



AVIGNON
UNIVERSITÉ

ShazaMurge

Walid Medouaz
Alexis Saccoman

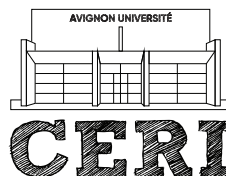
13 décembre 2023

Master Informatique
ILSEN

UE Urbanisation

Responsables d'UE
Yannis Labrak

UFR
SCIENCES
TECHNOLOGIES
SANTÉ



CENTRE
D'ENSEIGNEMENT
ET DE RECHERCHE
EN INFORMATIQUE
ceri.univ-avignon.fr

Sommaire

Titre	1
Sommaire	2
1 Introduction	4
1.1 Contexte du Projet	4
1.2 Objectifs de l'Application	4
1.3 Justification du Besoin	4
2 Manuel d'Utilisation	5
2.1 Installation de l'Application	5
2.2 Utilisation de la Fonction "Shazam des Vins"	5
2.2.1 Écran d'Accueil	5
2.2.2 Écran Wine Card	6
2.2.3 Écran DetailsVin	9
2.2.4 Écran Scanner	11
2.2.5 Écran Settings	12
2.2.6 Écran Login/Register	13
3 Choix de Technologies	14
3.1 Environnement de Développement (IDE, Outils)	14
3.1.1 Langage de Programmation	14
3.1.2 IDE (Environnement de Développement Intégré)	14
3.2 Serveur Web	14
3.2.1 Justification du Choix de Node.js + Express.js	14
3.2.2 Exécution Asynchrone et Événementielle	14
3.3 Base de Données	14
3.3.1 Justification du Choix de MongoDB	14
3.4 Frameworks et Bibliothèques	15
3.4.1 Justification de l'utilisation de Flutter Packages	15
4 Architecture de l'Application	15
4.1 Matrice de Flux	15
4.2 Architecture Technique	15
4.3 Modélisation UML de l'Application côté Client	17
4.4 Modélisation UML de la Base de Données Mongo	18
5 Autres Choix Stratégiques	19
5.1 Choix de Traitements et d'Organisation	19
5.1.1 Traitements côté serveur	19
5.1.2 Code-barres EAN-13	19
5.1.3 Organisation de la base de données et ORM	19
5.1.4 Sécurité et évolutivité	19
5.2 Choix d'Interface Utilisateur (UX)	20
5.3 Gestion des Erreurs et Retours Utilisateur	20

6	Montée en Charge	20
6.1	Optimisation du Code Flutter/Dart	20
6.2	Caching côté Client	20
6.3	Compression de Données	21
6.4	Utilisation de la base Mongo du serveur pedago	21
6.5	Utilisation de Redis	21
6.6	Répartition de Charge	22
6.7	Tests de charge	23
7	Conclusion	24
7.1	Récapitulation des Points Clés	24
7.2	Perspectives Futures	24

1 Introduction

1.1 Contexte du Projet

Le présent rapport expose le résultat du développement d'une application mobile innovante répondant à un besoin et un intérêt croissant dans le domaine de la découverte des vins et de l'oenologie dans un cadre plus large. L'objectif principal est de permettre aux utilisateurs d'obtenir des informations détaillées sur une bouteille de vin en capturant simplement le code-barre de son étiquette.

1.2 Objectifs de l'Application

Notre application vise à fournir une expérience conviviale et enrichissante aux amateurs de vin, en offrant un accès instantané à des fiches techniques détaillées, aux notes et commentaires des utilisateurs. De plus, l'intégration d'une technologie de reconnaissance d'image permettra une exploration rapide et intuitive des caractéristiques de chaque cuvée.

1.3 Justification du Besoin

Le marché des applications liées au vin est en constante expansion, et il existe un besoin croissant de solutions innovantes simplifiant l'accès à des informations précises sur les vins. Notre application vise à combler ce fossé en intégrant une technologie de pointe tout en offrant une plateforme d'interaction communautaire pour les passionnés de vin.

2 Manuel d'Utilisation

2.1 Installation de l'Application

Pour installer notre application, suivez ces étapes simples :

1. Téléchargez l'application à partir du Google Play Store (pour Android) ou de l'App Store (pour iOS).
2. Lancez l'application après l'installation.
3. Bienvenue sur ShazaMurge, suivez les astuces du Manuel d'Utilisation

2.2 Utilisation de la Fonction "Shazam des Vins"

2.2.1 Écran d'Accueil

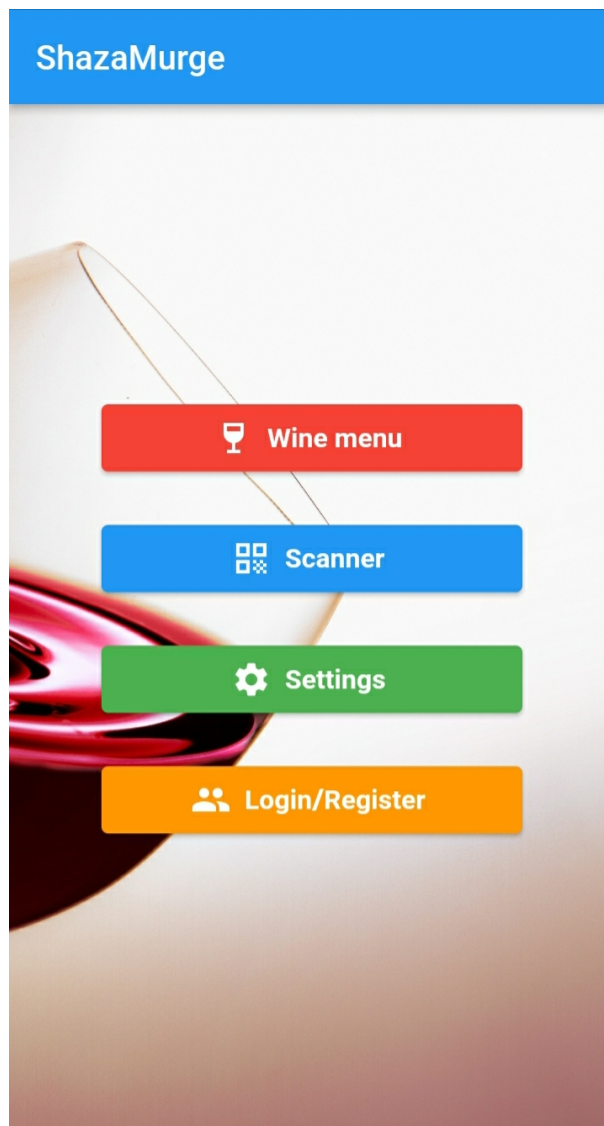


Figure 1. Ecran d'Accueil

L'écran d'accueil est le point central de l'application. Vous trouverez ici quatre boutons vous donnant accès aux fonctionnalités principales de l'application.

- **Wine Card** : Permet de visualiser la liste des vins enregistrés. Chaque carte (représentant un vin) peut être supprimée par un administrateur. En cliquant sur une carte, vous accédez à l'écran *DetailsVin*.
- **Scanner** : Active le scanner de code-barres pour vérifier si une bouteille est dans la base. Si un vin est trouvé, vous accédez à l'écran *DetailsVin*. Si aucun vin n'est trouvé, rien ne s'affiche et vous retournez sur l'écran initial.
- **Settings** : Permet de modifier le nom d'utilisateur et le mot de passe, ainsi que de se déconnecter avec le bouton *Logout*.
- **Login/Register** : Redirige vers l'écran de connexion ou d'inscription.

