

# Rapport Vintellect

Groupe ILSEN classique

**Ange Curé**  
**Aubertin Emmanuel**

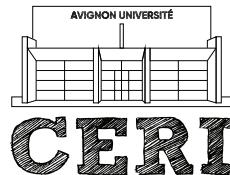
**12 décembre 2023**

**Master informatique**  
**ILSEN**

**UE** Nom de l'unité d'enseignement  
**ECUE** Urbanisation

**Responsables**  
Mickeal Rouvier  
Yanis Labrak

**UFR**  
**SCIENCES**  
**TECHNOLOGIES**  
**SANTÉ**



CENTRE  
D'ENSEIGNEMENT  
ET DE RECHERCHE  
EN INFORMATIQUE  
[ceri.univ-avignon.fr](http://ceri.univ-avignon.fr)

## Sommaire

<b>Titre</b>	<b>1</b>
<b>Sommaire</b>	<b>2</b>
<b>1 Introduction</b>	<b>4</b>
1.1 Objectifs du Projet . . . . .	4
1.2 Contraintes Techniques et de Conception . . . . .	4
1.2.1 Sécurité . . . . .	4
1.2.2 Les Coûts . . . . .	5
<b>2 Scénario d'Utilisation</b>	<b>6</b>
2.1 Phase de Conception . . . . .	6
2.2 Catégories d'Utilisateurs et Fonctionnalités . . . . .	6
2.3 Analyse des Cas d'Utilisation . . . . .	6
2.3.1 Interactions des Utilisateurs Standards . . . . .	6
2.3.2 Interactions des Administrateurs . . . . .	6
<b>3 Choix d'Architecture</b>	<b>7</b>
3.1 Cloud Public, Cloud Privé et Legacy Systems . . . . .	7
3.1.1 Définitions . . . . .	7
3.1.2 Avantages du Cloud Comparé aux Systèmes Legacy . . . . .	8
3.2 Justification du Choix de Google Cloud Platform (GCP) . . . . .	8
3.2.1 Expansion en Part de Marché . . . . .	8
3.2.2 Disponibilité Géographique Accrue . . . . .	8
3.2.3 Monitoring et Services Managés . . . . .	8
3.2.4 Potentiel de Mise en Œuvre de DevOps et GitOps sur GCP . . . . .	9
3.2.5 Simplicité d'Utilisation . . . . .	10
3.2.6 Intérêt Pédagogique des Choix Technologiques . . . . .	10
<b>4 Architecture Technique</b>	<b>11</b>
4.1 Microservices vs Monolith . . . . .	11
4.1.1 Définition des Microservices . . . . .	11
4.1.2 Définition de l'Architecture Monolithique . . . . .	11
4.2 Adopter les Microservices dans une Architecture Cloud . . . . .	11
4.3 Déploiement des Microservices . . . . .	12
App Engine . . . . .	12
Cloud Run . . . . .	12
Google Kubernetes Engine (GKE) . . . . .	12
4.3.1 Justification du Choix d'App Engine . . . . .	12
4.4 Stockage . . . . .	12
4.4.1 Évaluation des Options de Stockage . . . . .	12
4.4.2 Choix de Cloud Spanner . . . . .	13
4.4.3 Utilisation d'un Bucket pour les Images . . . . .	13
4.5 Gestion des Utilisateurs . . . . .	13
4.5.1 Sécurisation des Comptes . . . . .	13
4.5.2 Simplification de la Gestion des Utilisateurs . . . . .	14
4.5.3 Coût et Efficacité . . . . .	14
4.6 Google Vision . . . . .	14

<b>5 Choix Technologique pour le Front-End</b>	<b>14</b>
Critères de Sélection . . . . .	14
Evaluation des Technologies . . . . .	14
5.1 Justification du Choix de React Native . . . . .	15
Rapidité de Développement . . . . .	15
Performance . . . . .	15
Communauté et Support . . . . .	15
Flexibilité . . . . .	15
Compatibilité et Intégration . . . . .	15
<b>6 Implémentations</b>	<b>16</b>
6.1 Fonctionnement du Front-End . . . . .	16
6.1.1 Interface Unifiée pour Administrateurs et Utilisateurs . . . . .	16
6.1.2 Documentation sur l'Utilisation . . . . .	17
6.1.3 Accès aux Sources . . . . .	17
6.2 Microservice de Gestion des Utilisateurs . . . . .	17
6.2.1 Fonctionnement du Microservice . . . . .	17
6.2.2 Sécurité et Accès Restreint . . . . .	17
6.3 Microservice de Feedback . . . . .	18
6.3.1 Fonctionnalités du Microservice . . . . .	18
6.3.2 Gestion des Contenus et Sécurité . . . . .	18
6.4 Microservice Wine . . . . .	18
6.4.1 Fonctionnalités du Microservice . . . . .	18
6.5 Administration des Vins . . . . .	19
6.5.1 Fonctionnalités du Microservice . . . . .	19
6.5.2 Sécurité et Intégrité des Données . . . . .	19
<b>7 Fonctionnement de l'application</b>	<b>19</b>
7.1 Page de connexion . . . . .	19
7.2 Page scan d'un vin . . . . .	20
7.3 Page de Liste des Vins . . . . .	21
7.4 Page détaillé d'un vin . . . . .	22
7.5 Page profile . . . . .	23
7.6 Administration . . . . .	24
<b>8 Conclusion</b>	<b>25</b>
8.1 Conclusion d'Ange . . . . .	25
8.2 Conclusion d'Emmanuel . . . . .	26

## 1 Introduction

Dans le cadre académique de l'unité d'enseignement "Urbanisation", faisant partie du programme de premier cycle de Master au Centre d'Études et de Recherches Internationales (CERI), nous avons été chargés de la conception et du développement d'une application mobile. Cette dernière se concentre sur la reconnaissance de bouteilles de vin et donne diverses informations sur le vin en question.

Le code source et les explications pour le déploiement sont disponibles sur ce [lien Github](#).

### 1.1 Objectifs du Projet

#### Fonctionnalités Principales

L'ambition principale de ce projet réside dans la création d'un outil capable d'extraire automatiquement et avec précision une multitude d'informations pertinentes à partir de photographies de bouteilles de vin. Les données extraites par l'application incluront non seulement des détails élémentaires tels que la description du vin, l'année de la cuvée, et la variété de cépage, mais aussi des éléments plus subjectifs et nuancés tels que les avis et évaluations formulés par les consommateurs antérieurs.

#### Interaction Utilisateur

Pour enrichir l'expérience utilisateur et favoriser un engagement actif, l'application sera dotée de fonctionnalités interactives sophistiquées. Ces dernières permettront aux utilisateurs non seulement de consulter les informations sur les vins, mais également de contribuer à la communauté en partageant leurs propres expériences et évaluations des produits dégustés.

#### Administration et Modération

Afin de maintenir un environnement sain et respectueux au sein de l'application, une interface administrateur sera développée. Cette interface permettra aux gestionnaires de l'application d'exercer un contrôle rigoureux sur le contenu publié par les utilisateurs. Ils auront ainsi la possibilité de modérer, éditer ou supprimer tout contenu jugé inapproprié ou ne respectant pas les normes et directives préétablies.

### 1.2 Contraintes Techniques et de Conception

#### Compatibilité Multiplateforme

Un défi technique sera de garantir que l'application soit pleinement fonctionnelle et optimisée pour les systèmes d'exploitation iOS et Android. Cela implique le développement parallèle de deux versions distinctes de l'application, chacune étant adaptée aux spécificités et aux exigences techniques de la plateforme concernée.

#### Haute Disponibilité

L'accessibilité constante de l'application est un aspect non négligeable pour assurer une expérience utilisateur de qualité et la fiabilité du service. Pour atteindre cet objectif, il sera nécessaire d'implémenter des solutions telles que des serveurs redondants, des procédures régulières de sauvegarde des données et un système de surveillance en continu pour minimiser les temps d'arrêt et garantir une performance optimale de l'application.

#### 1.2.1 Sécurité

La sécurisation des données utilisateur est un enjeu primordial dans le développement de notre projet. Une architecture sécurisée assure la protection des données, la résilience face aux attaques externes et la conformité aux normes et réglementations en vigueur. Dans cette optique, plusieurs stratégies ont été adoptées :

- **Authentification et Autorisation :** Mise en place de mécanismes robustes d'authentification et de gestion des droits d'accès pour contrôler l'accès aux ressources et services de l'application.
- **Chiffrement des Données :** Utilisation de protocoles de chiffrement avancés pour sécuriser les données en transit et au repos, afin de prévenir les fuites et l'accès non autorisé.
- **Surveillance et Détection des Menaces :** Intégration de systèmes de détection des intrusions et de surveillance continue pour identifier et répondre rapidement aux activités suspectes ou malveillantes.
- **Mises à jour et Patchs de Sécurité :** Application régulière de mises à jour et de patchs pour les composants logiciels, afin de corriger les vulnérabilités et renforcer la sécurité du système.
- **Veille sur les Différents Composants de l'Application :** Pour assurer une sécurité optimale, une veille constante est maintenue sur tous les composants de l'application. Cette démarche est essentielle pour maintenir la confiance des utilisateurs et garantir la conformité avec les standards de sécurité les plus élevés.

