

DOSSIER PROJET



Application de mise en relation entre producteurs agricoles locaux et acheteurs

Kevin VELLARD

Concepteur développeur d'application



Avant-propos-----	2
Remerciements-----	3
Liste des compétences du référentiel-----	4
Introduction-----	5
L'entreprise-----	6
Présentation de l'entreprise-----	6
Objectifs de l'entreprise-----	6
Planification et gestion du temps-----	7
I. Compétences couvertes par ce projet-----	8
II. Cahier des charges-----	9
1.Présentation du projet-----	9
2.Fonctionnalités principales-----	9
3.Specifications techniques-----	9
4.Interface utilisateur-----	9
5.Gestion de projet-----	10
6.Tests et qualité-----	10
7.Maintenance et évolution-----	10
8.Aspects juridiques et conformité-----	10
9.Budget et ressources-----	10
10.Livrables attendus-----	10
III. Description du projet-----	11
1. Préambule-----	11
2. Objectifs-----	11
3. Application Web Coté Front-End-----	12
a) logo - slogan-----	12
b) formulaire de connexion et d'inscription - Application mobile-----	13
c) page profil (producteur, acheteur) - Application mobile-----	15
4. Page destinée aux producteurs-----	16
IV. Spécificités Techniques-----	17
1) Sécuriser le site web-----	18
2) Responsive-----	18
a) Version application mobile-----	18
b) Version web & ses modes responsives-----	20
V. Réalisation de l'application Web & Mobile-----	21
1. Outils-----	21
a) Outils de production-----	21
b) Outils organisationnels-----	24
2. Maquettage de l'application-----	24
3. Création de la Base de Données-----	25
4. Application Web avec NextJS-----	28
5. Backend avec Node.js et Express-----	29
6. Application mobile avec React-Native-----	30
7. Les Composants-----	31
8. Création de formulaires-----	36
9. Sécurité (login, rôles, authContext)-----	39
a) Login:-----	39
b) Rôles:-----	42
c) AuthContext:-----	44
10. Les requêtes http avec Fetch-----	46
11. Les Tests-----	48
VI. Perspectives Futures-----	49
VII. Conclusion-----	51

Avant-propos

Le présent document est un élément de présentation des activités de développement réalisées dans et autour du parcours de formation suivi lors de la promotion 2024 de l'organisme ACGD à Bordeaux.

Il présente une réalisation professionnelle en vue de l'obtention du titre professionnel “Concepteur Développeur d'Application”.

Remerciements

Avant tout développement sur le projet effectué durant la formation, il apparaît opportun de commencer ce rapport par des remerciements, à ceux qui m'ont beaucoup appris au cours de cette formation, et aussi à ceux qui ont eu la gentillesse de faire de ce projet un moment très profitable.

Aussi, je remercie :

- **Jérémie CHABANAIS** mon formateur, qui m'a accompagné tout au long de cette formation
- l'entreprise **Mulot Learning**, qui m'a permis de m'exercer et me confronter au quotidien du développeur d'application dans un environnement coworking axé start-up très stimulant. Je remercie son créateur et gérant, **Mathieu NYEMB**, ainsi que son ami et développeur d'applications, **Antoine Kourmanalieva**.
- Ma promotion CDA 2024, étant constituée de profils très différents et très enrichissants.

Liste des compétences du référentiel

Ci-dessous se trouve la liste des compétences utilisées dans le projet présenté dans ce dossier.

1) Développer une application sécurisée :

Installer et configurer son environnement de travail en fonction du projet.

Développer des interfaces utilisateur

Développer des composants métier

Contribuer à la gestion d'un projet informatique

2) Concevoir et développer une application sécurisée organisée en couches :

Analyser les besoins et maquetter une application

Définir l'architecture logicielle d'une application

Concevoir et mettre en place une base de données relationnelle

Développer des composants d'accès aux données SQL et NoSQL

3) Préparer le déploiement d'une application sécurisée :

Préparer et exécuter les plans de tests d'une application

Préparer et documenter le déploiement d'une application

Contribuer à la mise en production dans une démarche DevOps

Introduction

Depuis mon enfance, j'ai toujours été fasciné par l'univers de l'informatique, un domaine qui m'a captivé par ses possibilités infinies et son impact sur notre quotidien. Cette passion, bien que parfois mise de côté au profit d'autres intérêts académiques et professionnels, a toujours été présente en moi. Après avoir obtenu un Master MBA en Management des Métiers du Sport à l'Inseec Paris, ainsi qu'un Master 2 en Entraînement, Biologie, Nutrition et Santé à l'Université Paris Descartes, j'ai eu l'opportunité d'explorer divers secteurs professionnels. Mes expériences dans le domaine commercial, notamment dans l'immobilier et les dispositifs médicaux, m'ont permis de développer des compétences solides en vente, en négociation et en gestion de la relation client.

Cependant, au fil des années, j'ai ressenti un besoin croissant de me réorienter vers le développement informatique. Cette envie de fusionner ma passion pour la technologie avec mes compétences en management m'a conduit à entreprendre une formation en tant que concepteur-développeur d'applications. Je suis convaincu que le numérique offre des solutions innovantes aux défis contemporains, et je souhaite contribuer à cette évolution.

C'est dans ce contexte que j'ai conçu TerroTerro, une plateforme fictive innovante fondée en 2024. Ce projet est né d'une volonté de promouvoir l'agriculture durable tout en créant un lien direct entre consommateurs et producteurs locaux. TerroTerro vise à faciliter l'accès à des produits frais et locaux pour les consommateurs, tout en offrant une vitrine aux producteurs régionaux qui souhaitent valoriser leur savoir-faire. Je crois fermement que chaque achat local contribue non seulement à soutenir l'économie locale, mais aussi à renforcer les liens communautaires.

En développant cette application, je souhaite allier toutes mes compétences à mon expertise naissante dans le développement informatique. Mon objectif est de créer une solution qui non seulement répond aux besoins des consommateurs modernes, mais qui encourage également des pratiques d'achat responsables et solidaires. En somme, TerroTerro représente pour moi une belle opportunité de mettre à profit mon parcours diversifié tout en œuvrant pour un avenir plus durable et connecté.

L'entreprise

Présentation de l'entreprise

TERRROTTERRO est une plateforme fictive innovante fondée en 2024, dédiée à la promotion de l'agriculture durable et à la création d'un lien direct entre consommateurs et producteurs locaux.

Cette application vise à faciliter l'accès à des produits frais et locaux pour les consommateurs tout en offrant une vitrine aux producteurs régionaux. L'entreprise croit fermement que chaque achat local contribue à renforcer l'économie et à souder la communauté.

Objectifs de l'entreprise

- 1) Promouvoir l'agriculture durable dans la région
- 2) Créer un lien direct entre producteurs locaux et consommateurs
- 3) Soutenir l'économie locale et préserver le patrimoine culinaire
- 4) Encourager des choix alimentaires durables et responsables

TERRO TERRO se distingue par son engagement envers la transparence, la durabilité et le soutien à la communauté agricole locale, offrant ainsi une alternative éthique et écologique aux modes de consommation traditionnels.

Organisationnel et humain

L'équipe est composé de trois étudiants :

- Patience Koribirama, qui a finalement décidé d'utiliser le langage Symfony de son côté, et que l'on remercie pour son apport collectif en amont.
- Enzo Brison,
- Kévin Vellard (moi-même).

Planification et gestion du temps

Dans le domaine numérique, notamment pour un concepteur développeur d'applications, maîtriser la gestion du temps est essentiel pour livrer des projets de qualité dans les délais impartis, tout en maintenant un équilibre professionnel sain. Cela nécessite une planification minutieuse, une priorisation efficace des tâches et une capacité d'adaptation aux imprévus.

Nous avons utilisé l'outil en ligne **Trello** pour séparer aussi bien que possible les tâches, avec plusieurs colonnes, et un code couleur pour stratifier les degrés d'importance de ces dernières et pour suivre la progression de chaque tâche.

Nous avons également utilisé des outils comme **Microsoft Teams**, afin d'échanger quotidiennement et de manière interactive sur le projet.

Au début du projet, notre équipe a adopté une approche de répartition des tâches basée sur des domaines de compétences spécifiques : front-end, back-end et développement mobile. J'ai initialement pris en charge le développement de l'application mobile.

Cependant, suite au départ imprévu d'un membre de l'équipe et guidés par un souci d'efficacité, nous avons dû revoir notre organisation en cours de projet. Cette réorganisation

s'est avérée bénéfique, car elle nous a permis d'optimiser nos ressources et de nous adapter aux défis.

Dans ce contexte, mes responsabilités se sont élargies pour englober :

1. Le développement complet de l'application mobile
2. La gestion du back-end spécifique à l'application mobile, qui diffère par endroits de celui du front-end desktop
3. Une petite contribution au développement du front-end desktop, en soutien à mon collègue

Cette évolution de mes attributions s'est révélée être une opportunité enrichissante. Elle m'a permis de développer de nouvelles compétences, d'approfondir ma compréhension globale du projet et d'acquérir une expérience précieuse dans des domaines variés du développement logiciel.

I. Compétences couvertes par ce projet

Ce projet m'a permis de passer en revue de nombreuses compétences acquises et nécessaires à l'obtention de ce titre professionnel :

La partie back-end a été conçue en NodeJs, la partie application web en NextJs, et la partie application mobile en React-native.

- Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité :

- Maquetter une application
- Réaliser une interface utilisateur web
- Développer des composants d'accès aux données
- Développer une interface utilisateur web dynamique
- Réaliser une interface avec une solution de paiement

- Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité

- Créer une base de données
- Développer les composants d'accès aux données
- Développer la partie back-end d'une application web ou web mobile
- Développer des composants métiers
- Construire une application organisée en plusieurs couches

II. Cahier des charges

1. Présentation du projet

- Contexte et objectifs
- Public cible (producteurs locaux, consommateurs)
- Valeurs du projet (durabilité, transparence, soutien local)

2. Fonctionnalités principales

- Inscription et profils utilisateurs (producteurs et acheteurs)
- Système de mise en vente des produits
- Moteur de recherche et filtres pour les produits
- Système de messagerie interne
- Système de paiement sécurisé (stripe)
- Géolocalisation des producteurs
- Système de notation et avis

3. Spécifications techniques

- Architecture (Node.js pour le backend, Next.js pour le frontend web, React Native pour l'application mobile)
- Base de données (MySQL)
- Hébergement et déploiement
- Sécurité et protection des données

4. Interface utilisateur

- Design responsive pour web et mobile

- Maquettes des principales interfaces
- Charte graphique

5.Gestion de projet

- Méthodologie de développement (Agile)
- Planning prévisionnel
- Équipe et rôles

6.Tests et qualité

- Plan de tests (unitaires, d'intégration, utilisateur)
- Critères de qualité et de performance

7.Maintenance et évolution

- Plan de maintenance
- Mises à jour futures envisagées

8.Aspects juridiques et conformité

- RGPD et protection des données personnelles
- Conditions d'utilisation et mentions légales

9.Budget et ressources

- Estimation des coûts de développement
- Ressources humaines et matérielles nécessaires

10.Livrables attendus

- Code source
- Documentation technique et utilisateur

III. Description du projet

1. Préambule

Les langages front-end sont Nextjs pour la partie web, et la partie application mobile en React-native.

- Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité
 - Maquetter une application
 - Réaliser une interface utilisateur web et web-mobile

2. Objectifs

L'objectif principal est de développer une application mobile et web qui met en relation directe les producteurs locaux et les acheteurs, favorisant ainsi une consommation locale et durable.

On liste ensuite les objectifs spécifiques :

- a. Concevoir et développer une plateforme technologique (application mobile et site web) fonctionnelle et conviviale utilisant JavaScript, Node.js, Next.js et React Native.
- b. Simplifier le processus de vente directe entre producteurs et acheteurs en automatisant certaines tâches (mise en ligne des produits, gestion des commandes,...).
- c. Implémenter un système de géolocalisation permettant aux acheteurs de trouver facilement les producteurs locaux.
- d. Mettre en place un système de profils détaillés pour les producteurs, incluant des informations sur leurs pratiques agricoles, leurs produits et leur localisation.
- e. Intégrer un système de messagerie interne sécurisé pour faciliter la communication entre producteurs et acheteurs.
- f. Développer un système de paiement sécurisé pour les transactions au sein de l'application.
- g. Créer un système de notation et d'avis pour renforcer la confiance entre les utilisateurs.
- h. Assurer la conformité de l'application avec les réglementations en vigueur, notamment en matière de protection des données personnelles (RGPD).

On liste enfin les objectifs techniques :

a. Garantir la performance et la scalabilité de l'application pour supporter un nombre croissant d'utilisateurs.

b. Assurer la sécurité des données des utilisateurs et des transactions.

c. Développer une interface utilisateur intuitive et responsive pour une expérience optimale sur tous les appareils.

Ces objectifs devraient être mesurables, atteignables et temporellement définis.

3. Application Web Coté Front-End

- Inscription et connexion
- Chaque acheteur ou producteur aura sa propre page profil et pourra lister ses spécificités techniques et compétences
 - Le parcours d'inscription et de navigation doit pouvoir être réalisé de façon simple.
 - Avoir la possibilité de sélectionner ses produits, voir les "stacks" demandés, la zone géographique etc.
 - Les utilisateurs devront créer un compte et être connectés pour pouvoir regarder les produits, ou les producteurs
 - Création d'un tableau de bord pour les producteurs :
 - Listing de leurs produits mis en ligne.
 - Suivi de la quantité de produits restants par fiche produit.
 - Création d'une page de contact via un formulaire.
 - Donner la possibilité à chaque producteur de voir, modifier, supprimer, ajouter leurs propres produits et de voir les produits des autres producteurs (prix, quantités, description) .
 - Le site devra être dynamique et consultable sur ordinateur et mobile.

Nous sommes partis d'une application inexistante.

a) logo - slogan

Nous avons hésité entre plusieurs logos possibles tout en gardant des couleurs et un esprit cohérent avec le domaine abordé :



Nous avons finalement choisi ce logo :



TERROTERR

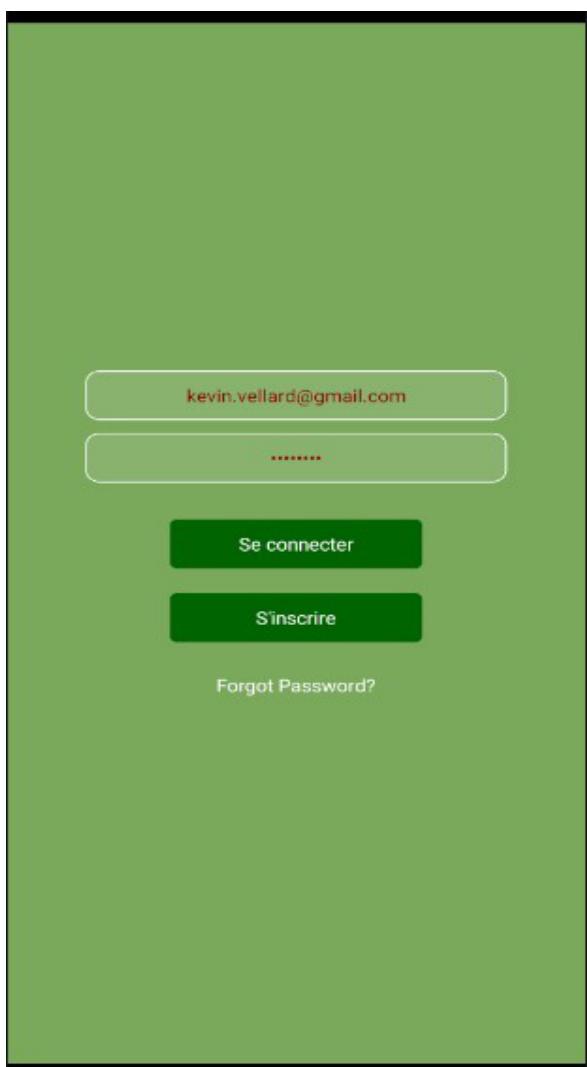
Pour le slogan, nous avons gardé le thème de la terre et de la gastronomie tout en maintenant un esprit impactant: "De la terre à votre table"

b) formulaire de connexion et d'inscription - Application mobile

Pour l'application, le visiteur doit être utilisateur pour accéder à l'application. Ainsi, il est dirigé naturellement vers une page d'accueil lui donnant accès à 2 choix (se connecter ou s'inscrire). Un choix annexe lui permet d'accéder à un système de réinitialisation de mot de passe.

Dans le cas d'une première utilisation, on choisit l'inscription: cette dernière propose 2 types d'utilisation, celui de client ou de vendeur. Les champs sont libellés avec des placeholders (ici remplacés par les entrées de l'utilisateur).

Si l'utilisateur choisit finalement d'être vendeur, un champ "raison sociale" s'ajoute immédiatement au formulaire. Tous les champs doivent obligatoirement être dûment remplis, et avec les contraintes standards et non-standards indiquées dans le fichier code du formulaire.



Inscription

Type d'utilisateur :

la ferme aquacole

kevin

vellard

kevin.vellard@gmail.com

0606060606

.....

5 rue du paradis

Bordeaux

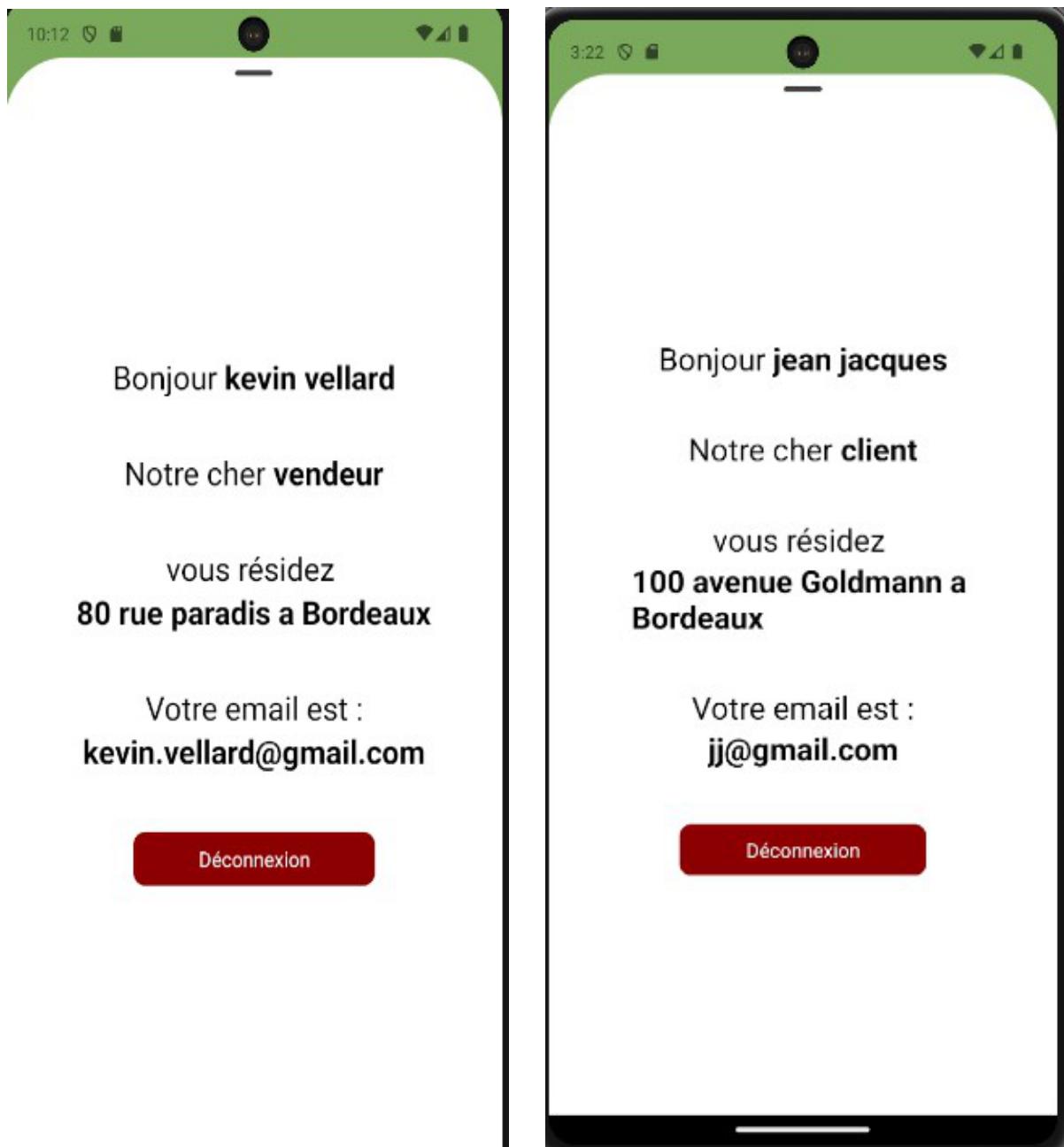
33000

[Retour à la connexion](#)

c) page profil (producteur, acheteur) - Application mobile

React-Native nous offre la possibilité de concevoir l'application autrement.