2.2.2 Écran Wine Card

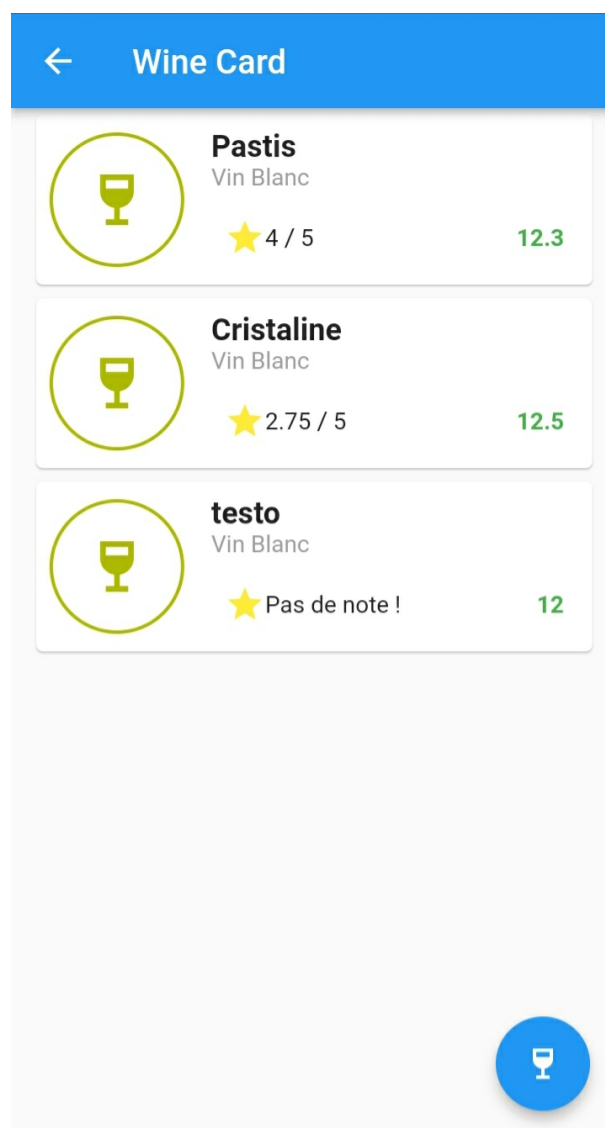


Figure 2. Ecran Wine Card

Cet écran affiche la liste des vins sous forme de cartes. Les administrateurs peuvent supprimer ou ajouter un vin en utilisant l'icône de la poubelle ou le bouton de verre rond en bas à droite de l'écran. En cliquant sur une carte, vous accédez à l'écran *DetailsVin*.

Voici ci-dessous la vue administrateur et la vue pour ajouter un vin :

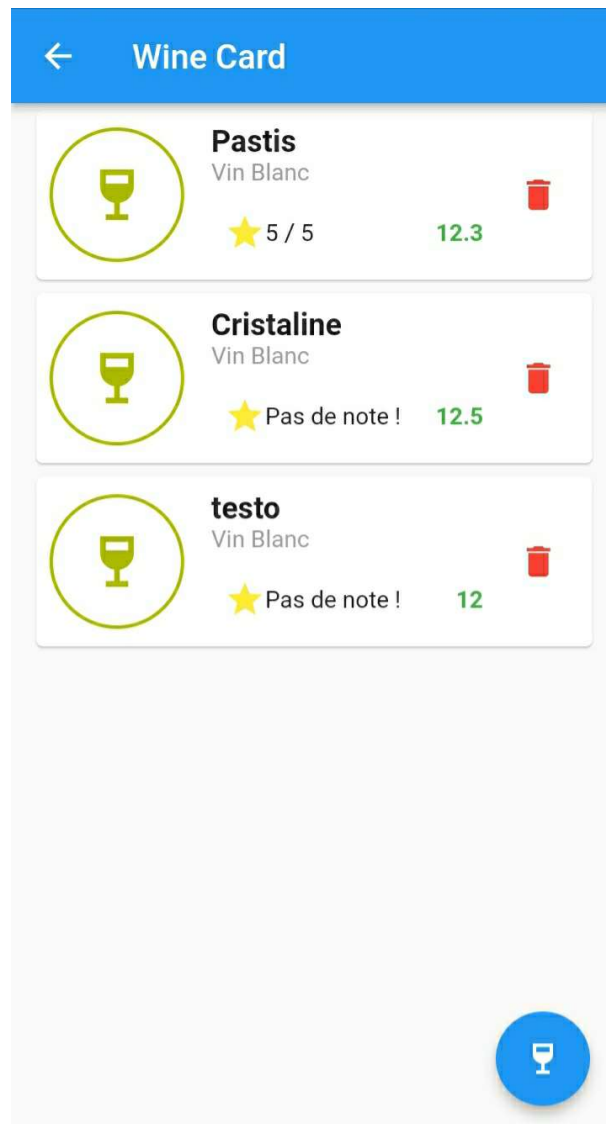
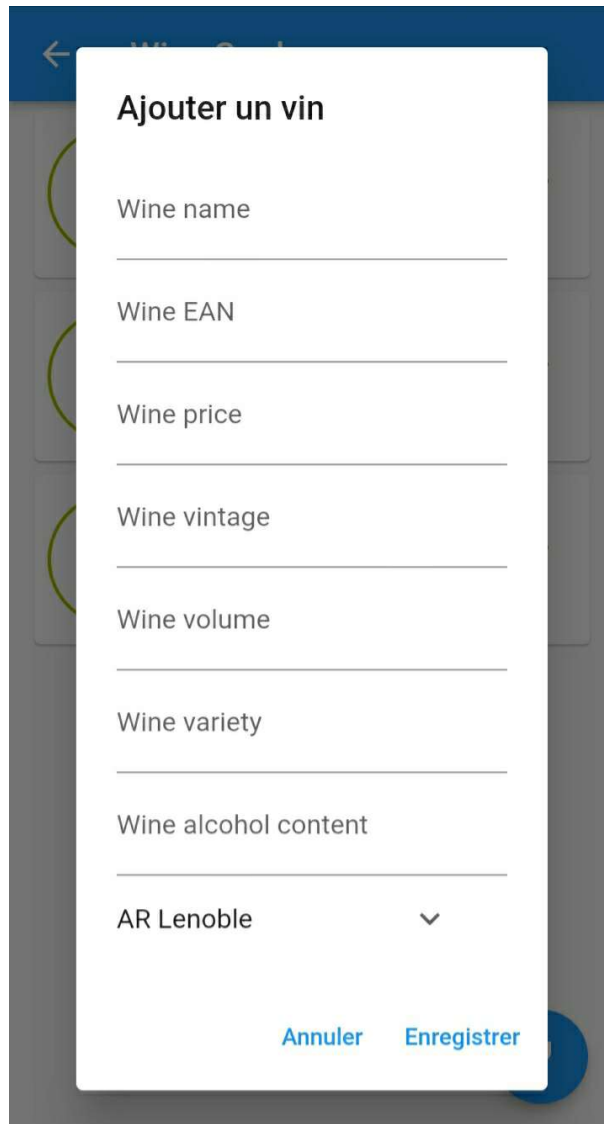


Figure 3. Ecran Wine Card Administrateur



Ajouter un vin

Wine name

Wine EAN

Wine price

Wine vintage

Wine volume

Wine variety

Wine alcohol content

AR Lenoble ▼

[Annuler](#) [Enregistrer](#)