Ces mesures, intégrées dans le cycle de développement de l'application, garantissent un niveau de sécurité adapté aux enjeux et aux exigences de notre projet.

## 1.2.2 Les Coûts

La maîtrise des coûts est un aspect crucial, en particulier pour une application dont le nombre d'utilisateurs peut varier. Il est essentiel d'optimiser les ressources pour aligner les dépenses avec les revenus générés par le trafic utilisateurs. Pour cela, plusieurs approches ont été adoptées :

- **Scalabilité Dynamique :** Implémentation d'une infrastructure capable de s'adapter dynamiquement à la charge, augmentant ou réduisant les ressources en fonction de la demande réelle.
- **Optimisation des Ressources :** Analyse et ajustement continu des ressources allouées pour s'assurer qu'elles correspondent précisément aux besoins de l'application, évitant ainsi le surdimensionnement.
- **Choix de Technologies Économiques :** Sélection de technologies et de services offrant un bon rapport qualité-prix, et exploration des options de tarification flexibles proposées par les fournisseurs cloud.
- **Surveillance des Coûts :** Mise en place d'outils de surveillance et de reporting des coûts pour identifier et corriger rapidement les inefficacités.

En adoptant ces stratégies, nous visons à maintenir les coûts à un niveau optimal tout en garantissant la performance et la fiabilité de l'application.

**En résumé**, la conception et le développement de cette application de reconnaissance de bouteilles de vin doivent s'articuler autour d'une intégration méticuleuse et rigoureuse de contraintes diverses, allant des spécificités techniques liées aux différentes plateformes à des impératifs de haute disponibilité, sans oublier les exigences réglementaires. Cette démarche est essentielle pour assurer non seulement la viabilité technique et fonctionnelle du projet, mais aussi sa conformité avec les normes et standards actuels.

## 2 Scénario d'Utilisation

### 2.1 Phase de Conception

La phase initiale du projet, comme détaillée dans le document de projet, a nécessité une compréhension approfondie des exigences techniques et fonctionnelles. Cette étape a été cruciale pour l'élaboration de l'architecture de l'application et la sélection des technologies à utiliser. Une séance de brainstorming a permis de conceptualiser l'interface utilisateur (front-end) de l'application, en tenant compte des spécificités de l'architecture et des choix technologiques envisagés.

### 2.2 Catégories d'Utilisateurs et Fonctionnalités

Selon les spécifications du projet, l'application cible deux catégories principales d'utilisateurs : les administrateurs et les utilisateurs standards, avec des fonctionnalités distinctes pour chacun.

#### L'Utilisateur Standard

L'utilisateur standard, lorsqu'il se trouve dans une cave coopérative, peut utiliser l'application pour obtenir des informations détaillées sur une bouteille de vin en photographiant son étiquette. Les informations accessibles incluent la cuvée, les avis, les notes, le nom du vin, le descriptif, le nom du cépage, le château, et le prix. Après inscription, l'utilisateur peut également laisser un avis ou une note sur le vin.

#### L'Administrateur

L'administrateur joue un rôle important dans la gestion du contenu de l'application. Il peut accéder à l'application avec ses identifiants administrateur pour supprimer des commentaires offensants et corriger les informations sur les vins, y compris la description, la cuvée, et le nom. Cette gestion comprend l'ajout, la suppression, et la modification des descriptifs des vins, ainsi que des notes et commentaires des utilisateurs.

### 2.3 Analyse des Cas d'Utilisation

Le diagramme de cas d'utilisation présenté en Figure [1] illustre les interactions entre les utilisateurs de l'application Vintellect et ses fonctionnalités système. Ce diagramme sert de cadre pour modéliser les exigences fonctionnelles de l'application en termes de capacités offertes aux différents acteurs.

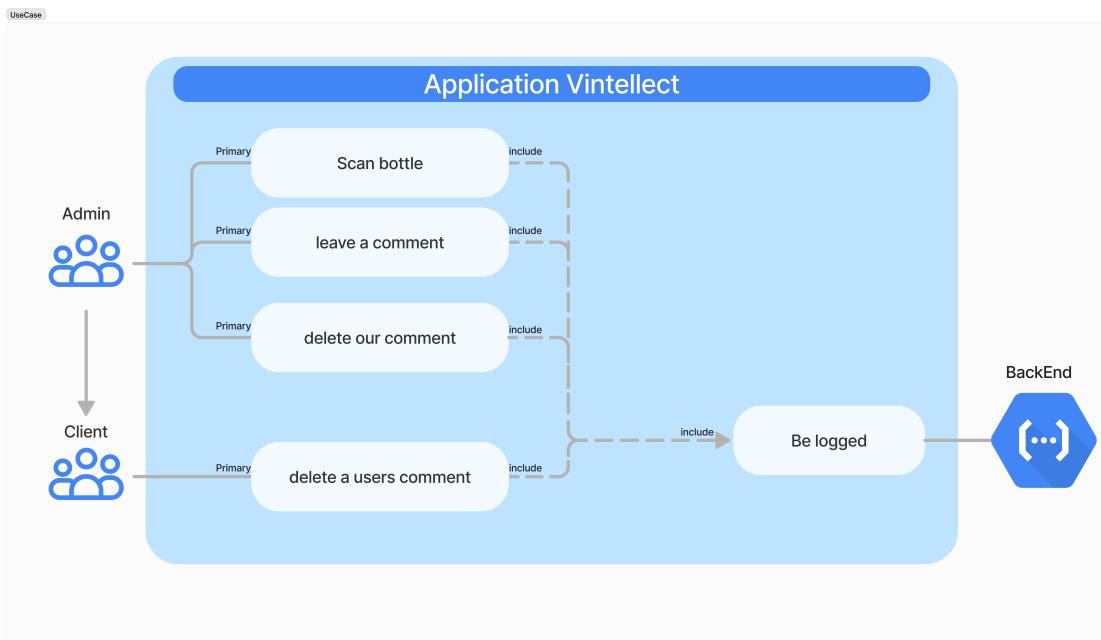
#### 2.3.1 Interactions des Utilisateurs Standards

Les utilisateurs standards, dénotés comme 'Client' dans le diagramme, peuvent effectuer les actions suivantes :

- **Scan de Bouteille :** En utilisant la fonctionnalité de scan, un client peut capturer l'image d'une bouteille de vin. L'application, grâce à son système de reconnaissance d'image intégré au BackEnd, traite cette image pour retrouver et afficher les informations pertinentes depuis la base de données.
- **Publication de Commentaire :** Un utilisateur peut également exprimer son avis sur un vin spécifique. Cette action est conditionnée par la vérification de l'authentification de l'utilisateur, nécessitant qu'il soit connecté au système. Si l'utilisateur n'est pas préalablement authentifié, le système l'invite à se connecter ou à créer un compte.

#### 2.3.2 Interactions des Administrateurs

L'administrateur, identifié distinctement dans le diagramme, dispose de priviléges étendus, notamment :

**Figure 1.** Diagramme UseCases

- **Gestion de Contenu :** Les administrateurs ont la capacité de modérer le contenu de l'application, ce qui inclut l'ajout, ou la suppression de commentaires et d'informations sur les vins. Ces actions nécessitent une authentification préalable de l'administrateur et une vérification que le compte utilisé possède les droits administratifs requis.

Chaque interaction utilisateur avec le système est médiée par le BackEnd, qui assure l'authentification (dénommée 'Be logged' dans le diagramme) et la gestion des données. Il est à noter que toutes les actions primaires des utilisateurs et des administrateurs sont incluses dans les fonctionnalités de base du BackEnd, soulignant l'importance de la robustesse et de la sécurité dans la conception de l'architecture système.

### 3 Choix d'Architecture

Dans cette section, nous procéderons à la justification et à l'explication détaillée des décisions relatives à notre choix d'architecture.

#### 3.1 Cloud Public, Cloud Privé et Legacy Systems

Dans cette section, nous examinerons les paradigmes d'architecture informatique, en particulier en comparant les infrastructures de cloud public et privé aux systèmes dits "legacy".

##### 3.1.1 Définitions

**Cloud Public** Le cloud public désigne des services de calcul délivrés par des fournisseurs de services sur Internet, offrant une élasticité et une échelle pouvant s'adapter aux besoins changeants des entreprises. Ces services sont partagés entre plusieurs organisations et accessibles via le réseau public.