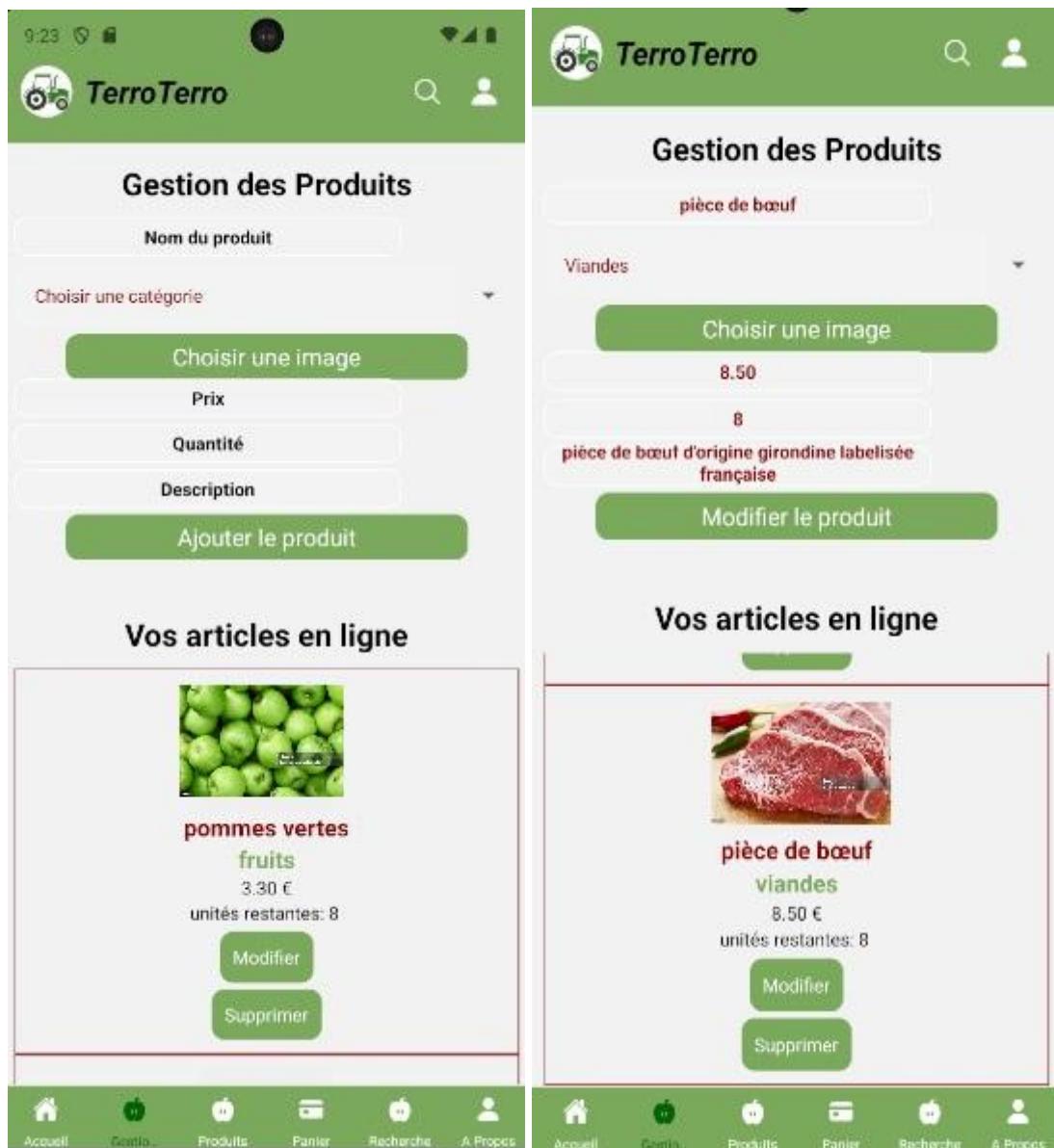
Ainsi, nous avons choisi de pouvoir afficher la fiche profil du principal intéressé, à tout moment en cliquant sur l'icône 'person' présent sur le header. Nous avons utilisé l'ouverture d'une modale, configurée pour s'ouvrir à 95% de la hauteur de l'écran (on peut bien évidemment facilement modifier ces paramètres, et même ajouter des strates (ou étapes)



d'ouverture de la modale.

4. Page destinée aux producteurs

Avec React-Native, nous avons créé un onglet spécifique à l'utilisateur dont le rôle est producteur. Il peut ainsi créer une fiche produit directement depuis son téléphone, sélectionnant une catégorie de produit, puis en lui donnant un nom, un prix, une quantité disponible, une description plus détaillée ainsi qu'en illustrant le tout avec une image (soit récupérée en local, soit en utilisant l'appareil photo intégré à son téléphone portable).



Par la suite, le producteur peut, bien évidemment modifier chacunes de ces informations. Ayant accès aux produits de tous les producteurs présents sur l'application, il peut ainsi comparer les produits, et ajuster ses prix en conséquence par exemple. Enfin, il existe la fonctionnalité de suppression d'un produit.

IV. Spécificités Techniques

Afin de répondre au mieux à nos attentes j'ai fait le choix d'utiliser NodeJs pour le coté back-end, le langage NextJs pour la partie web et son framework React Native pour la partie mobile, et expo pour avoir un rendu direct sur l'émulateur. Ici, nous avons la ligne pour installer React et toutes ses dépendances dont expo :

```
● ● ●  
npm install --global expo-cli
```



Tout d'abord : qu'est-ce qu'un **framework** ?

Littéralement “cadre de travail” en anglais, un framework est une sorte de boîte à outils destinée à simplifier le travail du développeur informatique.

Généralement spécifique à un langage de programmation, un framework propose l'accès à une architecture préétablie et des composants génériques, permettant ainsi de ne pas partir de zéro à chaque nouveau projet.

C'est donc un gain de temps tout en offrant la certitude de coder en conformité avec les conventions du langage utilisé. Ce cadre standardisé favorise également la maintenance et l'évolutivité des projets développés, qu'il s'agisse d'applications ou de sites web.

1) Sécuriser le site web

Il est nécessaire de passer le nom de domaine du site en https, HyperText Transfer Protocol Secure, littéralement “protocole de transfert hypertexte sécurisé” est la combinaison du HTTP avec une couche de chiffrement SSL ou TLS.

Le HTTPS permet au visiteur de vérifier l'identité du site web auquel il accède, grâce à un certificat d'authentification émis par une autorité tierce, réputée fiable. Il garantit théoriquement la confidentialité et l'intégrité des données envoyées par l'utilisateur (notamment les informations entrées dans les formulaires) et reçues du serveur. Il permet de valider l'identité du visiteur, si celui-ci utilise également un certificat d'authentification client.

2) Responsive

Un site web adaptatif est un site web dont la conception vise, grâce à différents principes et techniques, à offrir une consultation confortable sur des écrans de tailles très différentes. Un Site web adaptatif est appelé aussi responsive, les enjeux du responsive design sont nombreux et ils ne sont pas uniquement techniques.

L'utilisation d'Internet sur mobile en France a considérablement augmenté depuis 2017. En 2024, le taux de pénétration des smartphones en France est très élevé, avec environ 95% de la population adulte possédant un smartphone. Bien que les chiffres spécifiques à la France ne soient pas mentionnés, dans des marchés similaires comme les États-Unis, le temps moyen passé sur les applications mobiles est passé d'environ 3 heures par jour à 5 heures par jour en deux ans. Le m-commerce continue de croître, avec des prévisions de ventes mondiales atteignant 3,4 billions de dollars d'ici 2027.

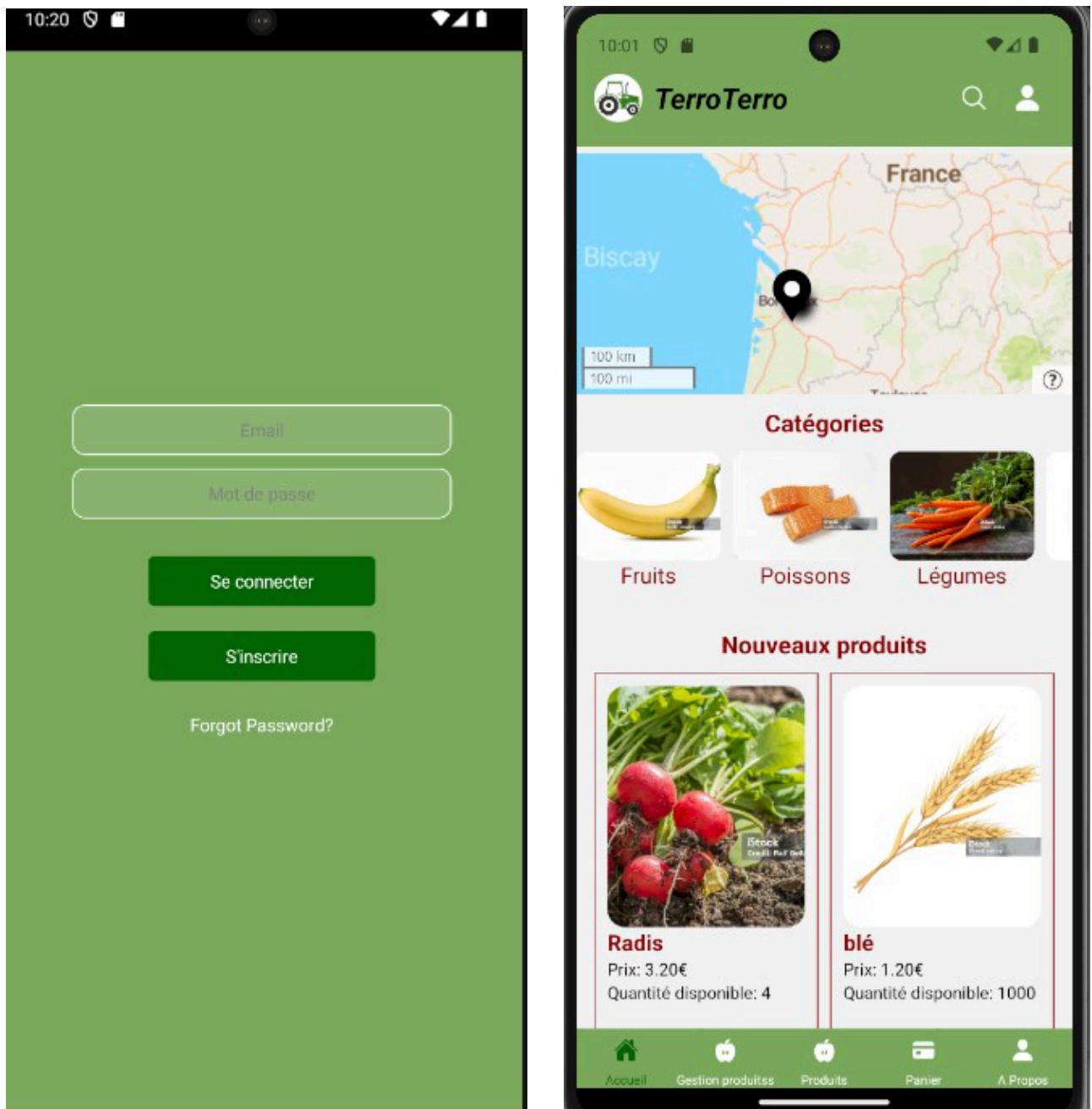
Ces tendances confirment la croissance continue de l'utilisation mobile et renforcent l'importance du développement "mobile-first" et du design responsive. L'augmentation du temps passé sur les applications mobiles et la croissance du m-commerce soulignent également l'importance croissante des stratégies centrées sur le mobile pour les entreprises.

