de l'application

### 3.3.3. Data Storage

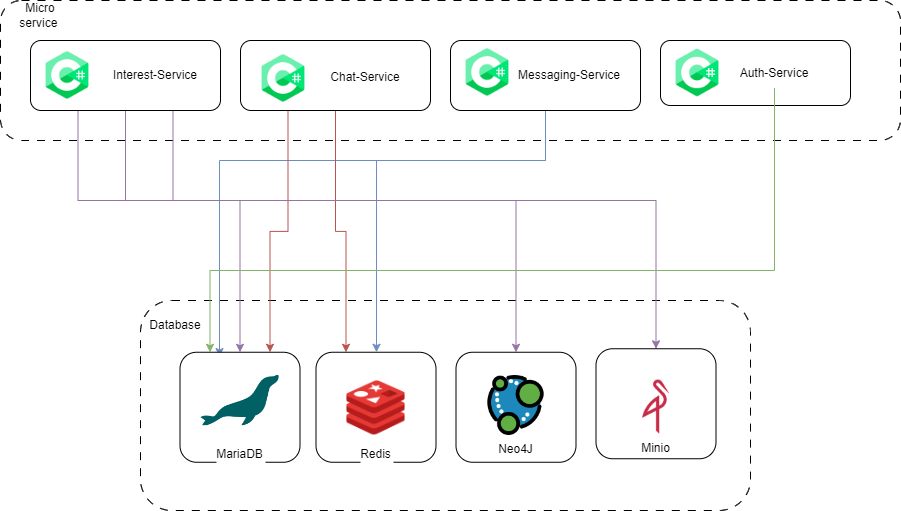


Figure 5: Accès au data / Micro-service

# 4. Conception détaillée des composants

## 4.1. Base de données relationnelle

### 4.1.1. Schéma BDD – Chat Service

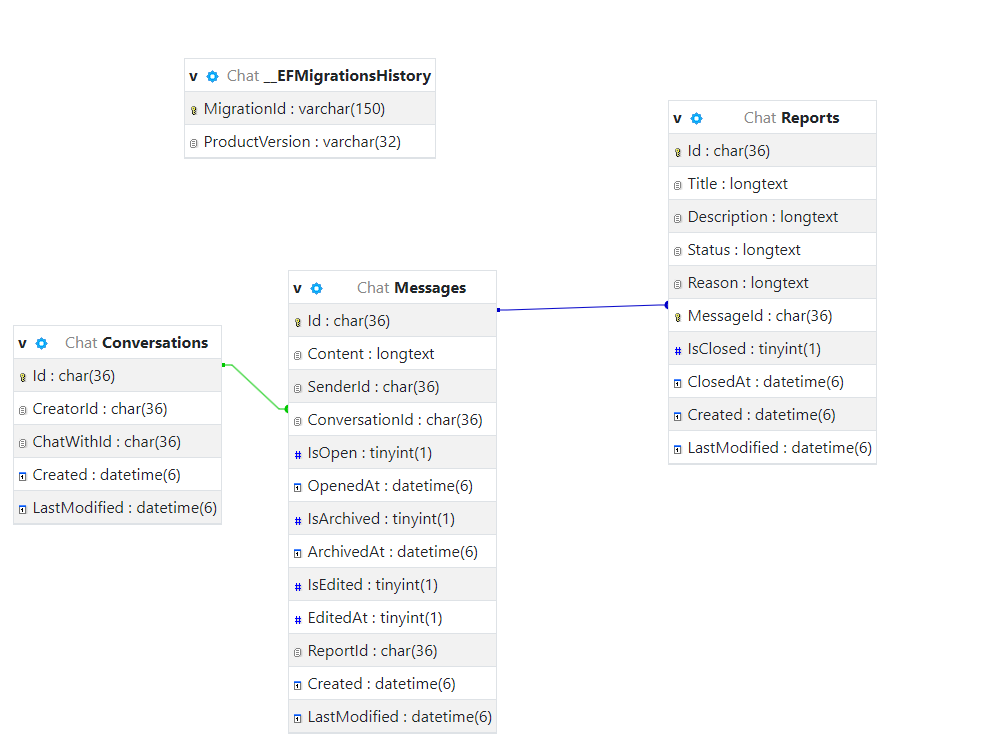


Figure 6: Schéma BDD - Chat Service

### 4.1.2. Schéma BDD - Messaging ServiceUne image contenant texte, capture d’écran, Police, nombre Description générée automatiquement