**Cloud Privé** Le cloud privé fait référence à l'utilisation de services et d'infrastructures de cloud dédiés à une seule organisation. Cela permet un contrôle accru et une personnalisation de l'environnement informatique répondant aux exigences spécifiques de l'entité.

**Legacy Systems** Les systèmes "legacy" désignent les anciennes technologies et systèmes informatiques qui continuent d'être utilisés, malgré l'existence de solutions plus récentes et plus efficaces. Ils sont souvent synonymes d'une infrastructure sur place et peuvent être fortement personnalisés pour répondre aux besoins spécifiques d'une organisation.

### 3.1.2 Avantages du Cloud Comparé aux Systèmes Legacy

**Coûts** Le cloud offre une structure de coût variable basée sur l'utilisation, permettant aux organisations de payer uniquement pour les ressources consommées, ce qui peut se traduire par des économies substantielles par rapport à l'investissement initial et à la maintenance continue des systèmes legacy. Et cela permet un alignement entre le coût des ressources et le nombre d'utilisateurs servis.

**Fiabilité** Les fournisseurs de cloud public assurent une haute disponibilité grâce à des infrastructures redondantes et des pratiques de reprise après sinistre robustes, surpassant souvent la fiabilité des systèmes legacy qui peuvent être vulnérables aux défaillances dues à l'âge et à l'obsolescence des composants.

**Services Managés** Les cloud providers offrent une gamme de services managés qui réduisent la charge de gestion des systèmes d'information. Ces services incluent, mais ne se limitent pas à, la gestion des bases de données, l'analytique, et l'intelligence artificielle, permettant aux organisations de se concentrer sur leurs compétences clés plutôt que sur la gestion de l'infrastructure IT.

En conclusion, le choix entre les solutions de cloud public, cloud privé et les systèmes legacy doit être guidé par une analyse détaillée des besoins, des contraintes budgétaires et des objectifs stratégiques de l'organisation. Le cloud public et privé offre une flexibilité, une évolutivité et une efficacité qui peuvent être essentielles pour les entreprises dynamiques et en croissance.

## 3.2 Justification du Choix de Google Cloud Platform (GCP)

La sélection de Google Cloud Platform (GCP) comme fournisseur de services cloud pour notre projet s'appuie sur plusieurs facteurs stratégiques et techniques.

### 3.2.1 Expansion en Part de Marché

GCP a démontré une croissance significative de sa part de marché dans les services d'infrastructure cloud, comme le montre le graphique du rapport du Synergy Research Group. Cette tendance positive reflète la confiance croissante des entreprises dans GCP et indique un investissement soutenu dans l'amélioration de ses services.

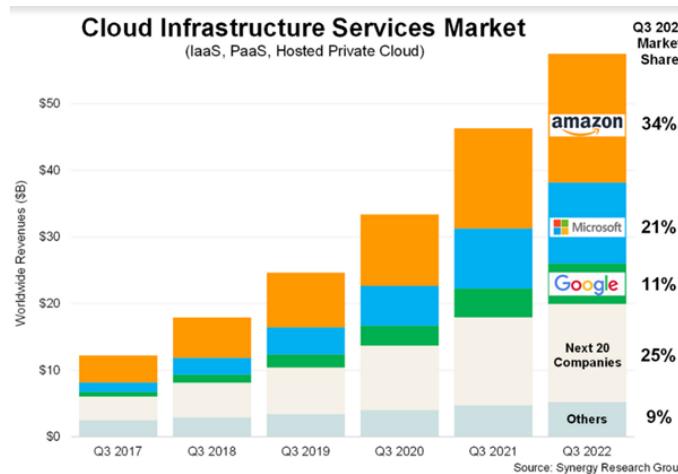
Â

### 3.2.2 Disponibilité Géographique Accrue

L'analyse de la carte de l'infrastructure de GCP révèle une expansion géographique étendue, avec une augmentation notable du nombre de régions et de zones. Cette présence mondiale assure une meilleure latence et une redondance accrue, garantissant une haute disponibilité des services pour les utilisateurs à travers le globe.

### 3.2.3 Monitoring et Services Managés

GCP offre une plateforme de monitoring intégrée et facile à utiliser, qui simplifie la surveillance des ressources et des services cloud. La simplicité de cette fonctionnalité, couplée aux services managés proposés par GCP, permet une gestion optimisée des infrastructures, réduisant ainsi la charge opérationnelle sur nos équipes de développement et d'exploitation.



**Figure 2.** Répartition du marché du Cloud Public

## GCP Infrastructure

6 regions, 18 zones, over 100 points of presence, and a well-provisioned global network comprised of hundreds of thousands of miles of fiber optic cable.



**Figure 3.** GCP dans le monde

### 3.2.4 Potentiel de Mise en Œuvre de DevOps et GitOps sur GCP

Envisageant le cadre d'une organisation, l'adoption de DevOps et GitOps constituerait une évolution stratégique significative, particulièrement dans l'environnement de Google Cloud Platform (GCP). L'infrastructure et les services proposés par GCP sont conçus pour soutenir et renforcer l'efficacité des pratiques DevOps et GitOps.

**Avantages de DevOps et GitOps sur GCP** L'intégration de DevOps au sein de GCP promet de transformer et d'optimiser nos pipelines de déploiement grâce à l'automatisation, réduisant ainsi les délais de développement et favorisant une mise sur le marché plus rapide. De manière complémentaire, GitOps pourrait bénéficier substantiellement de la synergie avec GCP, en particulier en utilisant des services comme Cloud Source Repositories pour une gestion et une révision du code source plus efficaces et cohérentes. En effet, comme le montre l'infographie donnée par McKinsey & Company, pour une valorisation optimale de l'entreprise, il faut sortir des nouveautés à hauteur d'une à quatre par mois.

**Considérations Opérationnelles** La migration vers DevOps et GitOps implique une intégra-

tion méticuleuse avec l'écosystème GCP, nécessitant une analyse approfondie des processus actuels. Cette démarche requiert une formation ciblée des équipes, l'établissement de nouvelles suites d'outils, et le développement de protocoles de surveillance et d'alerte pour assurer une opérationnalité ininterrompue et stable.

En somme, la mise en œuvre de DevOps et GitOps dans le contexte de GCP pourrait significativement améliorer l'agilité opérationnelle et la capacité d'innovation, tout en maintenant les standards de fiabilité et de sécurité exigés par les environnements de production actuels.

#### **Organizations with world-class digital capabilities release and refresh digital applications much faster than competitors.**

	<b>Traditional</b>	<b>Leading</b>	<b>World-class</b>	<b>Why it matters</b>
 Time to market	<b>1–2 years</b>	<b>2–6 months</b>	<b>8–12 weeks</b>	To compete for consumers on the basis of new tech functionality
 Release frequency	<b>1–4 per year</b>	<b>1–4 per month</b>	<b>10–50 per day</b>	To test and refine the customer experience

**Figure 4.** Infographie sur les "Releases Frequency"

### **3.2.5 Simplicité d'Utilisation**

Enfin, la simplicité d'utilisation de l'interface de gestion de GCP et de son intégration avec divers outils de développement a été un facteur déterminant. L'écosystème GCP, avec ses services intuitifs et bien documentés, permet une courbe d'apprentissage rapide et facilite l'adoption par les développeurs, ce qui est essentiel pour la mise en œuvre efficace de notre projet.

En combinant ces éléments, GCP se présente comme une solution robuste et en adéquation avec nos besoins en matière de performance, de fiabilité et de facilité de gestion pour l'architecture cloud de notre projet.

### **3.2.6 Intérêt Pédagogique des Choix Technologiques**

L'adoption de technologies émergentes et largement utilisées dans notre projet académique ne se limite pas uniquement à la poursuite de l'excellence technique. Elle présente également un intérêt pédagogique considérable pour notre formation et notre employabilité future.

**Acquisition de Compétences en Vogue** L'apprentissage et la maîtrise de technologies de pointe, telles que celles fournies par Google Cloud Platform, confèrent une valeur ajoutée significative à notre profil éducatif. Cela nous prépare non seulement à être à la frontière de l'innovation technologique, mais aussi à être immédiatement opérationnels dans un contexte professionnel post-universitaire.

**Alignement avec les Besoins du Marché** Les choix technologiques que nous avons faits correspondent aux outils et systèmes actuellement en vigueur dans de nombreuses entreprises de renom. Des corporations telles que IKEA, Google, Electronic Arts (EA), Riot Games, et d'autres, intègrent ces mêmes technologies dans leurs opérations quotidiennes. Par conséquent, notre familiarité avec ces environnements nous donne un avantage concurrentiel sur le marché du travail.