**: L'approche mobile first consiste à concevoir un site en commençant par sa version mobile, puis ajouter des améliorations pour les écrans plus grands.*

a) Version application mobile

Ici nous avons le mode application mobile, qui est responsive nativement : nous avons utilisé une page d'accueil, permettant d'accéder à l'application fonctionnelle par un simple login. A défaut, en cas d'oubli du mot de passe utilisateur, ce dernier a accès à la

page de réinitialisation de son mot de passe, et si le visiteur n'est pas encore inscrit, il a accès à la page d'inscription pour vendeur et/ou acheteur.



Une fois l'accès à l'application passé, l'utilisateur est dirigé vers la page principale, reprenant les principales fonctionnalités de l'application.

Une "BottomBar" de 4 onglets pour le client (1 de plus pour le producteur, lui permet de gérer la création, modification, suppression de ses produits en ligne) leur facilite la navigation et la compréhension de l'application, en naviguant entre la page d'accueil , la page produits, la page panier, et la page « à propos ». S'ajoute à cela, un « header » modulable pour la recherche de produits (ou de producteurs dans le futur), ainsi qu'une icône « person » permettant l'ouverture d'une modale englobant 95% de l'affichage, et permettant d'accéder à toutes les informations de son profil.

b) Version web & ses modes responsives

The image displays two responsive web application interfaces. The top interface features a map of France with major cities labeled (Paris, Lyon, Marseille, etc.) and a search bar at the top. The bottom interface shows a grid of food products under categories: 'À la une' (Filet mignon de porc, Escalope de dinde, Entrecôte de bœuf, Gigot d'agneau, Jambon sec, Camembert de Normandie AOP, Yaourt nature grec), 'Viandes' (Filet mignon de porc, Escalope de dinde, Entrecôte de bœuf, Gigot d'agneau, Jambon sec, Filet mignon de porc, Escalope de dinde), and 'Produits laitiers' (represented by small thumbnail images). Both interfaces include a green header bar with icons for search, cart, and user profile, and a bottom navigation bar with four items.

V. Réalisation de l'application Web & Mobile

1. Outils

a) Outils de production

J'ai eu recours pour ce projet à l'utilisation de différents outils nécessaires au bon déroulement de celui-ci.

⇒ FIGMA

Tout d'abord j'ai utilisé l'outil Figma pour réaliser le maquettage de mon application : un outil de design collaboratif basé sur le cloud, largement utilisé pour la conception d'interfaces utilisateur (UI) et d'expériences utilisateur (UX).

L'interface intuitive de Figma permet une utilisation simple via des fonctionnalités bien organisées et accessibles. Ces dernières aident à laisser des commentaires et à interagir directement sur les designs, ce qui améliore la collaboration entre designers, et développeurs. Enfin, Figma conserve un historique des modifications, permettant aux utilisateurs de revenir à des versions antérieures si nécessaire.



⇒ GITHUB

Développant mon projet sur ordinateur portable, il était bien évidemment indispensable pour moi d'utiliser Git ainsi que Github et Github desktop.

J'ai aussi bien travaillé en lignes de commandes Git qu'avec Github desktop, le but était pour moi de pouvoir aussi bien me sentir à l'aise dans ces deux utilisations afin de me



projeter demain facilement dans une entreprise utilisant aussi bien une méthode en ligne de commande que le support logiciel graphique de Github.

Git est un logiciel de versions décentralisé. Le code informatique développé est stocké non seulement sur l'ordinateur de chaque contributeur du projet, mais il peut également l'être sur un serveur dédié. C'est un outil de bas niveau, qui se veut simple et performant, dont la principale tâche est de gérer l'évolution du contenu d'une arborescence.

Github est un service web d'hébergement et de gestion de développement de logiciels, utilisant le logiciel de gestion de versions Git.

⇒ VISUALSTUDIOCODE

Je travaille avec Visual Studio Code comme IDE (environnement de développement), c'est un ensemble d'outils qui permet d'augmenter la productivité des développeurs.

Cet éditeur de code inclut la prise en charge du débogage, la mise en évidence de la syntaxe, la complétion intelligente du code, la refactorisation du code et Git intégré ainsi que les possibilités d'y intégrer de nombreuses extensions améliorant le travail au quotidien. Il prend aussi en charge plusieurs centaines de langages de programmation, tels que C, C#,

C++, CSS, HTML, Java, JavaScript, React et son framework React-native, JSON, Markdown, PHP, Powershell, Python, TypeScript, YAML...,

Son interface d'édition intègre des raccourcis clavier, des sélections multiples, un enregistrement automatique du travail réalisé, une fonction rechercher/remplacer, le formatage du code source...Ce qui en fait à mon goût un outil complet pour le développement d'application tel que ce projet.



Visual Studio Code

b) Outils organisationnels

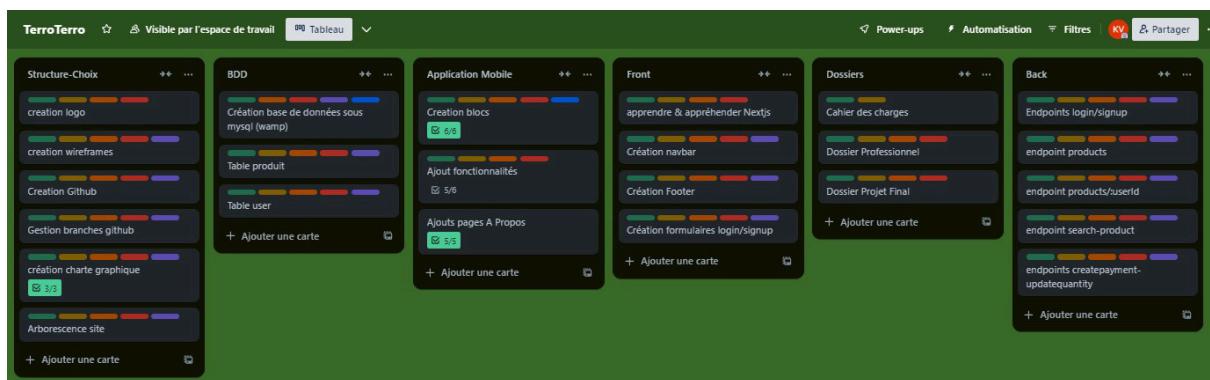
⇒ TRELLO

Trello est un outil de gestion de projet qui permet de structurer les tâches et de suivre l'avancement de celles-ci. Il peut être utilisé pour un travail de développeur seul en créant des listes pour chaque étape du projet, comme "à faire", "en cours" et "fait".

Pour utiliser Trello pour un projet de développement, vous pouvez créer une carte pour chaque tâche ou feature à implémenter. Ces cartes peuvent contenir des détails sur la tâche, des étiquettes pour indiquer l'importance ou la priorité, et des pièces jointes pour partager des fichiers ou des images.

Vous pouvez également utiliser des listes pour organiser les tâches selon leur état d'avancement, comme "à faire", "en cours" et "fait", et déplacer les cartes entre ces listes au fur et à mesure que vous progressez dans le projet.

Il est également possible d'utiliser des tableaux pour organiser les tâches selon leur catégorie ou selon leur impact sur le projet, pour une meilleure visualisation.