Figure 4. Ajouter un vin par un Administrateur

2.2.3 Écran DetailsVin

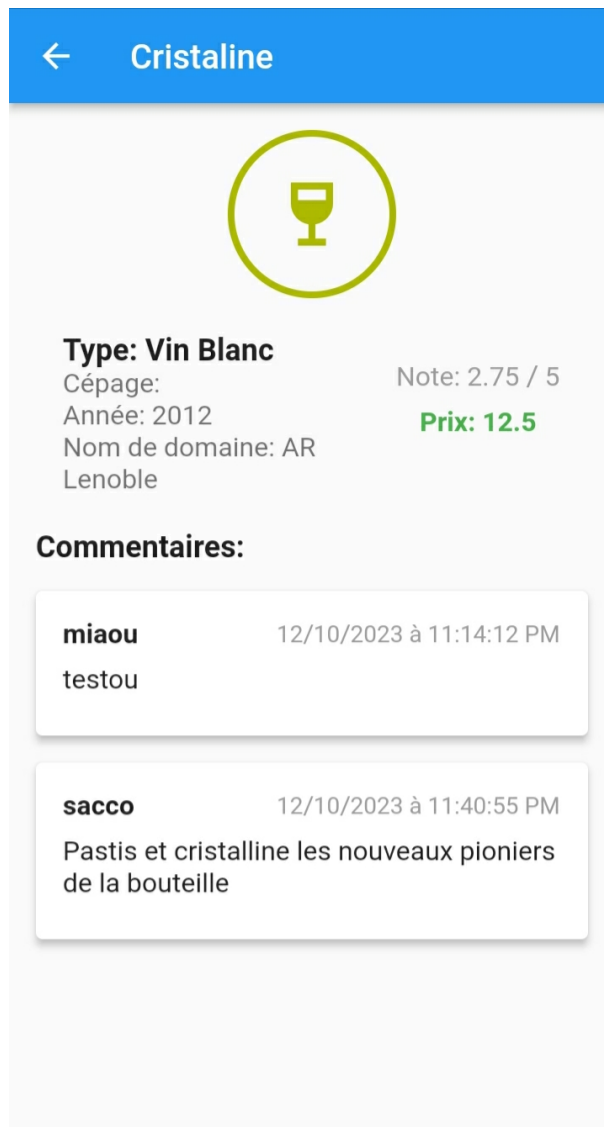


Figure 5. Écran DetailsVin

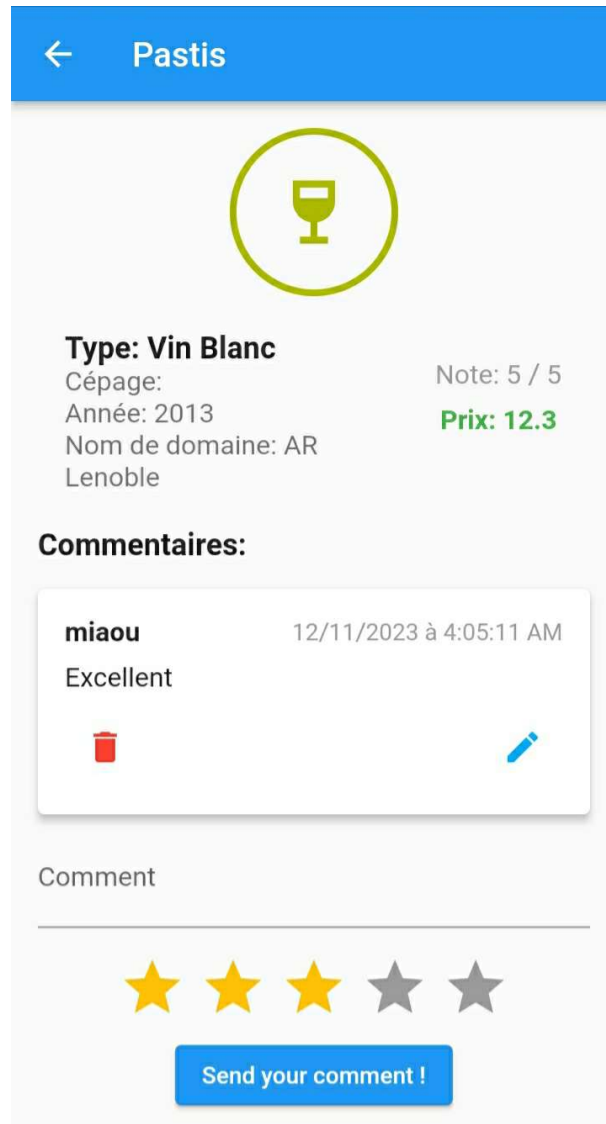


Figure 6. Écran DetailsVin Administrateur

Affiche des informations détaillées sur un vin particulier. Les utilisateurs peuvent ajouter des commentaires et les administrateurs peuvent supprimer des commentaires ou modifier les informations du vin.

2.2.4 Écran Scanner

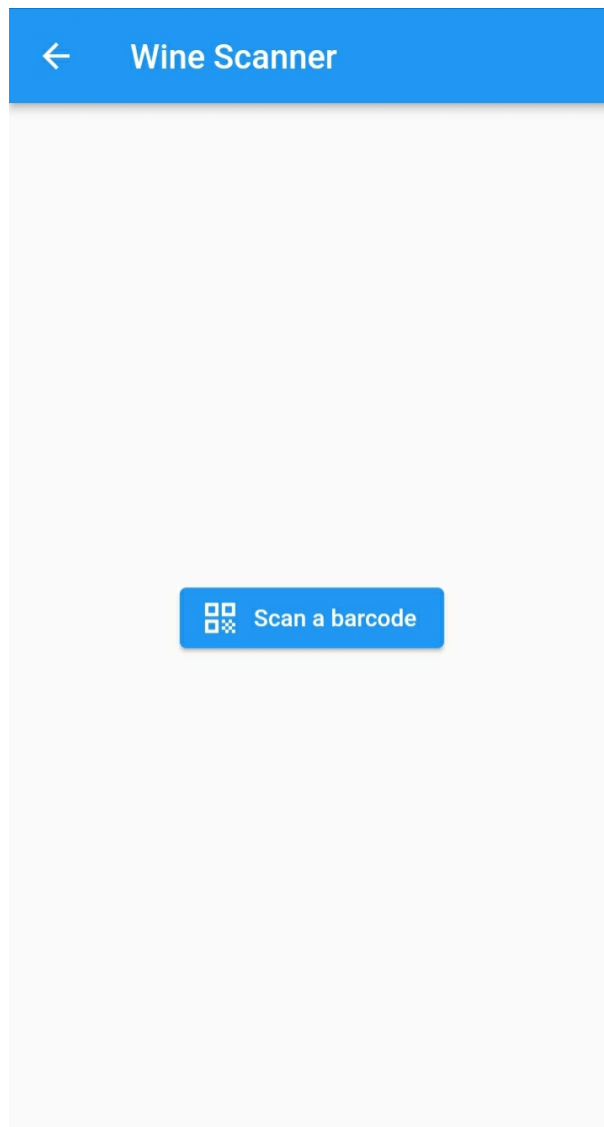
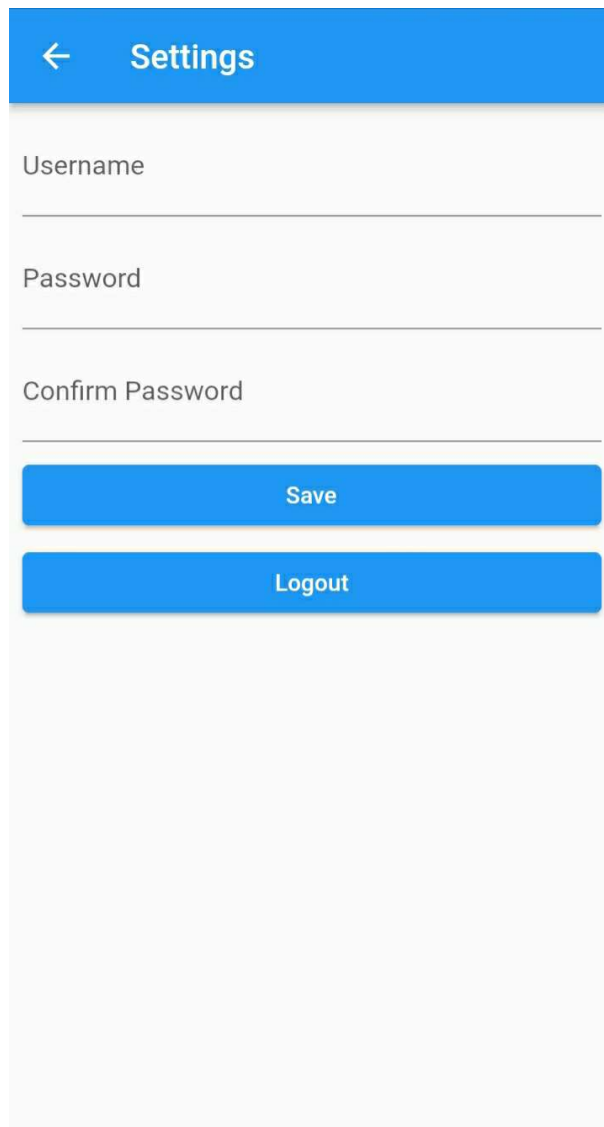


Figure 7. Écran Scanner

Cet écran contient un bouton qui active le scanner de code-barres. Si un vin est trouvé, vous accédez à l'écran *DetailsVin*. Sinon, un message d'erreur s'affiche.