## 4 Architecture Technique

Dans cette section, nous explorons l'architecture technique de notre projet, en mettant l'accent sur la comparaison entre les architectures microservices et monolithiques. Cette analyse est importante pour déterminer l'approche la plus adaptée à notre application dans un environnement cloud.

### 4.1 Microservices vs Monolith

L'architecture de toute application logicielle est un facteur déterminant dans sa performance, sa maintenabilité et sa scalabilité. Deux modèles principaux dominent le paysage actuel : les microservices et l'architecture monolithique.

#### 4.1.1 Définition des Microservices

L'architecture microservices se caractérise par la division de l'application en un ensemble de services plus petits et indépendants. Chaque microservice est conçu pour effectuer une fonction spécifique et peut être développé, déployé, opéré et mis à l'échelle de manière indépendante. Cette approche favorise la modularité, permet une plus grande agilité, une meilleure concentration sur les besoins client et facilite l'intégration continue et le déploiement continu (CI/CD).

#### 4.1.2 Définition de l'Architecture Monolithique

Dans une architecture monolithique, tous les composants de l'application sont intégrés en une seule et unique unité. Ce modèle traditionnel permet une simplicité de développement et de déploiement initiale, mais peut devenir complexe et rigide à mesure que l'application grandit, rendant difficile l'ajout en ligne de nouvelles fonctionnalités ou technologies.

### 4.2 Adopter les Microservices dans une Architecture Cloud

La décision d'adopter une architecture microservices pour notre projet s'appuie sur plusieurs considérations clés, en particulier dans le contexte d'une architecture cloud :

- **Scalabilité** : Les microservices permettent une scalabilité fine et dynamique, essentielle dans un environnement cloud où la demande peut fluctuer rapidement.
- **Déploiement Indépendant** : La capacité de déployer des services individuellement réduit les risques et accélère les cycles de mise en production.
- **Résilience** : En cas de défaillance d'un microservice, le reste du système reste fonctionnel, garantissant ainsi une meilleure disponibilité et fiabilité.
- **Polyglottisme** : Cette approche permet l'utilisation de différentes technologies et langages de programmation adaptés à chaque service, favorisant ainsi l'innovation et l'efficacité.
- **Orienté évènements** : L'architecture microservices facilite la conception orientée événements, où les services réagissent aux changements d'état ou aux actions spécifiques, favorisant une communication asynchrone et une réactivité accrue du système.
- **Data driven facilité** : Les microservices supportent une approche "data driven", où les décisions peuvent être prises en temps réel sur la base de données analytiques et métriques, améliorant ainsi la prise de décision et l'adaptabilité.
- **Maintenance et Mise à Jour** : La maintenance et la mise à jour des applications sont simplifiées, chaque microservice pouvant être modifié sans affecter les autres.

En conclusion, l'architecture microservices offre une flexibilité, une évolutivité et une robustesse qui s'alignent avec les exigences dynamiques d'une application cloud. Cette structure

permet une meilleure gestion des ressources, une adaptabilité accrue aux changements et une optimisation de la performance globale du système.

### 4.3 Déploiement des Microservices

L'orchestration efficace des microservices est cruciale pour assurer la performance, la scalabilité et la gestion des ressources de l'application. Dans cet esprit, nous avons évalué plusieurs options offertes par Google Cloud Platform, notamment App Engine, Cloud Run et Google Kubernetes Engine (GKE).

**App Engine** Google App Engine est une plateforme en tant que service (PaaS) qui facilite le déploiement d'applications sans la complexité de la gestion de l'infrastructure. La simplicité de configuration et de déploiement d'App Engine en fait un choix idéal pour des projets nécessitant une mise en œuvre rapide et une gestion simplifiée.

**Cloud Run** Cloud Run est un service entièrement géré qui permet de déployer des conteneurs sans se soucier de l'infrastructure sous-jacente. Bien qu'il offre une grande flexibilité et soit idéal pour les architectures orientées conteneurs, il nécessite une compréhension plus approfondie de la gestion des conteneurs.

**Google Kubernetes Engine (GKE)** GKE est une solution de conteneurisation basée sur Kubernetes qui offre une grande flexibilité et un contrôle puissant. Il est particulièrement adapté aux applications complexes nécessitant une personnalisation poussée de l'environnement d'exécution et d'opération. Cependant, cette flexibilité s'accompagne d'une complexité accrue en termes de configuration et de gestion.

#### 4.3.1 Justification du Choix d'App Engine

Pour notre projet, le choix s'est porté sur Google App Engine pour plusieurs raisons :

- **Simplicité et Rapidité de Déploiement** : App Engine permet un déploiement rapide et facile, réduisant considérablement le temps et les efforts nécessaires à la configuration et à la maintenance de l'infrastructure.
- **Scalabilité Automatique** : Il offre une scalabilité automatique, ce qui est essentiel pour gérer les fluctuations de la demande sans intervention manuelle.
- **Gestion Simplifiée** : La gestion simplifiée de l'infrastructure permet à notre équipe de se concentrer davantage sur le développement des fonctionnalités de l'application plutôt que sur les aspects opérationnels.
- **Coûts** : Grâce à sa structure de coûts basée sur l'utilisation réelle des ressources, le couts de l'App Engine est aligné et proportionnel au nombre d'utilisateur de la plateforme. Cela permet une bonne gestion financière de l'investissement technique.

En somme, bien que Cloud Run et GKE offrent des fonctionnalités plus avancées et une plus grande flexibilité, la simplicité et l'efficacité d'App Engine correspondent parfaitement aux besoins et aux contraintes de notre projet.

### 4.4 Stockage

Le choix de la solution de stockage est un aspect crucial de l'architecture de notre application, impactant directement sa performance, sa scalabilité et sa fiabilité. Nous avons évalué plusieurs services de stockage de données proposés par Google Cloud Platform, y compris Cloud Spanner, AlloyDB et Cloud SQL, avant de finaliser notre choix.

#### 4.4.1 Évaluation des Options de Stockage

- **Cloud Spanner** : Un service de base de données relationnelle globalement distribué qui offre une haute disponibilité, une forte cohérence des données et une scalabilité

horizontale.

- **AlloyDB** : Une option plus récente, optimisée pour les charges de travail PostgreSQL, combinant la performance et la scalabilité d'une base de données cloud avec la compatibilité et les fonctionnalités de PostgreSQL.
- **Cloud SQL** : Un service de base de données entièrement managé qui facilite la configuration, la maintenance, la gestion et l'administration des bases de données relationnelles sur Google Cloud.

#### 4.4.2 Choix de Cloud Spanner

Après une analyse approfondie, nous avons opté pour *Cloud Spanner* pour les raisons suivantes :

- **Scalabilité et Performance** : Cloud Spanner offre une scalabilité exceptionnelle sans compromettre les performances, ce qui est essentiel pour gérer efficacement les charges de travail importantes et fluctuantes de notre application.
- **Cohérence Globale et Haute Disponibilité** : Il assure une cohérence forte des données à l'échelle mondiale et une disponibilité élevée, des caractéristiques indispensables pour notre application distribuée à grande échelle.
- **Maintenance et Gestion Simplifiées** : La gestion simplifiée de la base de données, sans nécessiter une intervention manuelle importante pour la maintenance, permet à notre équipe de se concentrer sur le développement de l'application.

#### 4.4.3 Utilisation d'un Bucket pour les Images

Concernant le stockage des images, l'utilisation d'un bucket, a été choisie pour plusieurs raisons :

- **Séparation des Données** : La séparation des données binaires volumineuses, telles que les images, de la base de données transactionnelle améliore les performances et la gestion des données.
- **Accessibilité et Distribution** : Les buckets offrent une accessibilité élevée et une distribution efficace des ressources statiques, permettant un chargement rapide des images, indépendamment de la localisation géographique des utilisateurs.
- **Scalabilité et Flexibilité** : Ils fournissent une scalabilité automatique et une capacité de stockage pratiquement illimitée, s'adaptant aux besoins changeants de notre application sans intervention manuelle.
- **Sécurité et Contrôle des Accès** : Google Cloud Storage offre des options avancées de contrôle des accès et de sécurité pour protéger les ressources stockées contre les accès non autorisés.

En combinant Cloud Spanner et Google Cloud Storage, nous avons établi une solution de stockage robuste, scalable et performante, adaptée aux exigences techniques et fonctionnelles de notre application.

### 4.5 Gestion des Utilisateurs

La gestion efficace des utilisateurs est un pilier fondamental dans le développement de notre application, impliquant des aspects critiques tels que la sécurité des comptes et la facilité de gestion. Pour atteindre ces objectifs, nous avons choisi d'intégrer Google Identity Platform, une solution complète qui offre plusieurs avantages significatifs.