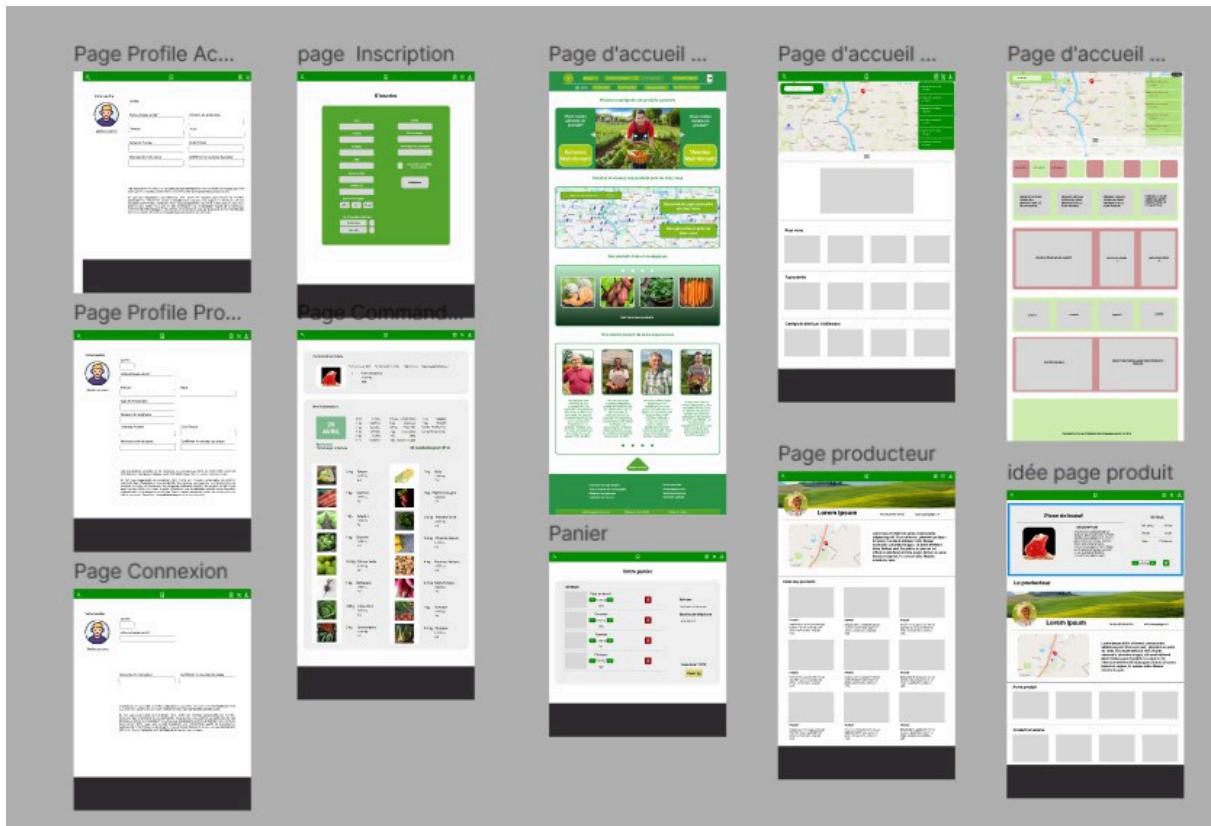


source : <https://trello.com/b/Vyx5ZHqi/terroterro>

2. Maquettage de l'application

Nous avons tout d'abord échangé sur le projet, à la manière la plus simple d'offrir aux futurs utilisateurs une expérience simple et ludique. Nous avons donc élaboré un wireframe, puis une première version de la maquette, nous avons ensuite échangé et apporté des modifications pour aboutir sur une V2 qui correspondait beaucoup plus à nos attentes, nous avons tous les 3 validé cette version.

Nous avons pu définir la charte graphique, la “font”, ainsi que les couleurs et le logo. Après validation de la maquette nous avons pu commencer la conception de l’application en version Web et en version Mobile.



3. Création de la Base de Données

Avant de créer la base de données, qu'est-ce qu'une base donnée ? Une base de données est une collection d'informations organisées afin d'être facilement consultables, gérables et mises à jour. Au sein d'une base donnée, les données sont organisées en lignes, colonnes et tableaux.

Elles sont indexées afin de pouvoir facilement trouver les informations recherchées. Chaque fois que de nouvelles informations sont ajoutées, les données sont mises à jour, et éventuellement supprimées. L'outil pour gérer la structure SQL utilisé ici est MySQL via l'interface phpMyAdmin. Cette interface simplifie grandement l'administration des bases de données MySQL grâce à son interface visuelle et ses nombreuses fonctionnalités, rendant accessible même aux utilisateurs moins expérimentés en SQL.

voici le **Modèle Conceptuel de Données (MCD)**:

USER	
id (PK)	
email (U)	
role	
mot_de_passe	
nom	
prenom	
adresse	
ville	
code_postal	
telephone	
date_creation	
raison_sociale	
+-----+	
	1
	v
0..*	
PRODUIT	
+-----+	
id (PK)	
nom	
categorie	
images	
prix	
quantite	
date_creation	
description	
user_id (FK)	
+-----+	

voici le **Modèle Logique de Données (MLD)**:

USER (

```
    id INT PRIMARY KEY AUTO_INCREMENT,  
    email VARCHAR(100) UNIQUE NOT NULL,  
    role ENUM('client', 'admin', 'vendeur') NOT NULL,  
    mot_de_passe VARCHAR(100) NOT NULL,  
    nom VARCHAR(100) NOT NULL,  
    prenom VARCHAR(100) NOT NULL,  
    adresse VARCHAR(255),  
    ville VARCHAR(100),  
    code_postal INT,  
    telephone VARCHAR(20),  
    date_creation DATETIME DEFAULT CURRENT_TIMESTAMP,  
    raison_sociale VARCHAR(255)
```

)

PRODUIT (

```
    id INT PRIMARY KEY AUTO_INCREMENT,  
    nom VARCHAR(255) NOT NULL,  
    categorie ENUM('fruits', 'poissons', 'legumes', 'produits_sucres', 'produits_laitiers',  
    'cereales', 'viandes') NOT NULL,  
    images MEDIUMBLOB,  
    prix DECIMAL(10,2) NOT NULL,  
    quantite DECIMAL(10,0),  
    date_creation DATETIME DEFAULT CURRENT_TIMESTAMP,  
    description LONGTEXT,  
    user_id INT,  
    FOREIGN KEY (user_id) REFERENCES USER(id) ON DELETE SET NULL ON  
    UPDATE CASCADE
```

)

voici le **Diagramme de classes (UML)**:



Table user:

	id	email	role	mot_de_passe	nom	prenom	adresse	ville	code_postal	telephone	date_creation	raison_sociale
<input type="checkbox"/>	Éditer Copier Supprimer	1 enzo@gmail.com	client	\$2b\$10\$0cYQhsrFCXHVfBPgO9uK10opCPCeTkybXWxRz2l...	Brison	enzo	allée des peupliers	Ambares	33440	06 69 38 33 47	2024-09-25 10:38:42	NULL
<input type="checkbox"/>	Éditer Copier Supprimer	5 jean@gmail.com	client	\$2b\$10\$sdRNExz@D4rCW3pxtKw0j5DaMuG3yikjkYDXdaJY...	fajid	fajid	bdx	33000	060606060608	2024-09-27 12:24:38	NULL	
<input type="checkbox"/>	Éditer Copier Supprimer	6 kak@gmail.com	client	\$2b\$10\$3uMMrSL8eyzAmoH7..UsWwzHDFPQ1OMvf6s4...	fsf	fsf	rue fjan	bdx	33000	060606060608	2024-09-27 12:29:14	NULL
<input type="checkbox"/>	Éditer Copier Supprimer	7 jean3@gmail.com	client	\$2b\$10\$ubKVLsUBMKvKYCxFa5D9KufUx2UzIUi6nZR2D00A...	jlsjkn	kevin	rue ad	bdx	33000	060606060608	2024-09-27 12:35:24	NULL
<input type="checkbox"/>	Éditer Copier Supprimer	8 kevin.villard@gmail.com	vendeur	\$2b\$10\$VtM0\$gTsp9oWcOice4oDzrogUT8tGJgfzXTJ...	veillard	kevin	80 rue paradis	Bordeaux	33000	060606060608	2024-09-27 14:08:56	dsgd
<input type="checkbox"/>	Éditer Copier Supprimer	9 jj@gmail.com	client	\$2b\$10\$ISUfIPaV9RrqdS2CzINZLeUp8kz8OHCiURrUsSY...	jacques	jean	100 avenue Goldmann	Bordeaux	33000	060606060608	2024-09-27 14:10:08	NULL
<input type="checkbox"/>	Éditer Copier Supprimer	10 patienceMan@gmail.com	client	\$2b\$10\$vtB0k1D4AMlow1bVO2WQDjuFoWTub5JXIs0HeHMU...	koribrama	patience	5 rue de la partie	Bordeaux	33000	060606060608	2024-10-01 09:52:01	NULL
<input type="checkbox"/>	Éditer Copier Supprimer	11 jsbach@gmail.com	vendeur	\$2b\$10\$6qVb1vPz13tzUCpPHOnZS6b0cpBq3roscl0YIV...	Bach	Jean-Sébastien	0 rue piano	Bordeaux	33000	060606060608	2024-10-01 13:40:58	Pomme de Terro
<input type="checkbox"/>	Éditer Copier Supprimer	12 fabluc@gmail.com	vendeur	\$2b\$10\$k0Tg5cE78TVAdPWE43Me00SPoJjdDE1QowfhZg...	Luchini	Fabrice	11 rue Racine	Bordeaux	33000	060606060608	2024-10-01 13:48:16	Ferme Aquacole de Bordeaux (fab)
<input type="checkbox"/>	Éditer Copier Supprimer	13 michPo@gmail.com	vendeur	\$2b\$10\$uPzusha0hsM8n14Xho5euX93REZbu0E14SHNbUS7z...	Polnareff	Michel	4 rue du paradis	Bordeaux	33000	060606060608	2024-10-01 13:54:02	AgnToutEnUn

Table produits:

	id	nom	categorie	images	prix	quantite	date_creation	description	user_id
<input type="checkbox"/>	Éditer Copier Supprimer	2 pommes vertes	fruits	[BLOB - 123, kio]	3.30	6	2024-09-27 20:51:43	pommes vertes alouette	8
<input type="checkbox"/>	Éditer Copier Supprimer	7 bananes jaunes	fruits	[BLOB - 37,7 kio]	1.20	55	2024-10-01 13:42:54	bananes jaunes de la cultivées dans la région Nouv...	11
<input type="checkbox"/>	Éditer Copier Supprimer	8 Saumon	poissons	[BLOB - 59,9 kio]	19.50	4	2024-10-01 13:51:52	Saumon Norvégiano-Bordelais a la pièce - Ferme Aqu...	12
<input type="checkbox"/>	Éditer Copier Supprimer	11 carotte	legumes	[BLOB - 3,3 Mio]	1.40	50	2024-10-01 13:57:31	carottes cultivées a sans pesticides	13
<input type="checkbox"/>	Éditer Copier Supprimer	12 Lait Entier	produits_laitiers	[BLOB - 100,0 kio]	5.00	15	2024-10-01 13:57:31	Lait entier direct producteur	8
<input type="checkbox"/>	Éditer Copier Supprimer	13 Sirop d'érable	produits_sucre	[BLOB - 60,5 kio]	15.00	50	2024-10-01 14:04:21	Sirop d'érable recueillies directement sur notre t...	13
<input type="checkbox"/>	Éditer Copier Supprimer	14 Miel	produits_sucre	[BLOB - 106,1 kio]	8.00	4	2024-10-01 14:04:21	Miel sans adjutants	13
<input type="checkbox"/>	Éditer Copier Supprimer	15 blé	cereales	[BLOB - 56,4 kio]	1.20	1000	2024-10-01 14:26:38	blé d'avoine	13
<input type="checkbox"/>	Éditer Copier Supprimer	16 Salade Laitue	legumes	[BLOB - 2,9 Mio]	3.00	5	2024-10-01 14:26:38	laitue provenant d'une serre bio-olimmatique	13
<input type="checkbox"/>	Éditer Copier Supprimer	17 Radis	legumes	[BLOB - 168,1 kio]	3.20	4	2024-10-01 14:27:44	bottes de radis blanc	13

4. Application Web avec NextJS

Next.js est un framework JavaScript open-source basé sur React, conçu pour créer des applications web modernes et performantes. Développé par Vercel, il offre une solution complète pour le développement front-end et back-end, combinant le meilleur de React avec

des fonctionnalités avancées. Next.js se distingue par son rendu hybride, permettant à la fois le rendu côté serveur (SSR) et la génération de sites statiques (SSG), ce qui améliore considérablement les performances et le référencement. Le framework propose un système de routage basé sur les fichiers, simplifiant la structure de l'application et facilitant la navigation. Il intègre également l'optimisation automatique des images, le fractionnement de code, et le support natif de TypeScript. Next.js offre une expérience de développement fluide avec le rechargement à chaud et une configuration minimale requise. Il prend en charge l'internationalisation, l'analyse de performance intégrée, et s'intègre parfaitement avec diverses API et bases de données. Grâce à sa communauté active et son écosystème riche, Next.js est devenu un choix populaire pour développer des applications web rapides, évolutives et optimisées pour le référencement, utilisé par de nombreuses entreprises de renom comme Netflix, Uber et Twitch.