2.2.5 Écran Settings

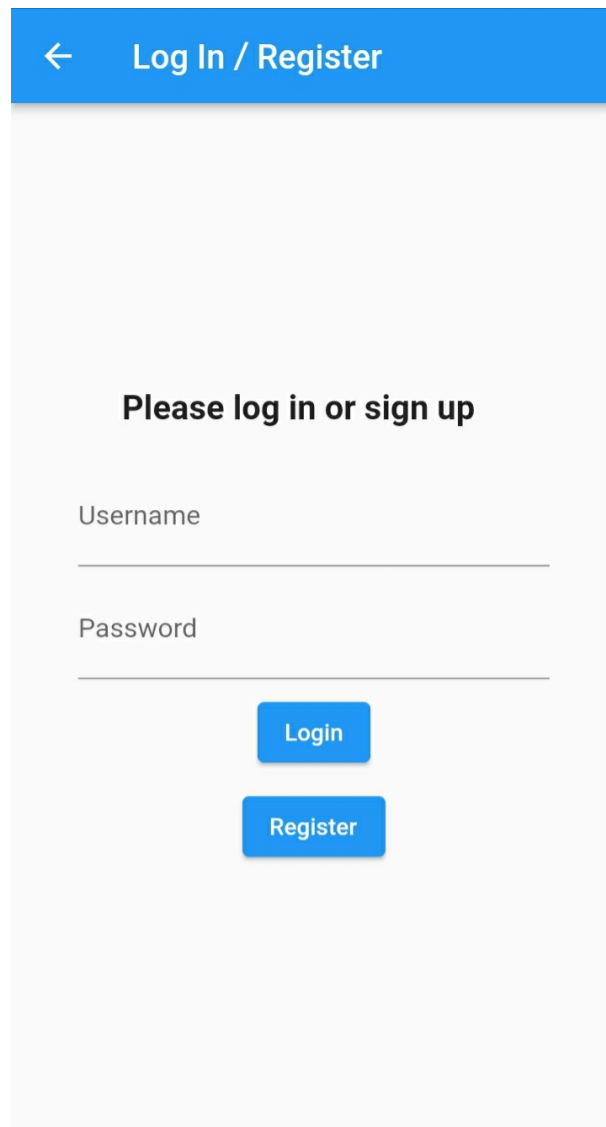


The screenshot shows a mobile application interface for the 'Settings' screen. At the top, there is a blue header bar with a white back arrow icon on the left and the word 'Settings' in white text. Below the header, the background is a light gray. There are three input fields, each with a label above it: 'Username', 'Password', and 'Confirm Password'. Each label is in a dark gray font. Below the input fields, there are two blue buttons with white text. The first button is labeled 'Save' and the second button is labeled 'Logout'. The buttons are stacked vertically.

Figure 8. Écran Settings

Permet de modifier le nom d'utilisateur et le mot de passe, ainsi que de se déconnecter avec le bouton *Logout*.

2.2.6 Écran Login/Register



The screenshot shows a mobile application interface for logging in or registering. At the top, there is a blue header bar with a white back arrow on the left and the text "Log In / Register" in white. Below the header, the background is a light gray. In the center, the text "Please log in or sign up" is displayed in bold black font. Underneath this text, there are two input fields: "Username" and "Password", each with a horizontal line for text entry. Below the "Password" field, there are two blue buttons with white text: "Login" and "Register", stacked vertically.

Figure 9. Écran Login/Register

Page permettant aux utilisateurs de se connecter à un compte existant ou de créer un nouveau compte.

3 Choix de Technologies

3.1 Environnement de Développement (IDE, Outils)

3.1.1 Langage de Programmation

L'application a été développée en utilisant le framework Flutter avec le langage de programmation Dart. Flutter offre une approche novatrice pour le développement d'applications mobiles en permettant une compilation native tout en maintenant un codebase unique pour les plateformes iOS et Android. Dart, en tant que langage de Flutter, offre une syntaxe concise et moderne, facilitant le développement rapide et la maintenance du code.

3.1.2 IDE (Environnement de Développement Intégré)

Visual Studio Code a été choisi comme IDE principal pour le développement. Il offre une intégration fluide avec Flutter, fournissant des fonctionnalités telles que la complétion automatique, le débogage avancé, et des extensions spécifiques à Flutter. La communauté active et les mises à jour fréquentes font de Visual Studio Code un choix robuste pour le développement Flutter.

3.2 Serveur Web

3.2.1 Justification du Choix de Node.js + Express.js

Node.js avec Express.js a été sélectionné comme serveur web en raison de sa vitesse et de sa scalabilité. Node.js, basé sur le moteur V8 de Google, excelle dans la gestion de requêtes concurrentes, ce qui est crucial pour une application destinée à avoir plusieurs milliers d'utilisateurs par jour. Express.js, en tant que framework minimaliste, offre une structure légère et permet une gestion efficace des routes et des middlewares.

3.2.2 Exécution Asynchrone et Événementielle

Node.js est construit sur un modèle asynchrone et événementiel. Cette approche permet de gérer de manière non bloquante les opérations d'entrées/sorties, ce qui facilite le traitement efficace d'un grand nombre de connexions simultanées sans bloquer le thread d'exécution. Comparé à des serveurs basés sur des threads, comme Java ou PHP avec Apache, Node.js excelle dans la gestion de multiples connexions grâce à son approche événementielle.

3.3 Base de Données

3.3.1 Justification du Choix de MongoDB

La base de données MongoDB a été choisie pour sa capacité de mise à l'échelle horizontale, sa flexibilité en termes de schéma, et son intégration naturelle avec Flutter/Dart via des plugins. MongoDB offre également des fonctionnalités telles que la réplication, le sharding, et une indexation efficace, répondant aux besoins d'une application avec un fort trafic et des données non structurées.

3.4 Frameworks et Bibliothèques

3.4.1 Justification de l'utilisation de Flutter Packages

Le choix d'utiliser des packages Flutter existants a accéléré le développement et amélioré la qualité du code. Par exemple, l'utilisation du package `simple_barcode_scanner` a simplifié l'intégration de la technologie de reconnaissance d'image, offrant une solution efficace pour scanner les codes-barres des bouteilles de vin.

D'autres bibliothèques comme http présentes nativement dans Flutter ont aussi grandement facilité et accéléré le développement, mais surtout, le lien entre les différents blocs applicatifs.

4 Architecture de l'Application

4.1 Matrice de Flux

Composant Source	Flux	Composant Cible
Client (Application Mobile)	Requête de Liste des Vins	Serveur Web
Serveur Web	Réponse Liste des Vins	Client (Application Mobile)
Client (Application Mobile)	Requête de Détails d'un Vin	Serveur Web
Serveur Web	Réponse Détails d'un Vin	Client (Application Mobile)
Client (Application Mobile)	Requête de Scanner de Code-barres	Serveur Web
Serveur Web	Réponse Scanner de Code-barres	Client (Application Mobile)
Client (Application Mobile)	Requête de Connexion/Inscription	Serveur Web
Serveur Web	Réponse Connexion/Inscription	Client (Application Mobile)
Client (Application Mobile)	Requête de Modification des Paramètres	Serveur Web
Serveur Web	Réponse Modification des Paramètres	Client (Application Mobile)

Table 1. Matrice des Flux de l'Application simplifiée

4.2 Architecture Technique

L'architecture globale de l'application "Shazam des Vins" est conçue de manière à répondre aux besoins de reconnaissance d'image, de gestion de base de données, et de communication entre le client (application mobile) et le serveur. Voici une description de l'architecture globale, en mettant l'accent sur les différents composants :

- **Client (Application Mobile - Flutter/Dart) :** L'application mobile développée avec Flutter/Dart sert d'interface utilisateur pour les utilisateurs finaux. Elle comprend différents écrans comme par exemple la liste des vins, les détails d'un vin, le scanner de code-barres, les paramètres utilisateur, et les fonctionnalités d'authentification. Les interactions utilisateur déclenchent des requêtes vers le serveur pour récupérer des données, effectuer des actions sur la base de données, ou réaliser des opérations de reconnaissance de code-barre.
- **Serveur Web (Node.js + Express.js) :** Le serveur web est construit avec Node.js et utilise le framework Express.js pour gérer les requêtes HTTP. Il reçoit les requêtes du client, effectue le routage approprié, et communique avec d'autres composants de l'infrastructure. Node.js permet une gestion efficace des opérations asynchrones, ce qui est crucial pour des applications avec une grande concurrence d'E/S, comme la

récupération de données depuis une base de données.