Figure 7: Schéma BDD - Messaging service

### 4.1.3. Schéma BDD – Interest Service

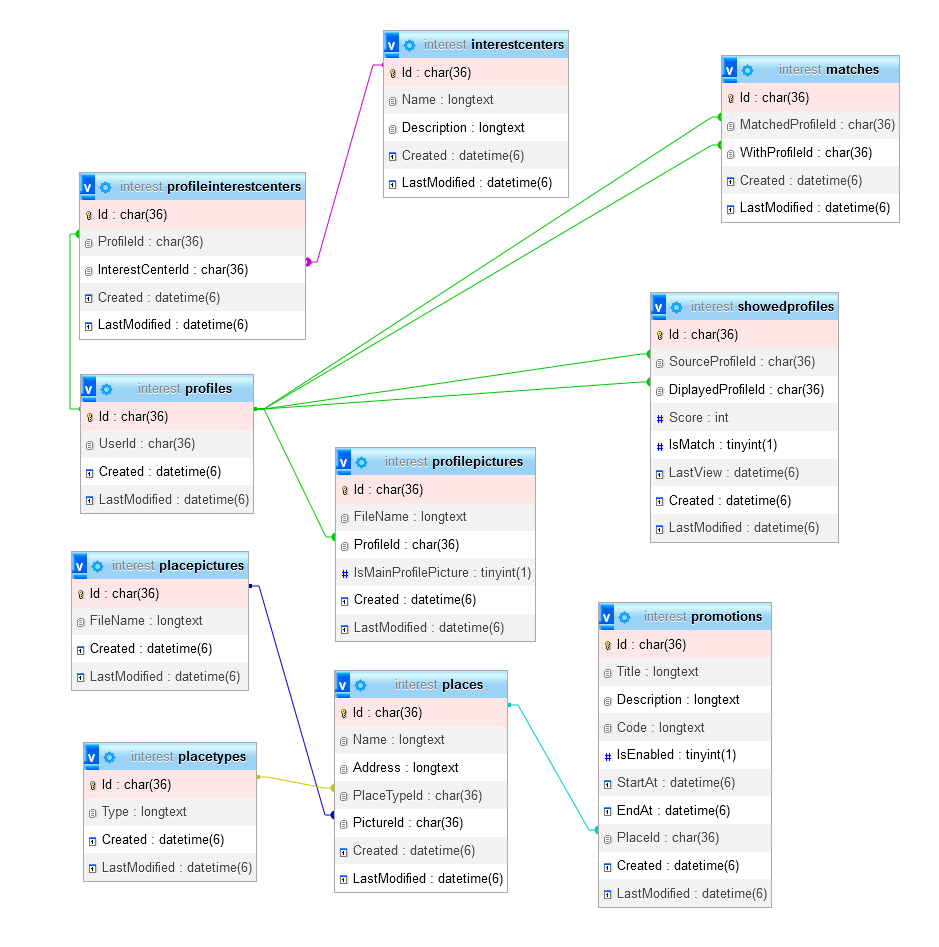


Figure 8: Schéma BDD - Interest Service

### 4.1.4. MCD – Auth Service

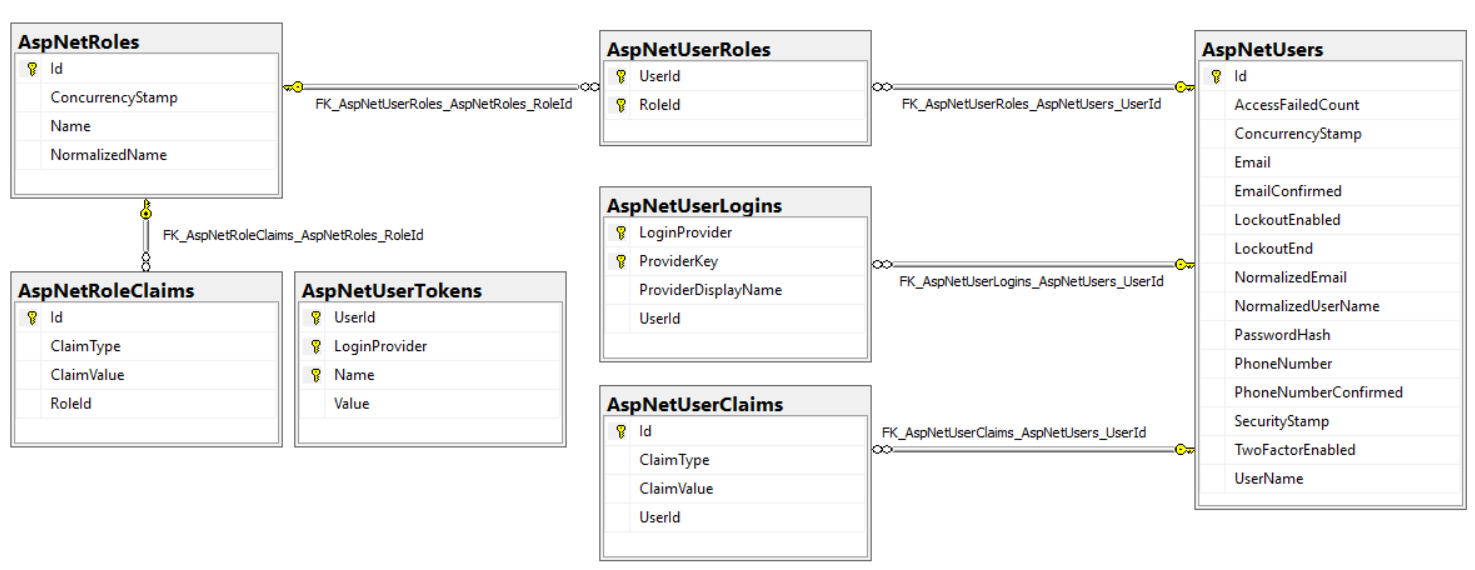


Figure 9: Schéma BDD Identity

## Le service d'authentification (Auth service) utilise le schéma de base de données par défaut de Microsoft Identity.

## Le schéma de base de données par défaut de Microsoft Identity est un schéma préconçu qui est utilisé pour stocker les informations relatives aux utilisateurs, aux rôles et aux autorisations dans un système d'authentification et d'autorisation. Ce schéma est couramment utilisé avec le framework ASP.NET Identity, qui fournit des fonctionnalités de gestion des utilisateurs, de l'authentification et des autorisations dans les applications web basées sur ASP.NET.