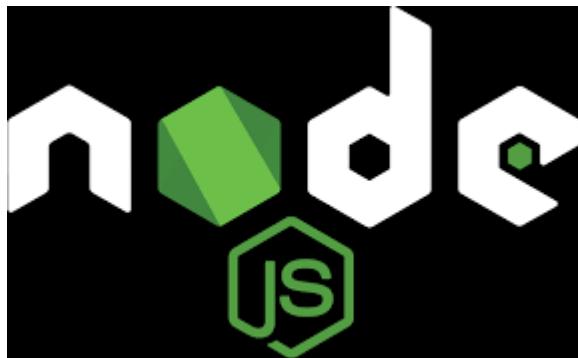
5. Backend avec Node.js et Express

Node.js est un environnement d'exécution JavaScript côté serveur qui permet d'exécuter du code JavaScript en dehors d'un navigateur web. Il offre une approche ouverte et orientée "événement" pour le développement d'applications scalables.

En association, Express est un framework web minimaliste pour Node.js qui simplifie le développement d'applications web et d'APIs. Il fournit une couche d'abstraction au-dessus de Node.js, offrant des fonctionnalités essentielles telles que le routage, en permettant une gestion facile des routes HTTP pour différentes URL et méthodes middleware (fonctions intermédiaires pour traiter les requêtes et les réponses).

Express intègre également une gestion des "vues" et des "erreurs"

L'association des deux permet de simplifier la création de serveurs web avec Node.js, réduisant le temps de développement, de structurer son application comme le développeur le souhaite, mais également de faciliter la création d'API RESTful robustes.



6. Application mobile avec React-Native

React Native est un framework open-source basé sur React.js qui permet de développer des applications mobiles multiplateformes. Il permet aux développeurs d'utiliser du code Javascript pour créer des applications natives pour IOS et Android, offrant ainsi une expérience utilisateur fluide et performante. En utilisant des composants pré-construits, React-native permet un développement rapide et efficace, avec la possibilité de réutiliser une grande partie du code entre les différentes plateformes. Il offre également un accès direct aux fonctionnalités natives des appareils, tels que l'appareil photo ou le GPS. Avec une large communauté de développeur et un support continu, React-native est une option populaire pour créer des applications mobile natives, offrant une productivité élevée et des résultats de haute qualité, cela m'a donc paru être un bon choix pour la partie mobile de notre application.



```

import { StatusBar } from "expo-status-bar";
import { StyleSheet, View, Text, ScrollView } from "react-native";
import Colors from "./constants/Colors";
import MyTabs from "./components/MyTabs";
import CustomHeader from "./components/CustomHeader";
import {BottomSheetModal,BottomSheetModalProvider} from "@gorhom/bottom-sheet";
import { GestureHandlerRootView } from "react-native-gesture-handler";
import React, { useRef, useMemo, useState } from "react";
import Profil from "./screens/ProfilScreen";

export default function MainApp() {
  const bottomSheetModalRef = useRef(null);
  const openModal = () => {bottomSheetModalRef.current?.present();};
  const snapPoints = useMemo(() => ["95%"], []);
  return (
    <GestureHandlerRootView>
      <BottomSheetModalProvider>
        <StatusBar style="auto" />
        <View style={styles.container}>
          <CustomHeader openModal={openModal}></CustomHeader>
          <BottomSheetModal
            ref={bottomSheetModalRef}
            snapPoints={snapPoints}
            index={0}
            backgroundStyle={{ borderRadius: 50 }}
          >
            <View style={styles.modalContainer}>
              <Profil></Profil>
            </View>
          </BottomSheetModal>
          <MyTabs />
        </View>
      </BottomSheetModalProvider>
    </GestureHandlerRootView>
  );
}

```

7. Les Composants

Les composants de React sont les blocs de construction fondamentaux d'une application React. Ils sont utilisés pour créer des interfaces utilisateur réutilisables et modulaires. Voici quelques composants les plus couramment utilisés :

- “Functional Components” : ce sont des fonctions Javascript qui renvoient du JSX pour décrire l'interface utilisateur, ils sont simples à écrire, légers et recommandés pour les composants sans état.

- “Props” : Les composants reçoivent des données appelées ‘props’ qui leur permettent de recevoir des valeurs externes. Les props sont utilisés pour configurer et personnaliser les composants.

- “State” : Les “class components” peuvent avoir leur propre état interne, géré par la propriété ‘state’. Le “state” permet aux composants de réagir aux interactions de l’utilisateur et de mettre à jour dynamiquement leur interface utilisateur en conséquence. En utilisant ces composants et les principes de la programmation réactive, les développeurs peuvent créer des interfaces utilisateur dynamiques, flexibles et réutilisables dans leurs applications React.

Exemple sur les props :

En React Native, les "props" (abréviation de "properties") sont utilisées pour passer des données de composant parent à un composant enfant. Voici un exemple simple d'utilisation des props en React-Native :

Composant Parent :

```
```javascript
import React from 'react';
import { View } from 'react-native';
import ChildComponent from './ChildComponent';

const ParentComponent = () => {
 const name = "John Doe";
 const age = 25;
 return (
 <View>
 <ChildComponent name={name} age={age} />
 </View>
);
};
```

#### Composant Enfant :

```
```javascript
import React from 'react';
import { Text } from 'react-native';
const ChildComponent = (props) => {
  return (
    <Text>
      My name is {props.name} and I'm
      {props.age} years old.
    </Text>
  );
};

export default ChildComponent;
```
```

Dans cet exemple, le composant “ParentComponent” passe les “props” `name` et `age` à “ChildComponent” en les spécifiant en tant qu’attributs sur la balise `<ChildComponent />`. Dans “ChildComponent”, les props sont reçues en tant que paramètre de la fonction et peuvent être utilisées pour afficher les données dans le composant enfant. Ainsi, le

composant "ChildComponent" affichera le message : "My name is John Doe and I'm 25 years old." en utilisant les valeurs passées via les props.

Nous avons un composant qui a été créé pour le Header (appelé CustomHeader.js). Grâce au côté modulaire de ce langage, je peux appeler ce "CustomHeader" spécifique dans le fichier souhaité (qui sera son composant parent). Dans notre cas, il s'agit du fichier MainApp.js (fichier à l'origine du rendu global de l'affichage)

L'appel de ce composant spécifique, s'effectue dans le fichier parent (MainApp) comme ceci:  
`<CustomHeader />`

Résultat : notre "CustomHeader" s'affiche et peut être facilement interchangeable et modifiable, cela permet au développeur d'être encore plus réactif aux demandes d'un potentiel client.

```

import { StyleSheet, Text, View, SafeAreaView, TextInput, TouchableOpacity, Animated } from "react-native"
import React, { useState, useEffect, useRef } from "react";
import { Image } from "react-native";
import Colors from "../constants/Colors";
import { Ionicons } from "@expo/vector-icons";
import { Keyboard } from "react-native";

const CustomHeader = ({ openModal }) => {
 const [isSearchActive, setIsSearchActive] = useState(false);
 const animatedHeight = useRef(new Animated.Value(60)).current;

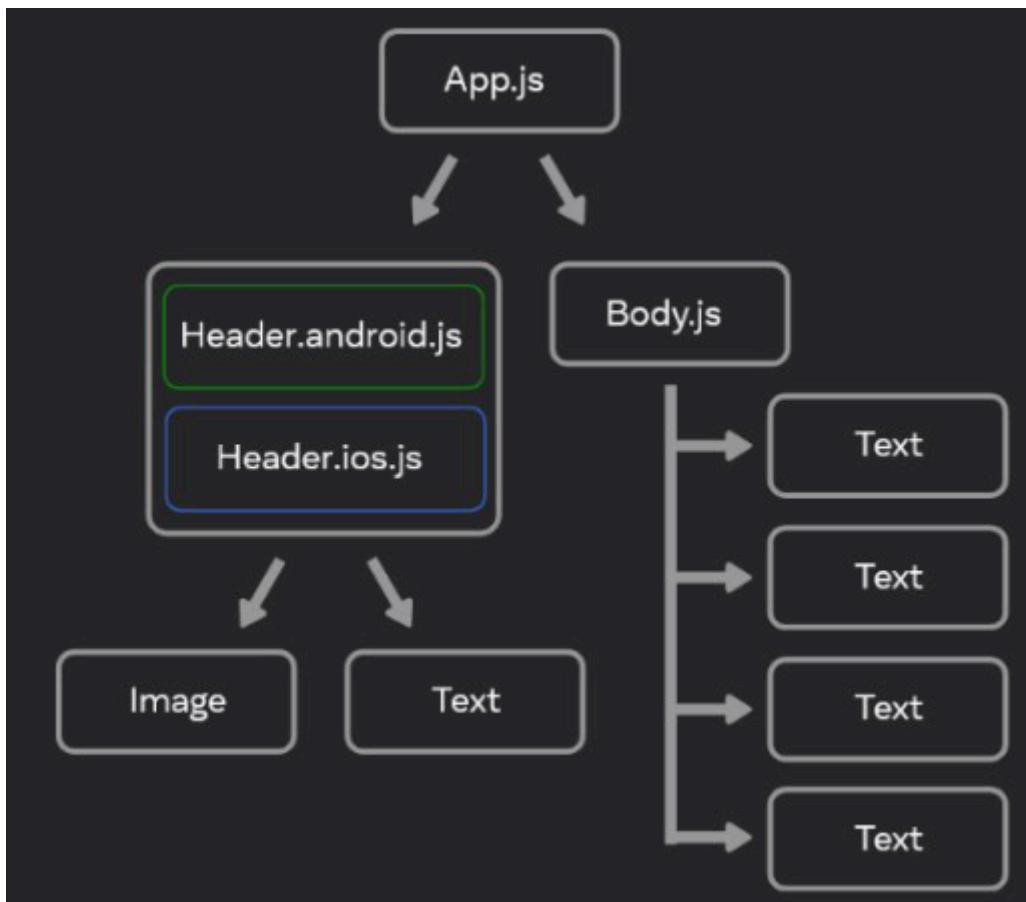
 useEffect(() => {
 const keyboardDidHideListener = Keyboard.addListener('keyboardDidHide', () => {
 setIsSearchActive(false);
 });
 return () => {
 keyboardDidHideListener.remove();
 };
 }, []);