- **Base de Données (MongoDB) :** MongoDB est utilisé comme système de gestion de base de données (SGBD) pour stocker les informations sur les vins, les utilisateurs, les commentaires, etc. La base de données est conçue pour être évolutive grâce à l'utilisation de sharding, permettant de répartir les données sur plusieurs serveurs. Les opérations de lecture et d'écriture sont optimisées pour les requêtes fréquentes et la récupération rapide des données.
- **Système de Reconnaissance de Code-Barre :** Une technologie de reconnaissance de code-barre est intégrée pour permettre aux utilisateurs de scanner les codes-barres des bouteilles et d'obtenir des informations sur les vins.
- **Mécanismes d'Authentification et Gestion des Sessions :** Le système comprend des mécanismes d'authentification pour permettre aux utilisateurs de se connecter et de s'inscrire. Les sessions utilisateur sont gérées pour suivre l'état d'authentification des utilisateurs et fournir des fonctionnalités adaptées aux utilisateurs connectés.
- **Communication entre les Composants :** La communication entre le client et le serveur web se fait via des requêtes HTTP. Les interactions entre le serveur et la base de données se font uniquement grâce aux requêtes Mongo et le protocole TCP/IP.

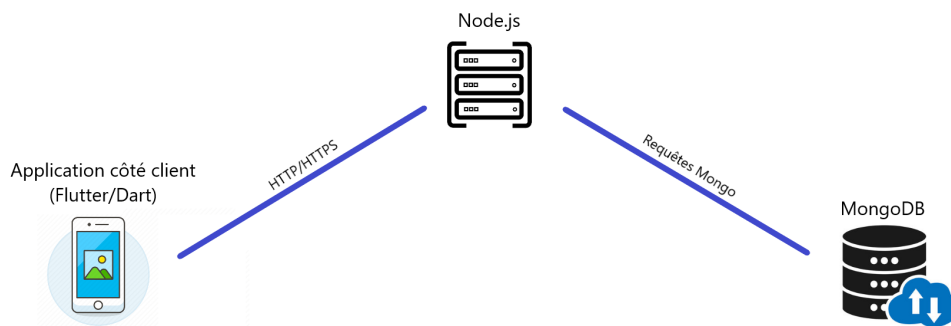


Figure 10. Diagramme d'Architecture Technique

4.3 Modélisation UML de l'Application côté Client

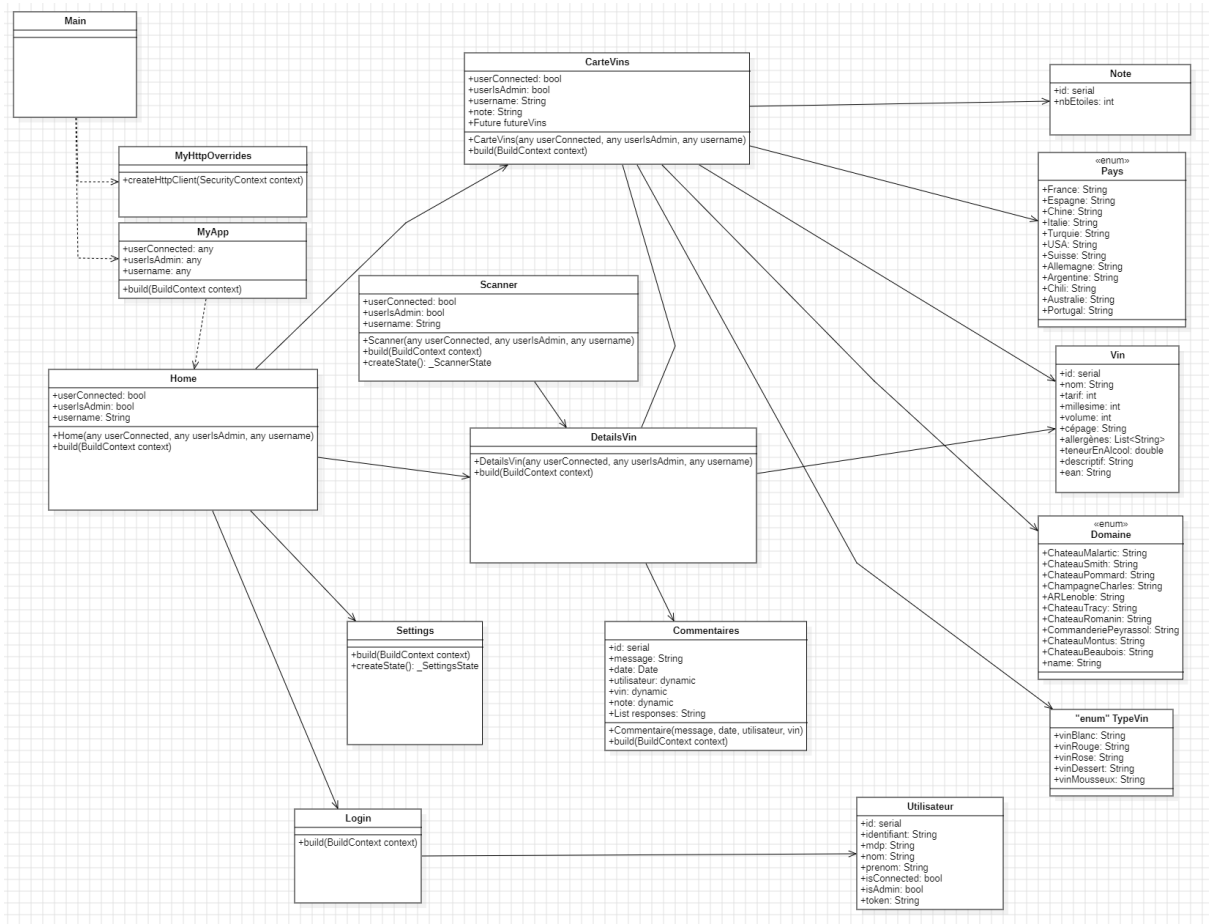


Figure 11. Modélisation

4.4 Modélisation UML de la Base de Données Mongo

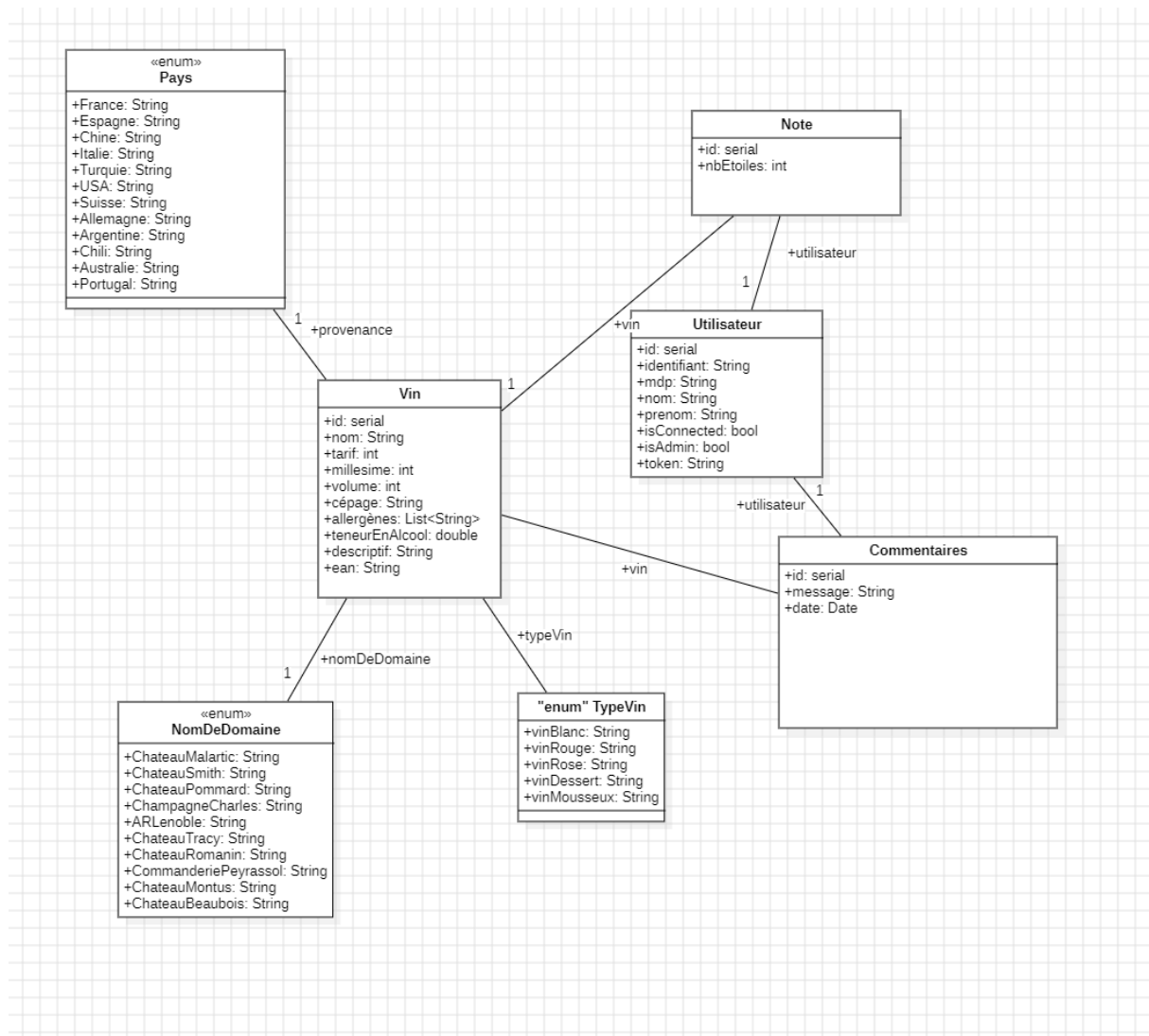


Figure 12. Modélisation

5 Autres Choix Stratégiques

5.1 Choix de Traitements et d'Organisation

5.1.1 Traitements côté serveur

Nous avons choisi de faire la plupart des traitements côté serveur pour profiter des avantages que nous offre **Node.js** en termes de performances.

5.1.2 Code-barres EAN-13

Etant donné que nous avons choisi d'utiliser le format de code-barres "**EAN-13**", nous avons aussi choisi de n'accepter que des codes EAN à 13 chiffres, respectant ainsi ce format. Il faut savoir que les codes EAN se retrouvent sur quasi tous les produits vendus aujourd'hui et permettent d'identifier des articles ou des unités logistiques de façon unique :

- **les codes EAN 8** sont réservés à l'usage sur des produits de petite taille (paquets de cigarettes par exemple)
- **les codes EAN 13** sont utilisés dans les domaines alimentaires et non-alimentaires
- **Les codes EAN 128** sont un code-barre long pour l'identification numérique de produits et informations alphanumériques.

5.1.3 Organisation de la base de données et ORM

Pour simplifier la gestion des requêtes, nous avons défini des ORM pour requêter des données en fonction de chaque type de données (Vin, Utilisateurs etc...). Nous avons donc une arborescence côté serveur adaptée avec un dossier par type de données et pour chaque dossier : un **Model**, un **Controller** et un **Service**.

Nous avons choisi de définir un **Controller** ainsi qu'un **Service** pour permettre de faire tous les traitements en lien avec notre base de données dans le **Service** et de renvoyer les informations appropriées (succès, échec etc..) depuis le controller.

C'est donc le **Controller** qui fait appel au **Service** et renvoie les informations concernant la requête.

Ainsi, nous découpons le travail en deux parties : le **Service** s'occupe de communiquer avec la base de données et le **Controller** de faire des traitements liés au données en entrée (par exemple, si un code EAN comporte bien 13 chiffres).

5.1.4 Sécurité et évolutivité

Nous avons aussi configuré notre serveur en **HTTPS** pour s'assurer que la communication entre le client et le serveur soit sécurisée.