#### 4.5.1 Sécurisation des Comptes

Google Identity Platform fournit un cadre robuste pour la sécurisation des comptes utilisateurs. Les fonctionnalités clés comprennent :

- **Authentification Forte** : Des mécanismes d'authentification avancés garantissent que seuls les utilisateurs autorisés peuvent accéder à leurs comptes.
- **Protection contre les Menaces** : La plateforme intègre des systèmes de détection et de prévention des menaces, réduisant ainsi le risque de compromission des comptes.

#### 4.5.2 Simplification de la Gestion des Utilisateurs

L'adoption de Google Identity Platform simplifie considérablement la gestion des utilisateurs :

- **Double Authentification** : La plateforme facilite l'intégration de la double authentification, augmentant la sécurité sans compliquer le processus de connexion pour l'utilisateur.
- **Connexion via des Comptes Externes** : Elle permet également aux utilisateurs de se connecter en utilisant leurs comptes existants tels que Gmail, Facebook, GitHub, etc., offrant ainsi une expérience utilisateur fluide et familière.

#### 4.5.3 Coût et Efficacité

Un aspect non négligeable de Google Identity Platform est son rapport efficacité/coût :

- **Modèle de Tarification Avantageux** : La plateforme offre un modèle de tarification compétitif, ce qui est particulièrement avantageux pour les projets avec des contraintes budgétaires.
- **Réduction des Coûts Opérationnels** : L'intégration de cette solution réduit les coûts opérationnels liés à la gestion des utilisateurs, grâce à l'automatisation et à l'efficacité de ses processus.

En conclusion, Google Identity Platform offre une solution complète pour la gestion des utilisateurs, combinant sécurité accrue, facilité de gestion et efficacité en termes de coûts. Cette intégration s'aligne parfaitement avec les objectifs de notre projet, garantissant une gestion utilisateur à la fois sûre et efficiente.

#### 4.6 Google Vision

Une grande partie de notre application est basé sur la détection d'un vin à l'aide d'une image. Notre choix c'est donc évidemment porté sur l'outil Google Vision ayant une capacité avancée de reconnaissance d'image, possédant une fonctionnalité Optical Character Recognition (OCR) permettant d'extraire du texte d'une image. Google Vision possédant une analyse de fiabilité avec le confidence score, cet outil permet d'extraire facilement les informations nécessaires pour identifier le vin pris en photo. Malgré que l'outil soit le plus coûteux de notre projet, il est important de noter qu'il est très efficace en sachant que le texte présent sur les bouteilles utilisent des polices souvent non conventionnelle et potentiellement difficile à reconnaître.

### 5 Choix Technologique pour le Front-End

La conception de notre application mobile a nécessité une analyse des technologies front-end disponibles. Cette section détaille notre processus de sélection, au cours duquel React Native a été identifié comme la technologie la plus adaptée pour répondre à nos besoins spécifiques, en lien avec les objectifs académiques et professionnels du projet.

**Critères de Sélection** Les critères principaux guidant notre choix technologique ont été définis comme suit : la rapidité de développement, les performances, la flexibilité et la compatibilité/intégration avec d'autres technologies.

**Evaluation des Technologies** Nous avons examiné plusieurs frameworks, y compris Xamarin, Flutter et Ionic, en les comparant selon les critères établis. Ce processus d'évaluation est résumé dans le tableau 1.

Framework	Prise en main/language utilisé	Performances	Cas d'utilisations
Xamarin	Moyenne : C#	Bonne	Applications simples
React Native	Rapide : Typescript	Forte	Tout type d'applications
Flutter	Plus longue : Dart	Excellente	Tout type d'applications
Ionic	Rapide : HTML, CSS, Typescript	Bonne	Applications simples

**Table 1.** Comparatif des technologies de développement d'applications mobiles

## 5.1 Justification du Choix de React Native

React Native a été sélectionné pour les raisons suivantes :

**Rapidité de Développement** Grâce à l'utilisation de TypeScript, un langage largement adopté, React Native facilite une montée en compétences rapide et un développement agile, un atout considérable dans le contexte contraint par le temps d'un projet universitaire. De plus nous avions tout deux déjà des bases en React Native.

**Performance** Les tests de performance ont montré que React Native est particulièrement efficace pour les applications nécessitant une interaction utilisateur dense et en temps réel, offrant une expérience utilisateur proche des applications natives.

**Communauté et Support** L'écosystème de React Native bénéficie d'une communauté de développeurs très active et d'un soutien continu, avec un large éventail de composants tiers et des mises à jour régulières, facilitant ainsi la résolution des problèmes et l'apprentissage continu.

**Flexibilité** React Native est réputé pour être un framework polyvalent, capable de gérer une grande variété de types d'applications.

**Compatibilité et Intégration** La compatibilité avec divers outils et services back-end, en particulier ceux offerts par la Google Cloud Platform, permet une intégration fluide et une architecture cohérente.

## 6 Implémentations

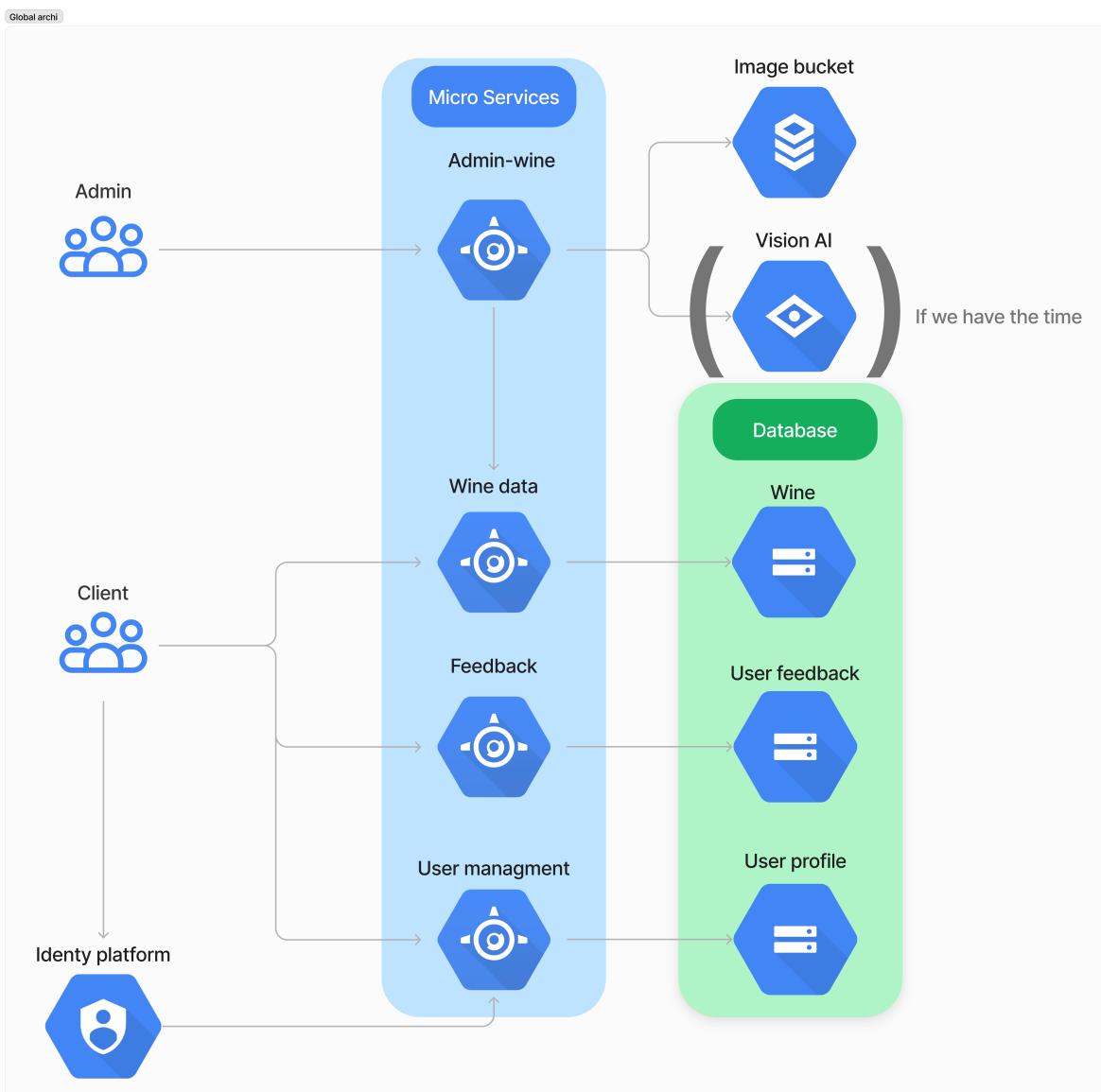


Figure 5. Diagramme d'architecture Technique

### 6.1 Fonctionnement du Front-End

Le front-end de notre projet est conçu comme une application mobile unifiée, destinée à servir à la fois les administrateurs et les utilisateurs finaux. Cette approche permet une gestion cohérente et centralisée des fonctionnalités, tout en offrant une expérience utilisateur personnalisée en fonction du rôle de chaque utilisateur.