 useEffect(() => {
 Animated.timing(animatedHeight, {
 toValue: isSearchActive ? 110 : 60,
 duration: 300,
 useNativeDriver: false,
 }).start();
 }, [isSearchActive]);
 const toggleSearch = () => {
 setIsSearchActive(!isSearchActive);
 };

 return (
 <SafeAreaView style={styles.safeArea}>
 <Animated.View style={[styles.container, { height: animatedHeight }]}>
 <View style={styles.topRow}>
 <Image
 source={require("../assets/images/logo.jpg")}
 style={styles.logo}
 />
 <Text style={styles.title}>TerroTerro</Text>
 <TouchableOpacity onPress={toggleSearch} style={styles.searchButton}>
 <Ionicons name="search-outline" size={24} color={Colors.white} />
 </TouchableOpacity>
 <TouchableOpacity
 style={styles.profileIcon}
 onPress={openModal}
 accessible={true}
 accessibilityLabel="Bouton d'accessibilité"
 accessibilityHint="Appuyez pour accéder à votre page profil"
 accessibilityRole="button"
 >
 <Ionicons name="person" size={24} color={Colors.white} />
 </TouchableOpacity>
 </View>
 {isSearchActive && (
 <View style={styles.searchContainer}>
 <Ionicons name="search-outline" size={20} color={Colors.danger} style={styles.searchIcon} />
 <TextInput
 style={styles.input}
 placeholderTextColor={Colors.black}
 placeholder="Agriculteurs, Producteurs, ..."
 autoFocus
 />
 </View>
)}
 </Animated.View>
 </SafeAreaView>
);
};

export default CustomHeader;

```



Source: <https://makersden.io/blog/why-your-business-should-consider-react-native-for-app-development>

The screenshot shows a portion of a React Native code file. The code includes components like `<GestureHandlerRootView>`, `<BottomSheetModalProvider>`, `<StatusBar style="auto" />`, `<View style={styles.container}>`, `<CustomHeader openModal={openModal}/>`, `<BottomSheetModal ref={bottomSheetModalRef} snapPoints={snapPoints} index={0} backgroundStyle={{ borderRadius: 50 }}>`, `<View style={styles.modalContainer}>`, `<Profil></Profil>`, and `<MyTabs />`. A green callout box on the right highlights the `<CustomHeader>` component with the text "Composant <CustomHeader/> appelé ici".

```

<GestureHandlerRootView>
 <BottomSheetModalProvider>
 <StatusBar style="auto" />
 <View style={styles.container}>
 <CustomHeader openModal={openModal}/>
 <BottomSheetModal
 ref={bottomSheetModalRef}
 snapPoints={snapPoints}
 index={0}
 backgroundStyle={{ borderRadius: 50 }}
 >
 <View style={styles.modalContainer}>
 <Profil></Profil>
 </View>
 </BottomSheetModal>
 <MyTabs />
 </View>
 </BottomSheetModalProvider>
</GestureHandlerRootView>

```

## 8. Création de formulaires

Les formulaires en React et React-native sont des éléments clés pour l'interaction utilisateur dans les applications. En React, les formulaires sont gérés en utilisant l'état 'state' pour suivre et mettre à jour les valeurs des champs de saisie à mesure que l'utilisateur interagit. Les événements tels que onChange sont utilisés pour détecter les modifications et mettre à jour l'état.

En React-native, des composants spécifiques tels que TextInput et TouchableOpacity sont utilisés pour créer des champs de saisie et des boutons respectivement. Les méthodes de gestion des événements, comme onPress, permettent de détecter les actions de l'utilisateur. Pour la validation des données, des bibliothèques telles que Formik ou React Hook Form sont disponibles, offrant des fonctionnalités avancées comme la validation des champs et la gestion des erreurs. En résumé, les formulaires en React et React-native sont essentiels pour collecter des données et offrir une interaction utilisateur fluide, en utilisant des composants spécialisés et des techniques de gestion de l'état et des événements.

```

import React, { useState } from "react";
import {
 View,
 TextInput,
 TouchableOpacity,
 Text,
 StyleSheet,
 ActivityIndicator,
} from "react-native";
import { useAuth } from "../context/AuthContext";
import Colors from "../constants/Colors";
import { signIn } from "../api/api";
import tinyColor from "tinyColor";
import { API_URL } from "../config";

const LoginScreen = ({ navigation }) => {
 const [email, setEmail] = useState("");
 const [password, setPassword] = useState("");
 const { signInContext } = useAuth();
 const [error, setError] = useState(null);
 const [isLoading, setIsLoading] = useState(false);
 const validateForm = () => {
 if (!email || !password) {
 setError("Veuillez remplir tous les champs");
 return false;
 }
 if (!/^\S+@\S+\.\S+$/.test(email)) {
 setError("Veuillez entrer une adresse email valide");
 return false;
 }
 setError(null);
 return true;
 };
 const handleLogin = async () => {
 if (!validateForm()) return;

 setIsLoading(true);
 try {
 console.log("Début de la tentative de connexion");
 const response = await signIn(email, password);
 console.log("Réponse reçue du serveur:", response);
 const { token, user } = response;
 console.log("Avant mise à jour du contexte", response);
 // Utilisez la fonction signIn du contexte pour mettre à jour l'état d'authentification
 await signInContext(token, user);
 console.log("Après mise à jour du contexte");
 // La navigation sera gérée par le navigateur principal basé sur l'état d'authentification
 // Vous pouvez supprimer cette ligne si vous utilisez un navigateur conditionnel
 // navigation.navigate("NavigationTest", { user: user.name });
 } catch (error) {
 console.error("Erreur détaillée:", error);
 setError(error.message || "Identifiants invalides");
 } finally {
 setIsLoading(false);
 }
 };
}

```

Sur la cette première image, on peut voir les constantes qui prennent le “useState”, la constant login qui vérifie le rôle (via le « user ») et le token afin de déterminer si un producteur (vendeur) ou un acheteur est connecté, car les formulaires qui en résultent ne sont pas les mêmes. Sur la seconde photo, la partie “front-end” du formulaire, avec des composants Text-input, associés à leurs “placeholder” (qui est un texte d'exemple ou une courte phrase apparaissant dans un champ de saisie lorsque ce champ est vide. Ce texte

s'efface automatiquement lorsque l'utilisateur commence à saisir des données). L'attribut value est utilisé pour spécifier la valeur par défaut d'un champ de formulaire. Cette valeur est ce qui sera envoyé au serveur lorsque le formulaire sera soumis, en association avec le nom du champ. Pour les champs de type texte (<input type="text">), value définit le texte affiché dans le champ au moment du chargement du formulaire. L'utilisateur peut modifier cette valeur avant de soumettre le formulaire.

```
return (
 <View style={styles.container}>
 <TextInput
 style={styles.input}
 placeholder="Email"
 value={email}
 onChangeText={setEmail}
 keyboardType="email-address"
 autoCapitalize="none"
 placeholderTextColor={Colors.gray}
 />
 <TextInput
 style={styles.input}
 placeholder="Mot de passe"
 value={password}
 onChangeText={setPassword}
 secureTextEntry
 placeholderTextColor={Colors.gray}
 />
 {error && <Text style={styles.errorText}>{error}</Text>}
 <TouchableOpacity
 style={styles.SignLogButton}
 onPress={handleLogin}
 disabled={isLoading}
 >
 {isLoading ? (
 <ActivityIndicator color={Colors.white} />
) : (
 <Text style={styles.buttonText}>Se connecter</Text>
)}
 </TouchableOpacity>
 <TouchableOpacity style={styles.SignLogButton} onPress={() => navigation.navigate("signup")}>
 <Text style={styles.buttonText}>S'inscrire</Text>
 </TouchableOpacity>
 <TouchableOpacity style={styles.passwordLink} onPress={() => navigation.navigate("ForgotPassword")}>
 <Text style={styles.forgot}>Forgot Password?</Text>
 </TouchableOpacity>
 </View>
);
};
```

Pour les boutons radio et les cases à cocher, value définit la valeur qui sera envoyée si l'élément est sélectionné.

Pour un bouton de soumission (<input type="submit">), value détermine le texte affiché sur le bouton. Également, "onChangeText" est une fonction qui est appelée chaque fois que le texte du champ de saisie change. La nouvelle chaîne de texte est passée en argument à cette fonction. Cette propriété permet de mettre à jour l'état d'un composant en fonction de la saisie de l'utilisateur, ce qui est essentiel pour gérer des formulaires et des entrées utilisateur.

## 9. Sécurité (login, rôles, authContext)

La sécurité est un aspect essentiel dans le développement d'applications, y compris en React-native. Pour gérer la sécurité des utilisateurs, plusieurs concepts clés sont utilisés :

### a) Login:

Le processus d'authentification où les utilisateurs fournissent leurs informations d'identification (ex : le mail et le mot de passe) pour accéder à l'application. Cela permet de vérifier l'identité des utilisateurs et de restreindre l'accès aux ressources protégées.

**Création d'un endpoint login ('/signin'):**