Node.js présente des vulnérabilités en termes de sécurité. Les failles **XSS** sont des cas classiques, il faut pour cela encoder les données utilisateurs mais aussi définir des règles **CSP** pour restreindre le champ de possibilités d'intrusions.

Les failles SQL sont aussi fréquemment utilisées, il y a plusieurs façons de s'en protéger, nous avons fait le choix d'utiliser des requêtes préparées à partir de l'**ODM Mongoose**.

Cependant les ORM/ODM ne garantissent pas une sécurité totale, face aux vulnérabilités **CSRF** par exemple, qui peuvent permettre à un attaquant d'effectuer des opérations CRUD sur la base de données.

Pour cela, l'utilisation de tokens est nécessaire (grâce à **JWT** par exemple pour notre serveur **Node.js**). Aussi, il faut sécuriser le CORS, en définissant les possibles points d'entrée, puisque par défaut, la configuration du CORS autorise tous les points d'entrée ("**origin**").

Pour garantir de la sécurité dans notre application, notre serveur Node est configuré en **HTTPS** (avec notre propre certificat auto-signé) avec toutes les routes accessibles par la méthode **POST**.

5.2 Choix d'Interface Utilisateur (UX)

- **Bouton de déconnexion** : Nous avons choisi de mettre en place un bouton de déconnexion dans le menu Settings au lieu d'en avoir un sur toutes les pages.
- **Affichage de l'icone d'un vin** : Nous avons choisi de ne pas mettre d'image pour plusieurs raisons :
 - Pas de jugement de la part de l'utilisateur sur le packaging, mais uniquement sur les qualités gustatives
 - Un style plus moderne et plus en adéquation avec l'ambiance générale de notre application

Nous avons défini des constantes pour chaque type de vin et en fonction de la couleur du vin, nous affichons l'icone du vin avec la couleur correspondante.

5.3 Gestion des Erreurs et Retours Utilisateur

Comme précisé auparavant, nous avons fait la majorité des traitements côté serveur avec un principe simple. Tout d'abord, l'utilisateur effectue sa requête côté client, ensuite le serveur traite la requête et renvoie la réponse appropriée (par exemple, succès ou Nom de domaine manquant). Ensuite, en fonction de la réponse, nous affichons une **SnackBar** de couleur verte ou rouge indiquant le succès de la requête ou l'erreur renvoyée par le serveur. L'utilisateur est informé en temps réel et guidé dans l'utilisation de l'application. Ainsi nous n'avons pas à nous occuper des traitements des erreurs côté client, mais seulement d'afficher celles renvoyées par le serveur **Node**. Nous avons inclus des messages d'erreur détaillés dans la réponse du serveur. Cela guidera l'utilisateur pour comprendre d'où vient le problème lorsqu'il y a une erreur.

6 Montée en Charge

L'application "Shazam des Vins" pourrait connaître un fort succès dès son lancement, avec plusieurs milliers d'utilisateurs quotidiens. Pour anticiper la montée en charge, une analyse des prévisions de trafic a été effectuée. Les moments de pic d'utilisation, tels que les heures de pointe ou les lancements de campagnes promotionnelles, ont été pris en compte. Cette analyse permet de dimensionner l'infrastructure en conséquence pour garantir une expérience utilisateur fluide. Voici une liste de modifications/ajouts/fonctionnalités déjà existantes permettant de prévoir la montée en charge.

6.1 Optimisation du Code Flutter/Dart

Le code a été optimisé pour minimiser la consommation de ressources côté client et maximiser la réactivité de l'interface utilisateur, puisque la plupart des traitements et vérification de données sur les différentes requêtes se fait côté serveur.

6.2 Caching côté Client

Les données fréquemment consultées, telles que la liste des vins, pourrait-être mises en cache côté client pour réduire les temps de chargement et minimiser les requêtes redondantes vers le serveur.

Il est possible d'utiliser une base de données légère de type SQLite pour stocker des données en cache qui sont fréquemment consultées.