#### 6.1.1 Interface Unifiée pour Administrateurs et Utilisateurs

L'application mobile est structurée de manière à fournir une interface utilisateur fluide et intuitive pour tous les utilisateurs, avec des fonctionnalités distinctes en fonction de leur rôle :

- **Utilisateurs Finaux :** Les utilisateurs standards accèdent à l'ensemble des fonctionnalités conçues pour leur expérience, y compris la navigation, l'interaction et l'utilisation des

services offerts par l'application.

- **Administrateurs :** Les utilisateurs avec des priviléges administratifs ont accès à des fonctionnalités supplémentaires, permettant la gestion, la surveillance et la maintenance de l'application. Ces fonctionnalités sont intégrées de manière discrète pour ne pas impacter l'expérience utilisateur standard.

### 6.1.2 Documentation sur l'Utilisation

Pour une compréhension détaillée du fonctionnement et de l'utilisation de l'application, les utilisateurs et les administrateurs peuvent se référer à la section "Fonctionnement de l'Application" de ce rapport.

### 6.1.3 Accès aux Sources

Le code source de l'application front-end est disponible à l'adresse suivante : [https://github.com/Vintellect/Mobile\\_App](https://github.com/Vintellect/Mobile_App). Il est important de noter que le dépôt est privé jusqu'à la fin des rendus pour éviter tout risque de plagiat. L'accès sera rendu public par la suite pour permettre la consultation et l'utilisation de notre travail par la communauté.

Cette approche de conception du front-end assure non seulement une expérience utilisateur optimale, mais permet également une gestion efficace et sécurisée des différentes fonctionnalités de l'application.

## 6.2 Microservice de Gestion des Utilisateurs

Le microservice de gestion des utilisateurs est conçu spécifiquement pour la vérification du statut administratif des utilisateurs. Il joue un rôle prépondérant dans la sécurisation de l'accès aux fonctionnalités administratives de l'application.

### 6.2.1 Fonctionnement du Microservice

Ce microservice est doté d'un point de terminaison unique, `/isAdmin`, qui fonctionne de la manière suivante :

- **Récupération du Token :** Lorsqu'une requête est effectuée à ce point de terminaison, elle doit inclure le token de l'utilisateur comme paramètre.
- **Vérification de l'Identité :** Le service extrait ensuite l'adresse e-mail associée à ce token et procède à la vérification de l'identité de l'utilisateur.
- **Consultation de la Table `is_admin` :** Après avoir obtenu l'e-mail, le microservice consulte la table `is_admin` pour déterminer si l'utilisateur est répertorié comme administrateur.

### 6.2.2 Sécurité et Accès Restreint

La sécurité de ce microservice est renforcée par plusieurs mesures :

- **Restriction d'Accès à la Table :** La table `is_admin` est sécurisée et n'est accessible que via la console de Google Cloud, ce qui empêche tout accès externe non autorisé.
- **Contrôles d'Accès Robustes :** Les contrôles d'accès mis en place garantissent que seuls les membres autorisés de l'équipe du projet ont la capacité de modifier les entrées dans cette table, préservant ainsi l'intégrité de la gestion des rôles administratifs.

En conclusion, ce microservice est essentiel pour assurer que les fonctionnalités administratives de l'application soient sécurisées et limitées aux utilisateurs légitimement autorisés. La conception de ce service met un accent particulier sur la sécurité et la gestion rigoureuse des accès.

## 6.3 Microservice de Feedback

Le microservice dédié au feedback joue un rôle central dans la gestion des avis des utilisateurs. Il interagit spécifiquement avec la table feedback, qui enregistre les informations telles que l'identifiant du vin (`wine_id`), l'identifiant de l'utilisateur (`user_id`), ainsi que la note (`note`) et le commentaire (`comment`) attribués par l'utilisateur au vin concerné.

### 6.3.1 Fonctionnalités du Microservice

Les fonctionnalités clés de ce microservice sont les suivantes :

- **Récupération des Avis** : Il permet la récupération des avis des utilisateurs en fonction du vin spécifié, facilitant ainsi l'accès à des retours ciblés et pertinents.
- **Ajout d'Avis** : Le microservice offre la possibilité d'ajouter un nouvel avis, comprenant un commentaire et une note. Il est à noter qu'un utilisateur ne peut émettre qu'un seul avis par vin, assurant ainsi l'intégrité des retours.
- **Modification d'Avis** : Les utilisateurs peuvent modifier leurs avis, mais cette action nécessite une vérification de l'identité de l'utilisateur pour s'assurer que seuls les auteurs des avis puissent les modifier.
- **Suppression d'Avis** : La suppression d'avis est également possible et est conditionnée à l'authentification de l'utilisateur en tant qu'administrateur ou propriétaire de l'avis. Cette fonctionnalité est essentielle pour maintenir le contrôle et la pertinence des avis publiés.

### 6.3.2 Gestion des Contenus et Sécurité

Ce microservice est conçu pour garantir une gestion efficace et sécurisée des avis :

- **Contrôles d'Accès** : Des contrôles d'accès stricts sont mis en place pour s'assurer que seuls les utilisateurs autorisés peuvent modifier ou supprimer des avis.
- **Intégrité des Données** : Des mécanismes sont en place pour préserver l'intégrité des données, notamment en empêchant les duplications d'avis par un même utilisateur pour un vin donné.

L'objectif de ce microservice est d'offrir une plateforme fiable et sécurisée pour la gestion des avis, renforçant ainsi la confiance des utilisateurs dans l'application et assurant la qualité des informations partagées.

## 6.4 Microservice Wine

Le microservice Wine constitue le cœur de notre application, jouant un rôle essentiel dans la récupération des informations relatives aux vins stockés dans notre base de données. Il offre une flexibilité considérable dans la manière dont les données peuvent être interrogées.

### 6.4.1 Fonctionnalités du Microservice

Ce microservice fournit plusieurs modes de récupération, offrant ainsi une grande variété d'options pour répondre aux besoins spécifiques des utilisateurs :

- **Récupération de Plusieurs Vins** : Les utilisateurs peuvent récupérer un groupe de vins enregistrés dans la base de données.
- **Récupération d'un Vin Spécifique** : Il est possible de récupérer les détails d'un vin spécifique, en fournissant des identifiants uniques tels que le code barre ou l'ID du vin.
- **Récupération des Vins Avant ou Après une Date** : Les utilisateurs peuvent filtrer les vins en fonction de leur date de vendange, permettant ainsi de cibler des vins selon une période spécifique.

- **Récupération des Vins en Fonction d'une Année :** Cette option permet de récupérer des vins en se basant uniquement sur leur année de production. Ces diverses fonctionnalités rendent le microservice Wine extrêmement puissant et polyvalent, facilitant l'accès aux données sur les vins pour les utilisateurs de l'application.

## 6.5 Administration des Vins

Le microservice dédié à l'administration des vins joue un rôle clé dans la gestion et le maintien de la base de données des vins. Ce service est conçu pour assurer non seulement l'insertion efficace des données relatives aux vins, mais aussi pour garantir la cohérence et la sécurité des informations stockées.

### 6.5.1 Fonctionnalités du Microservice

Les principales fonctions de ce microservice sont les suivantes :

- **Insertion des Données :** Le service permet l'ajout de nouvelles entrées de vins dans la base de données, veillant à ce que toutes les informations nécessaires soient correctement et complètement enregistrées.
- **Gestion des Jointures de Tables :** En cas de besoin, ce microservice effectue les jointures entre différentes tables, facilitant ainsi la récupération et l'analyse de données complexes et interdépendantes.

### 6.5.2 Sécurité et Intégrité des Données

L'aspect le plus important de ce microservice est la mise en œuvre de mesures de sécurité rigoureuses :

- **Prévention des Injections SQL :** Des mécanismes anti-injection SQL sont intégrés pour protéger la base de données contre les attaques et les manipulations malveillantes. Ces mécanismes garantissent que toutes les requêtes et les transactions sont sécurisées et fiables.
- **Vérification des Données :** Le microservice s'assure de la validité des données insérées, empêchant l'enregistrement d'informations incohérentes ou erronées.

Ces fonctionnalités et mesures de sécurité assurent une gestion efficace et sûre des données sur les vins. Le microservice d'administration des vins est donc essentiel pour maintenir l'intégrité, la cohérence et la sécurité des données de l'application.