## 4.2. Base de données Graph

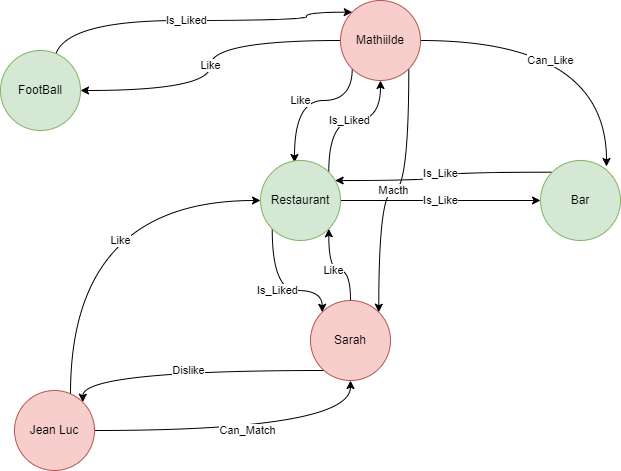


Figure 10: Graph Relation Utilisater <-> Centre d'intéret

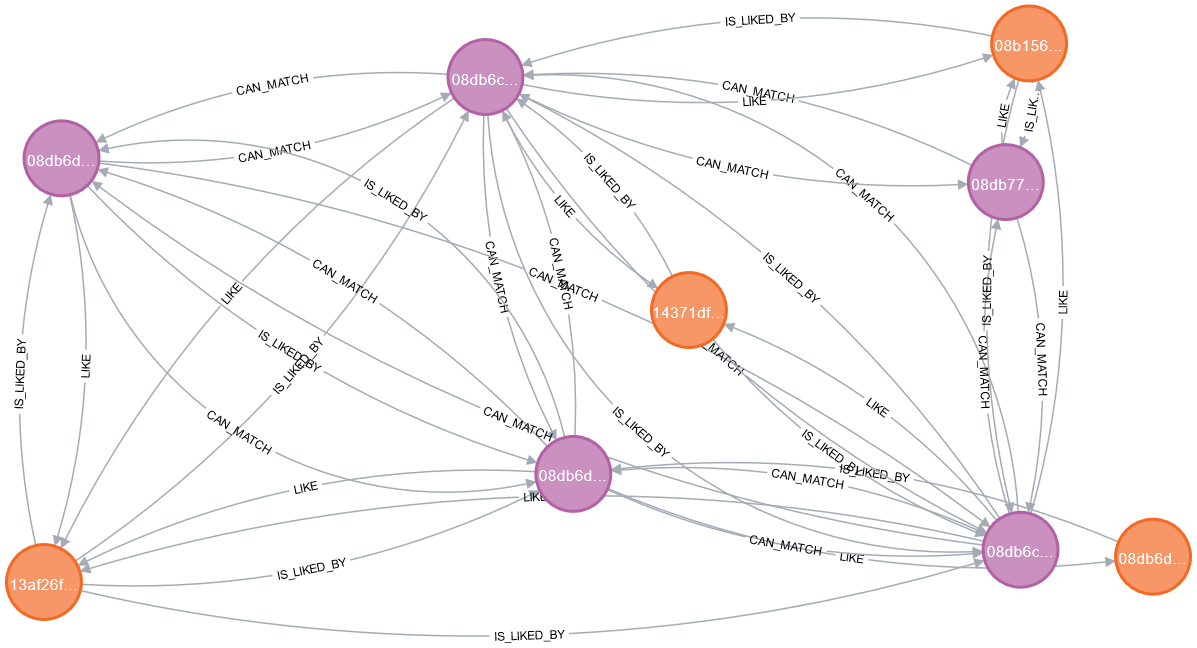


Figure 11: Exemple graph neo4J

# 5. Processus d’authentification

L'application Dijonons est basée sur le système d'authentification Microsoft Identity, qui intègre la norme RFC 6238 pour la génération et la validation des codes d'authentification. Cette norme, également connue sous le nom de TOTP (Time-Based One-Time Password), permet de générer des codes temporaires qui sont utilisés pour renforcer la sécurité lors de l'authentification.

Cependant, Dijonons utilise sa propre solution d'envoi de codes d'authentification via des clients Android disposant d'une carte SIM et d'un forfait téléphonique. Cette solution permet d'envoyer les codes d'authentification directement aux utilisateurs via des SMS, en utilisant les fonctionnalités des appareils mobiles.