6.3 Compression de Données

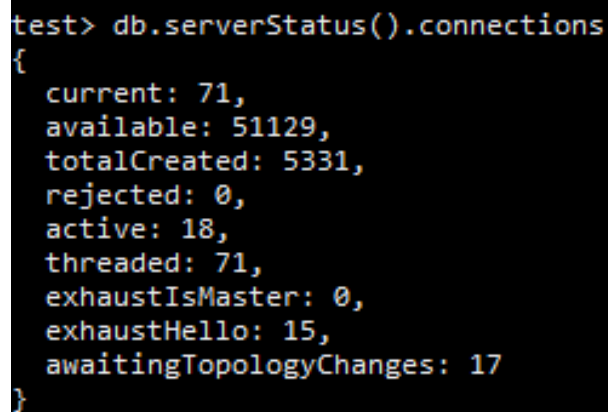
Les données échangées entre le client et le serveur pourraient être compressées pour réduire la bande passante utilisée, améliorant ainsi la rapidité des échanges.

Des compressions de données sont souvent utilisées pour réduire la charge des échanges client/serveur notamment grâce à gzip. Sachant que la quasi totalité des navigateurs acceptent la compression de données, nous pourrions très bien compresser une liste de Vin, les transmettre au client et laisser le client décompresser et mettre à jour l'affichage des vins.

6.4 Utilisation de la base Mongo du serveur pedago

Nous avons choisi d'utiliser la base de données MongoDB hébergée sur le serveur pedago en raison de sa limite de nombre de connexions.

La limite est de 51 200 connexions à la base de données MongoDB sur le serveur pédago.



```
test> db.serverStatus().connections
{
  current: 71,
  available: 51129,
  totalCreated: 5331,
  rejected: 0,
  active: 18,
  threaded: 71,
  exhaustIsMaster: 0,
  exhaustHello: 15,
  awaitingTopologyChanges: 17
}
```

Figure 13. Limite du nombre de connexions à la base Mongo

6.5 Utilisation de Redis

L'utilisation d'outils comme Redis pourrait-être envisagé en complément pour l'augmentation des performances de la base de données au plus haut niveau. Les requêtes fréquentes et les résultats peuvent être mis en cache dans Redis pour éviter des accès excessifs à la base de données principale.

- **Redis présente de nombreux avantages dont :**
 - Rapide au niveau des opérations en mémoire
 - NoSQL, HashMap de clé/valeur donc une structure simple
 - Scalable horizontalement grâce à des clusters Redis
- **Cependant, Redis a des désavantages :**
 - Les coûts de la RAM qui peut vite augmenter lorsqu'on a une grande quantité de données
 - Les clusters Redis sont coûteux en hardware et en maintenance
 - Pour la réplication, Redis a adopté le modèle maître-esclave. Mais il y a un maître pour plusieurs esclaves et quand le maître tombe en panne, c'est toute l'architecture qui tombe en panne à son tour

Redis permet de pouvoir faire jusqu'à des millions de requêtes par secondes. Cependant, nous sommes limités par la taille mémoire et Redis est très coûteux pour faire des opérations sur de grandes quantités de données.

6.6 Répartition de Charge

Un mécanisme de répartition de charge pourrait-être mis en place pour équilibrer la charge entre les différentes instances du serveur web. Cela garantirait une utilisation optimale des ressources disponibles.

La mise à l'échelle horizontale du serveur web est envisagée en ajoutant de nouvelles instances en cas de pic de charge. Ceci est particulièrement efficace pour gérer un grand nombre de connexions simultanées.

Node.js permet aussi de tirer parti de services cloud tels qu'AWS ou MCP. Ces services offrirait une mise à l'échelle automatique en fonction de la demande, assurant une évolutivité sans avoir à réorganiser manuellement l'infrastructure.

La base de données MongoDB est optimisée grâce à l'indexation, permettant des accès plus rapides aux données. De plus, le sharding pourrait-être mis en œuvre pour répartir la charge sur plusieurs instances de MongoDB (comme shards), assurant ainsi une évolutivité horizontale (ex : le 1er shard contient les users avec un id de 1 à 500, le 2nd shard contient les users de 501 à 1000...etc...).

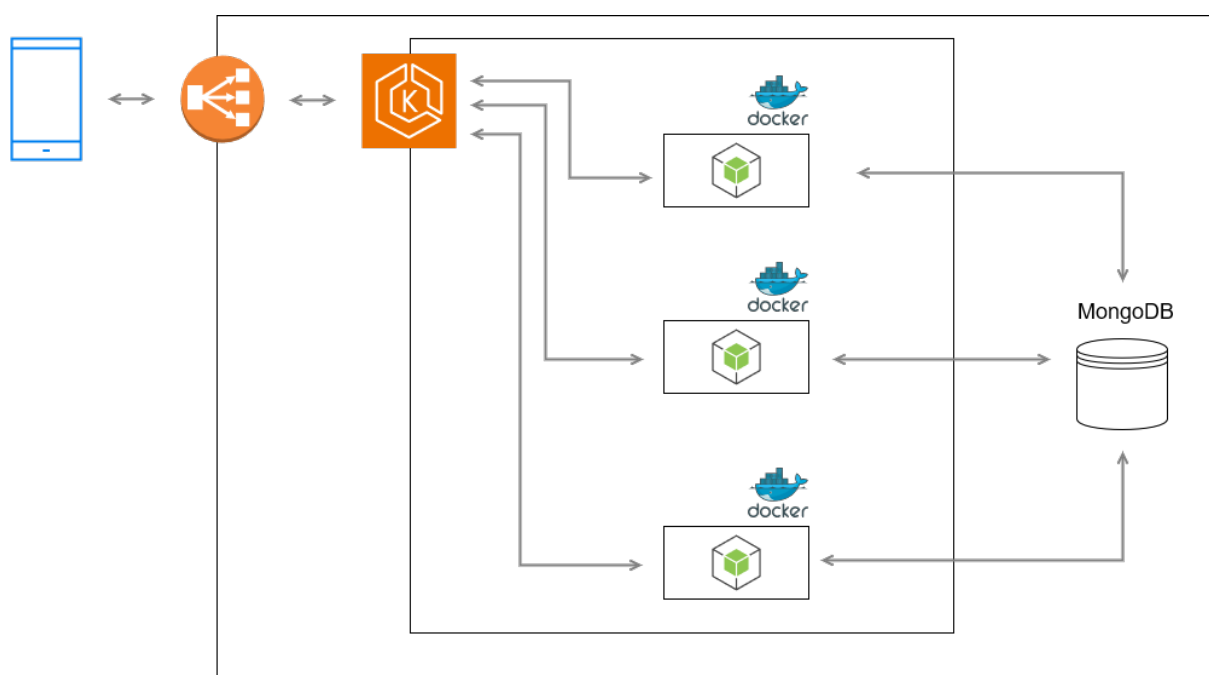


Figure 14. Architecture adaptée à la montée en charge

6.7 Tests de charge

Ce rapport présente une analyse détaillée des simulations de montée en charge effectuées sur le système ShazaMurge en utilisant l'outil Apache JMeter. L'objectif était d'évaluer les performances du système sous différentes charges d'utilisateurs et de déterminer un seuil critique au-delà duquel des optimisations sont nécessaires (ou la création d'une nouvelle instance).