## 7 Fonctionnement de l'application

### 7.1 Page de connexion

Cette page est la toute première page de l'application, ayant décidé que chaque utilisateur devait créer un compte et se connecter, chaque utilisateur de l'application devra obligatoirement y passer. Cette page possède deux versions, une pour créer un compte l'autre pour s'y connecter.

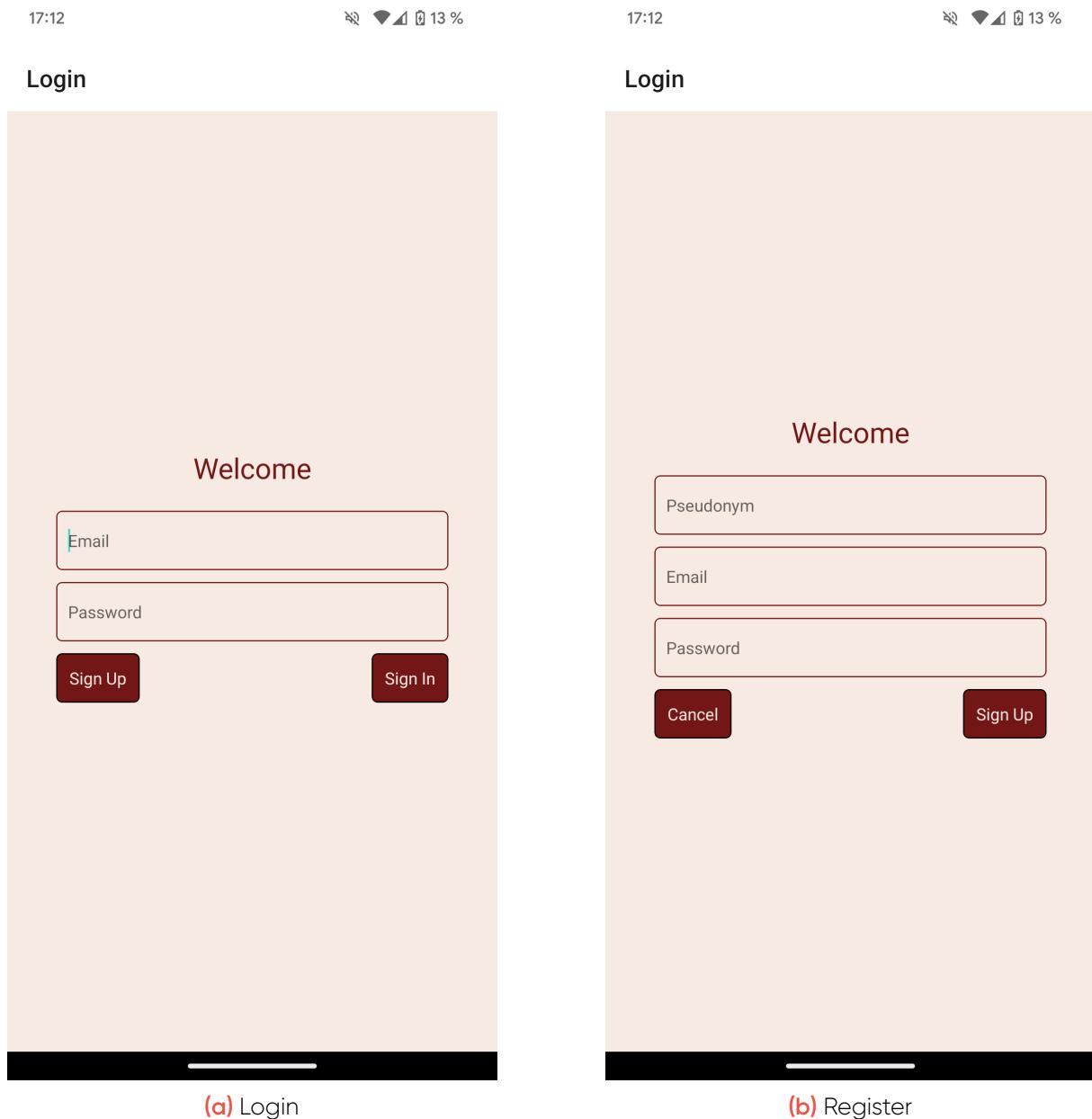
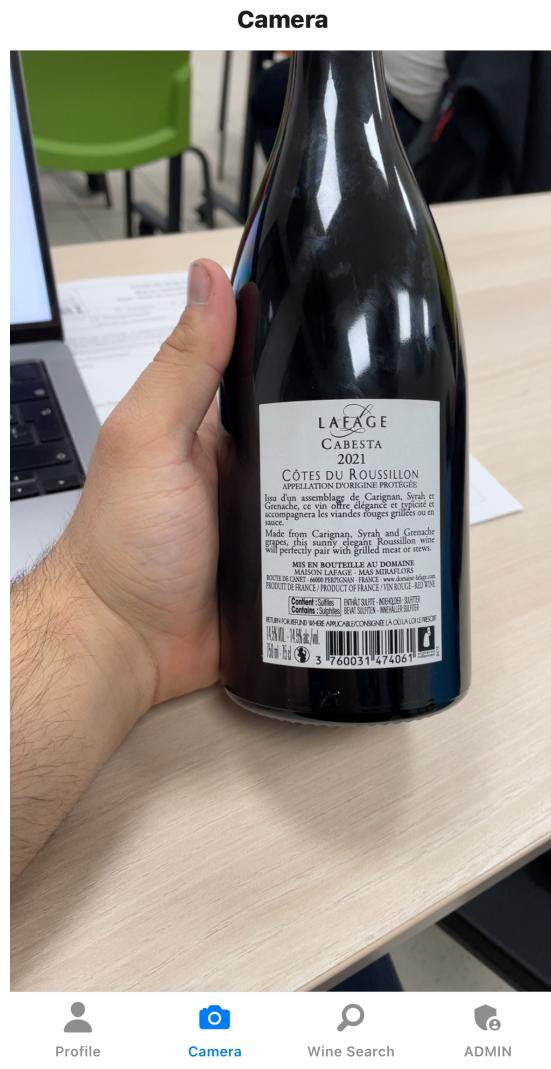


Figure 6. Page de connexion (Android)

## 7.2 Page scan d'un vin

Une fois l'utilisateur connecté, celui-ci arrive sur la première page de l'application. Cette page à pour utilité de scanner un code barre présent sur la bouteille de vin. Une fois le code barre automatiquement reconnu, on effectue une vérification de la présence de la bouteille dans la base de données, puis ensuite la page détaillé du vin est ouverte (cf. section 7.4).



**Figure 7.** Scanner (iOS)

### 7.3 Page de Liste des Vins

Cette page permet à l'utilisateur de parcourir la liste de tous les vins disponibles dans la base de données. Il peut également cliquer sur une carte de vin pour accéder aux informations ainsi qu'aux avis des utilisateurs présents dans la page détaillée du vin (cf. section 7.4).

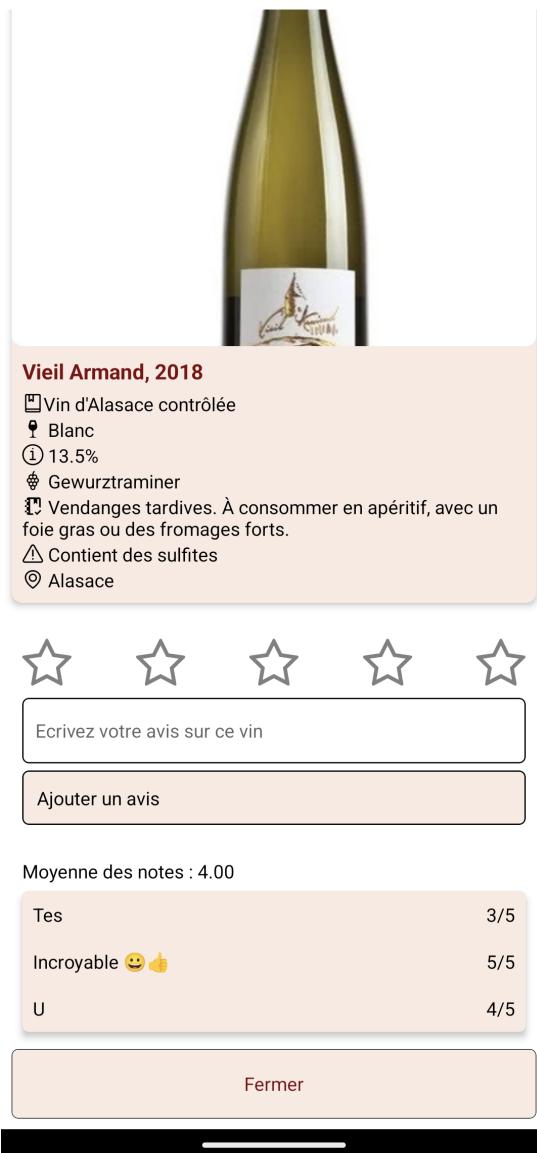
## Wine Search



Figure 8. Liste des Vins (Android)

**7.4 Page détaillé d'un vin**

Cette page présente en détail les informations relatives à un vin. L'utilisateur à la possibilité d'attribuer une note de 1 à 5 avec un commentaire. Il peut aussi consulter les commentaires mis par les utilisateurs sur ce vin. En maintenant son doigt appuyé sur son commentaire, le modifier ou le supprimer. Les administrateur ont, quand à eux, la possibilité de supprimer n'importe quel commentaire, mais ne sont pas autorisés à le modifier, pour des raisons de sécurité.