En utilisant cette approche, Dijonons garantit une sécurité accrue en combinant l'authentification basée sur la norme RFC 6238 avec l'envoi sécurisé des codes d'authentification via des SMS. Cela permet aux utilisateurs de recevoir rapidement et en toute sécurité les codes d'authentification sur leurs appareils mobiles, renforçant ainsi la protection de leurs comptes et de leurs données.

Il est important de noter que l'utilisation d'une solution d'envoi de codes d'authentification personnalisée nécessite une expertise technique et une mise en œuvre soigneuse pour assurer la fiabilité et la sécurité du processus d'authentification à deux facteurs.

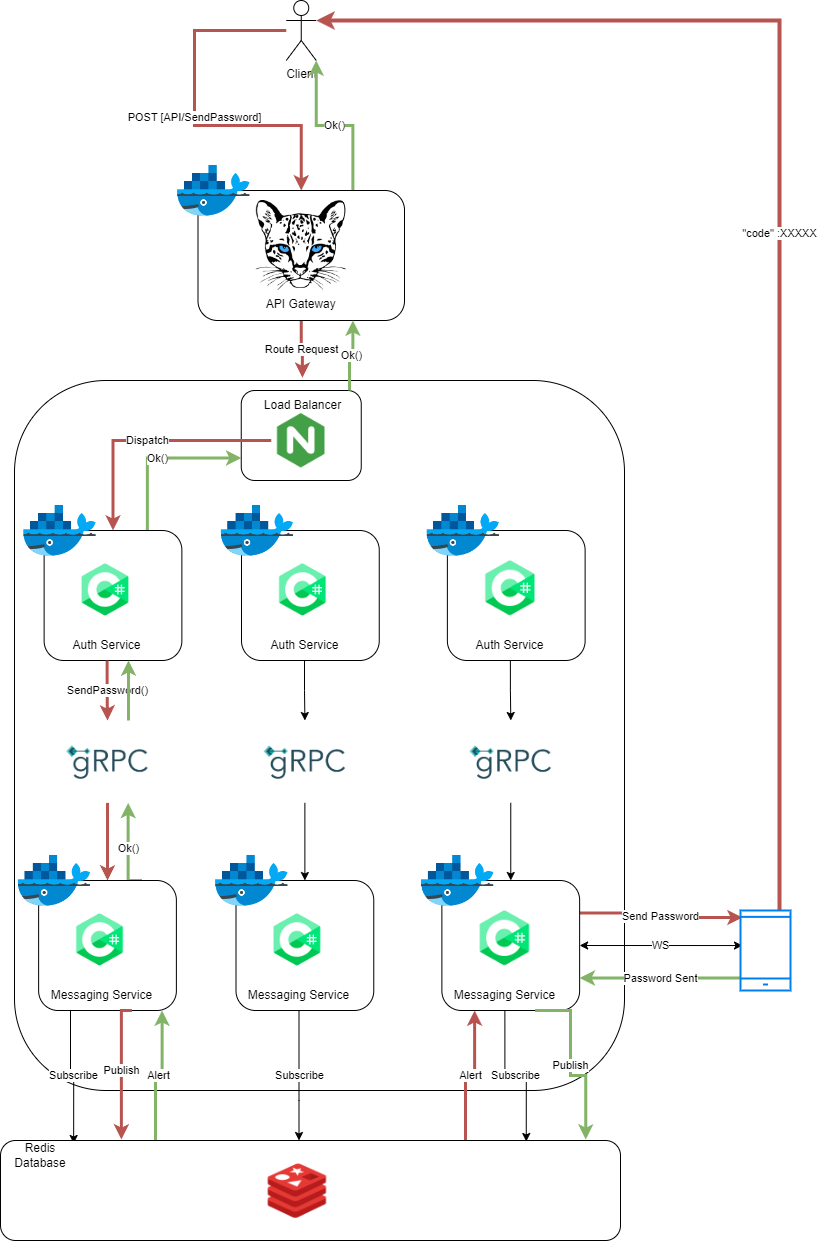


Figure 12: Architecture envoi SMS