Ici, nous avons (avec notre architecture actuelle), réalisé quelques tests de charge permettant de déterminer le trafic maximal pour une seule instance de Node, en tenant compte du fait qu'au delà de 1% de perte de requêtes, des améliorations sont nécessaires (ou la création d'une nouvelle instance).

```
PS C:\Program Files\apache-jmeter-5.6.2\bin> ./jmeter -n -t "C:\Users\Alexis\Documents\files\Cours\master\Semestre 7\UE Urbanisation\shazam\rapport-SACCOMAN-MEDOUAZ\tests montée en charge\Test Montée en Charge - ShazaMurge.jmx" -l "C:\Users\Alexis\Documents\files\Cours\master\Semestre 7\UE Urbanisation\shazam\rapport-SACCOMAN-MEDOUAZ\tests montée en charge\résultats\10000-2-5.jtl"
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
Creating summariser <summary>
Created the tree successfully using C:\Users\Alexis\Documents\files\Cours\master\Semestre 7\UE Urbanisation\shazam\rapport-SACCOMAN-MEDOUAZ\tests montée en charge\Test Montée en Charge - ShazaMurge.jmx
Starting standalone test @ 2023 Dec 12 18:57:39 CET (1702403859196)
Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump message on port 4445
summary + 517 in 00:00:21 = 24.9/s Avg: 18131 Min: 13790 Max: 20489 Err: 0 (0.00%) Active: 2083 Started: 2600 Finished: 517
summary + 2083 in 00:00:24 = 87.4/s Avg: 24960 Min: 18682 Max: 42156 Err: 204 (9.79%) Active: 0 Started: 2600 Finished: 2600
summary + 2600 in 00:00:45 = 58.2/s Avg: 23602 Min: 13790 Max: 42156 Err: 204 (7.85%) Active: 0 Started: 2600 Finished: 2600
tidying up ... @ 2023 Dec 12 18:58:23 CET (1702403903905)
... end of run
PS C:\Program Files\apache-jmeter-5.6.2\bin> ./jmeter -n -t "C:\Users\Alexis\Documents\files\Cours\master\Semestre 7\UE Urbanisation\shazam\rapport-SACCOMAN-MEDOUAZ\tests montée en charge\Test Montée en Charge - ShazaMurge.jmx" -l "C:\Users\Alexis\Documents\files\Cours\master\Semestre 7\UE Urbanisation\shazam\rapport-SACCOMAN-MEDOUAZ\tests montée en charge\résultats\10000-2-5.jtl"
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
Creating summariser <summary>
Created the tree successfully using C:\Users\Alexis\Documents\files\Cours\master\Semestre 7\UE Urbanisation\shazam\rapport-SACCOMAN-MEDOUAZ\tests montée en charge\Test Montée en Charge - ShazaMurge.jmx
Starting standalone test @ 2023 Dec 12 18:58:39 CET (1702403919786)
Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump message on port 4445
summary + 377 in 00:00:20 = 18.7/s Avg: 18462 Min: 15521 Max: 19686 Err: 0 (0.00%) Active: 2134 Started: 2510 Finished: 376
summary + 2133 in 00:00:23 = 92.1/s Avg: 22526 Min: 18171 Max: 41859 Err: 106 (4.97%) Active: 0 Started: 2510 Finished: 2510
summary + 2510 in 00:00:43 = 58.0/s Avg: 21916 Min: 15521 Max: 41859 Err: 106 (4.22%) Active: 0 Started: 2510 Finished: 2510
tidying up ... @ 2023 Dec 12 18:59:23 CET (1702403963157)
... end of run
PS C:\Program Files\apache-jmeter-5.6.2\bin> ./jmeter -n -t "C:\Users\Alexis\Documents\files\Cours\master\Semestre 7\UE Urbanisation\shazam\rapport-SACCOMAN-MEDOUAZ\tests montée en charge\Test Montée en Charge - ShazaMurge.jmx" -l "C:\Users\Alexis\Documents\files\Cours\master\Semestre 7\UE Urbanisation\shazam\rapport-SACCOMAN-MEDOUAZ\tests montée en charge\résultats\10000-2-5.jtl"
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
Creating summariser <summary>
Created the tree successfully using C:\Users\Alexis\Documents\files\Cours\master\Semestre 7\UE Urbanisation\shazam\rapport-SACCOMAN-MEDOUAZ\tests montée en charge\Test Montée en Charge - ShazaMurge.jmx
Starting standalone test @ 2023 Dec 12 18:59:42 CET (1702403982271)
Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump message on port 4445
summary + 1335 in 00:00:18 = 75.6/s Avg: 14688 Min: 11048 Max: 16568 Err: 0 (0.00%) Active: 1168 Started: 2505 Finished: 1337
summary + 1170 in 00:00:06 = 210.1/s Avg: 17515 Min: 15479 Max: 22139 Err: 0 (0.00%) Active: 0 Started: 2505 Finished: 2505
summary + 2505 in 00:00:23 = 107.8/s Avg: 16088 Min: 11048 Max: 22139 Err: 0 (0.00%) Active: 0 Started: 2505 Finished: 2505
tidying up ... @ 2023 Dec 12 19:00:05 CET (1702404005572)
... end of run
PS C:\Program Files\apache-jmeter-5.6.2\bin> ./jmeter -n -t "C:\Users\Alexis\Documents\files\Cours\master\Semestre 7\UE Urbanisation\shazam\rapport-SACCOMAN-MEDOUAZ\tests montée en charge\Test Montée en Charge - ShazaMurge.jmx" -l "C:\Users\Alexis\Documents\files\Cours\master\Semestre 7\UE Urbanisation\shazam\rapport-SACCOMAN-MEDOUAZ\tests montée en charge\résultats\10000-2-5.jtl"
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and will be removed in a future release
Creating summariser <summary>
Created the tree successfully using C:\Users\Alexis\Documents\files\Cours\master\Semestre 7\UE Urbanisation\shazam\rapport-SACCOMAN-MEDOUAZ\tests montée en charge\Test Montée en Charge - ShazaMurge.jmx
Starting standalone test @ 2023 Dec 12 19:01:18 CET (1702404078868)
Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump message on port 4445
summary + 0 in 00:00:13 = 0.6/s Avg: 10235 Min: 9148 Max: 12365 Err: 0 (0.00%) Active: 2493 Started: 2507 Finished: 14
summary + 2497 in 00:00:32 = 77.2/s Avg: 20353 Min: 11855 Max: 42420 Err: 44 (1.76%) Active: 3 Started: 2507 Finished: 2504
summary + 2505 in 00:00:45 = 55.8/s Avg: 20321 Min: 9148 Max: 42420 Err: 44 (1.76%) Active: 0 Started: 2507 Finished: 2507
summary + 2 in 00:00:00 = 40.8/s Avg: 42815 Min: 42707 Max: 42924 Err: 0 (0.00%) Active: 0 Started: 2507 Finished: 2507
summary + 2507 in 00:00:45 = 55.8/s Avg: 20339 Min: 9148 Max: 42924 Err: 44 (1.76%) Active: 0 Started: 2507 Finished: 2507
tidying up ... @ 2023 Dec 12 19:02:03 CET (1702404123854)
... end of run
PS C:\Program Files\apache-jmeter-5.6.2\bin>
```

Figure 15. Tests de charge pour plusieurs milliers d'utilisateurs sur 2 secondes sur une requête https permettant d'afficher la carte des vins

Ici on peut donc voir que le seuil "critique", est de 2506 utilisateurs.

7 Conclusion

7.1 Récapitulation des Points Clés

Pour résumer, l'application offre la possibilité de consulter, modifier et supprimer des vins ainsi que des notes et des commentaires.

Une technologie de reconnaissance d'images permet de scanner le code-barres d'un vin pour afficher ses informations.

L'application a été développée en utilisant Flutter, un serveur web Node.js ainsi qu'une base de données MongoDB tous deux hébergés sur le serveur pedago.

L'interface comprend plusieurs pages tel que l'écran d'accueil, la carte des vins, le détails des vins, les settings et la page de login/register.

Nous avons prévu pour le scaling l'utilisation des avantages que nous offre MongoDB ainsi que Node.js mais aussi les possibilités d'évolution avec potentiellement l'utilisation de serveurs basés sur le Cloud tel que les services Amazon AWS ou Google Cloud.

7.2 Perspectives Futures

Dans le cadre d'une démarche de préservation de la biodiversité et du mouvement de "vins dynamique", un nouveau QR code est obligatoire sur les bouteilles de vin depuis le 8 décembre 2023. Suite à une nouvelle réglementation, les informations concernant la composition d'un vin doivent être renseignée par les producteurs de vin. La liste des produits est maintenant accessible en scannant le QR code associé à une bouteille de vin.

Suite à cette réglementation, nous pourrons ajouter une nouvelle fonctionnalité dans notre application permettant de prendre en charge le scan de QR code (en plus du code-barres), pour nous permettre d'afficher des informations supplémentaires sur le vin, en l'occurrence sa composition ([ce qui va changer en 2023](#)).