**Figure 9.** Détail d'un vin (Android)

## 7.5 Page profile

Cette page est là pour toute la gestion du profile de l'utilisateur. Elle pourra être développer à l'avenir, mais aujourd'hui elle n'est utile que pour la déconnexion d'un utilisateur. On pourra à l'avenir ajouter plus d'option, comme le choix de langue de l'application, le choix d'un thème de couleur. Afficher l'historique d'activité, des réglage de préférence sur les notifications de l'application. Il serait aussi judicieux d'ajouter une section de modification des informations personnelles de l'utilisateur, comme son adresse e-mail, son mot de passe,

...

19:37 ☺ 🔋

41 %

### Profile

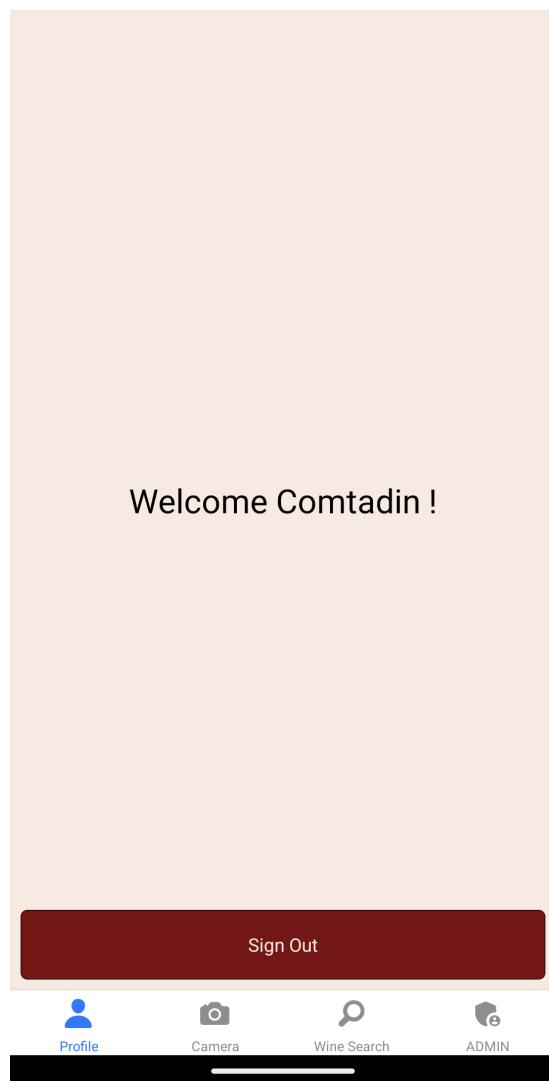
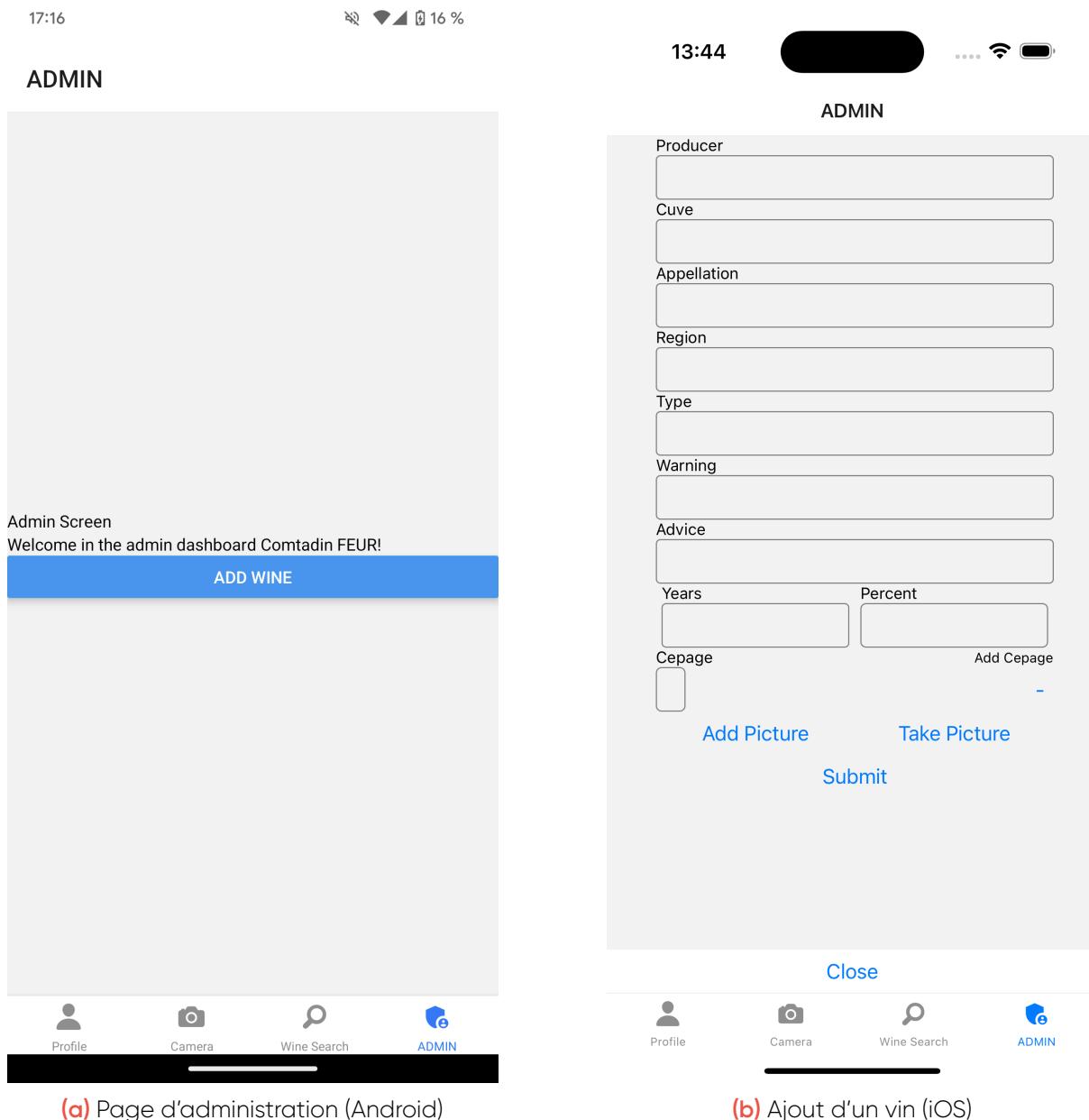


Figure 10. Profile (Android)

## 7.6 Administration

Cette portion de l'application n'est disponible uniquement par les administrateur de celle-ci. Elle permet de scanner un vin pour l'ajouter dans la base de données si celui-ci n'existe pas déjà. Toute la partie administration n'étant pas disponible pour l'utilisateur lambda, le style du composant importe très peu, c'est donc pour ça que le composant peut être visuellement simple.



**Figure 11.** Administration

## 8 Conclusion

### 8.1 Conclusion d'Ange

Le projet, dans son ensemble, présente une durée relativement courte par rapport aux exigences finales spécifiées. Malgré cette concision, l'expérience s'est révélée notable en raison de la flexibilité accordée quant aux choix technologiques. Cela nous a permis d'explorer de nouvelles technologies qui n'ont pas été abordées dans les autres cours. La mise en œuvre réussie de diverses technologies dans le cadre de ce projet offre une perspective enrichissante, me permettant d'enrichir mes compétences professionnelles.

## 8.2 Conclusion d'Emmanuel

Je suis très satisfait du résultat obtenu, qui a donné lieu à de nouvelles réflexions. Ce projet m'a permis de mieux comprendre et d'utiliser des technologies avancées telles que GCP et React Native. J'ai apprécié la liberté accordée dans la réalisation de ce projet concret. La collaboration a été efficace et fructueuse.

Je suis convaincu que cette expérience constitue une contribution importante à mon parcours professionnel et mérite une place de choix dans mon CV.