```
app.post('/signin', (req, res) => {
 const { email, password } = req.body;
 console.log('Tentative de connexion - Email reçu:', email);
 console.log('Tentative de connexion - Mot de passe reçu:', password ? '[PRÉSENT]' : '[MANQUANT]');
 if (!email || !password) {
 console.log('Erreur: Email ou mot de passe manquant');
 return res.status(400).json({ error: 'Email et mot de passe requis' });
 }

 pool.query(
 'SELECT * FROM user WHERE email = ?',
 [email],
 (err, rows) => {
 if (err) {
 console.error('Erreur durant la requête SQL:', err);
 return res.status(500).json({ error: 'Erreur interne du serveur' });
 }

 console.log('Nombre d\'utilisateurs trouvés:', rows.length);
 if (rows.length === 0) {
 console.log('Aucun utilisateur trouvé pour cet email');
 return res.status(401).json({ error: 'Email ou mot de passe incorrect' });
 }

 const user = rows[0];
```

```

 console.log('Utilisateur trouvé:', { id: user.id, email: user.email });

 console.log('Mot de passe haché stocké:', user.mot_de_passe
? '[PRÉSENT]' : '[MANQUANT]');

 if (!user.mot_de_passe) {
 console.error('Mot de passe haché manquant pour l\'utilisateur:', email);
 return res.status(500).json({ error: 'Erreur interne du serveur' });
 }

 bcrypt.compare(password, user.mot_de_passe, (err, validPassword) => {
 if (err) {
 console.error('Erreur durant la comparaison du mot de passe:', err);
 return res.status(500).json({ error: 'Erreur interne du serveur' });
 }

 console.log('Résultat de la comparaison du mot de passe:', validPassword);

 if (!validPassword) {
 console.log('Mot de passe incorrect pour l\'utilisateur:', email);
 return res.status(401).json({ error: 'Email ou mot de passe incorrect' });
 }

 const token = jwt.sign({ userId: user.id },
 'votre_secret_jwt', { expiresIn: '1h' });
 console.log('Connexion réussie pour l\'utilisateur:', email);

 res.json({ token,
 user: {
 id: user.id,
 firstName: user.prenom,
 lastName: user.nom,
 email: user.email,
 address: user.adresse,
 telephone: user.telephone,

```

```

 postalCode: user.code_postal,
 ville: user.ville,
 role:user.role,
 raisonSociale:user.raisonSociale
 }
}
);
}
)
;
}
);

```

Dans ce endpoint d'authentification pour la connexion de l'utilisateur, l'utilisation de JWT (JSON Web Tokens) et bcrypt joue un rôle crucial pour la sécurité et la gestion des sessions. **JWT** est une méthode compacte et autonome pour transmettre de manière sécurisée des informations entre les parties sous forme d'objet JSON. Il facilite la mise à l'échelle des applications, signe et chiffre les informations sans surcharger le serveur (pas stockées) et est cross domaine.

**Bcrypt** est une fonction de hachage sécurisé des mots de passe notamment.

Elle est résistante aux attaques par force brute grâce à sa lenteur délibérée, permet une adaptabilité à l'évolution sécuritaire, et possède un salt intégré (chaque hash a son propre "salt", ce qui simplifie le stockage et la vérification).

### **API signin:**

```

export const signIn = async (email, password) => {
 try {
 const response = await fetch(` ${API_URL}/signin`, {
 method: "POST",
 headers: {
 "Content-Type": "application/json",
 },
 body: JSON.stringify({ email, password }),
 });
 const data= await response.json();
 if (!response.ok) {
 throw new Error(data.error || 'Erreur de connexion');
 }
 }
}

```

```

 console.log('Données reçues de l\'API:', data);
 return data;
 }

} catch (error) {
 console.error("Erreur lors de la connexion (api):", error);
 throw error;
}
};


```

## b) Rôles:

Les rôles sont utilisés pour définir les niveaux d'accès des utilisateurs. Ils permettent de restreindre certaines fonctionnalités ou ressources en fonction des permissions accordées à chaque rôle. Par exemple, un candidat n'aura pas les mêmes accès aux données qu'une entreprise.

```

app.post('/signup', async (req, res) => {
 try {
 const { firstName, lastName, email, password, telephone, address,
ville, postalCode, role, raisonSociale } = req.body;

 // Log des données reçues (attention à ne pas logger les mots de
passe en production)
 console.log('Received signup request:', { firstName, lastName,
email, telephone, address, ville, postalCode, role, raisonSociale });

 // Vérification que tous les champs nécessaires sont présents
 if (!firstName || !lastName || !email || !password || !telephone
|| !address || !ville || !postalCode || !role) {
 return res.status(400).json({ error: 'Tous les champs sont
requis' });
 }
 if (role === 'vendeur' && (!raisonSociale)) {
 return res.status(400).json({ error: 'Raison sociale et image
sont requis pour les vendeurs' });
 }
 }
}


```

```

<Tab.Navigator
 screenOptions={{
 tabBarActiveTintColor: "darkgreen",

```

```

 tabBarInactiveTintColor: Colors.white,
 tabBarStyle: { backgroundColor: Colors.greenAgri },
 tabBarHideOnKeyboard: true,
 headerShown: false,
 } }
>
<Tab.Screen
 name="Accueil"
 component={HomeStack}
 options={ {
 tabBarLabel: "Accueil",
 tabBarIcon: ({ color }) => (
 <Ionicons name="home" size={20} color={color} />
) ,
 } }
/>
{user?.role === "vendeur" && (
 <Tab.Screen
 name="ProduitManagement"
 component={ProductManagementStack}
 options={ {
 tabBarLabel: "Gestion produitss",
 tabBarIcon: ({ color }) => (
 <Ionicons name="nutrition" size={20} color={color} />
) ,
 } }
/>
) }

```

```
const HomeStack = () => (
```

```

const ProductManagementStack = () => (
 <Stack.Navigator screenOptions={screenOptions}>
 <Stack.Screen
 name="ProductManagementScreen"
 component={ProductManagementScreen}
 options={ { headerShown: false } }
 />
 </Stack.Navigator>
);

```

### c) AuthContext:

Le contexte d'authentification est un mécanisme fourni par React-native pour gérer l'état de connexion et l'authentification des utilisateurs de manière globale dans l'application. Il permet de stocker les informations d'authentification telles que le token et de les rendre accessibles à tous les composants de l'application.

En utilisant ces concepts, on peut mettre en place un système de sécurité robuste dans une application React-native. Le processus de login authentifie les utilisateurs, les rôles permettent de définir les niveaux d'accès appropriés et le contexte d'authentification facilite la gestion de l'état de connexion à travers l'application. Cela contribue à assurer la confidentialité et la sécurité des données des utilisateurs et à offrir une expérience utilisateur sécurisée.

**Dans dossier context, le fichier AuthContext.js:**

```
export const AuthProvider = ({ children }) => {
```

```
const signInContext = async (authToken, userData) => {
 try {
 console.log("Token reçu:", authToken);
 console.log("Données utilisateur reçues:", userData);
 await AsyncStorage.setItem("userToken", authToken);
 await AsyncStorage.setItem("userData", JSON.stringify(userData));
 setUserToken(authToken);
 setUserData(userData);
 } catch (e) {
 console.error(
 "Erreur lors de la sauvegarde des données de connexion:",
 e
);
 }
};

return (
 <Provider value={signInContext}>
 {children}
 </Provider>
);
```

```

<AuthContext.Provider
 value={ {
 isLoading,
 userToken,
 user,
 error,
 signInContext,
 signOut,
 signUpUser,
 clearError,
 useProducts
 } }
>
 {children}
</AuthContext.Provider>
);
};

export const useAuth = () => useContext(AuthContext);

```

```

export default function App() {
 return (
 <AuthProvider>
 <StripeProvider
 publishableKey="pk_test_51PfNmyRpKgZkfjqiZU6RXcDkGN4tjVTxY5TA9twzE48MEUMQe8fQojaXd7wJWUaSbRg2jgHmprVUWBGvQ8v8b41K00NeOYreRk"
 // merchantIdentifier="merchant.com.terroterro.app" // Pour Apple Pay
 >
 <AppContent />
 </StripeProvider>
 </AuthProvider>
);
}

```

## 10. Les requêtes http avec Fetch

Les requêtes http avec Fetch en React-native permettent de communiquer avec des serveurs et d'échanger des données. Fetch est une fonction intégrée dans Javascript qui facilite l'envoie de requêtes http et la gestion des réponses.

Voici un résumé des principales étapes pour effectuer une requête HTTP avec Fetch en React-native :

- a) On importe la fonction Fetch depuis la bibliothèque native de React-native.
- b) On utilise la méthode Fetch pour envoyer une requête HTTP vers une URL spécifique.
- c) On spécifie la méthode HTTP appropriée (GET, POST, PUT, DELETE, etc.) et les en-têtes nécessaires.
- d) On gère la réponse de la requête en utilisant les méthodes de la promesse renvoyée par Fetch, telles que .then() et .catch().
- e) On transforme la réponse en JSON en utilisant la méthode .json() pour faciliter la manipulation des données.
- f) On traite les données de la réponse et on effectue les actions nécessaires dans notre application.

```
export const signUp = async (userData) => {
 try {
 const response = await fetch(`${API_URL}/signup`, {
 method: "POST",
 headers: {
 "Content-Type": "application/json",
 },
 body: JSON.stringify(userData),
 });

 if (!response.ok) {
 const errorData = await response.json();
 throw new Error(errorData.error || "Erreur lors de l'inscription");
 }

 return await response.json();
 } catch (error) {
 console.error("Erreur lors de l'inscription:", error);
 throw error;
 }
};
```

⇒ 1ère étape: Déclaration de la fonction :

La fonction est déclarée comme asynchrone, ce qui permet d'utiliser await à l'intérieur de la fonction. Elle prend un paramètre userData, qui est un objet contenant les

données nécessaires pour l'inscription d'un utilisateur (comme le nom, l'email, le mot de passe, etc.). Le bloc try est utilisé pour encapsuler le code qui pourrait générer une erreur. Si une erreur se produit dans ce bloc, elle sera capturée par le bloc catch.

## ⇒ 2ème étape: Appel à l'API :

```
const response = await fetch(` ${API_URL}/signup`, {
 method: "POST",
 headers: {
 "Content-Type": "application/json",
 },
 body: JSON.stringify(userData),
});
```

**const response = await fetch()** : Cette ligne effectue une requête HTTP en utilisant la méthode fetch. Le mot-clé await indique que la fonction doit attendre que la promesse retournée par fetch soit résolue avant de continuer.

**` \${API\_URL}/signup`** : Cela construit l'URL de l'API à laquelle la requête est envoyée. API\_URL est une constante définie dans un fichier config.js, représentant l'URL de base du serveur backend.

**method: "POST"** : Cela spécifie que la requête HTTP doit utiliser la méthode POST, qui est ici utilisée pour envoyer des données au serveur.

**headers: { "Content-Type": "application/json" }** : Cela définit les en-têtes HTTP de la requête. Ici, il indique que le corps de la requête contiendra des données au format JSON.

**body: JSON.stringify(userData)** : Cela convertit l'objet userData en une chaîne JSON afin qu'il puisse être envoyé dans le corps de la requête POST.

## ⇒ 3ème étape: Gestion des erreurs:

**if (!response.ok)** : Cette condition vérifie si la réponse du serveur indique un succès. response.ok est vrai si le code de statut HTTP est dans la plage 200-299. Ici on vérifie si la réponse n'est pas vraie, et cela signifie qu'il y a eu une erreur lors de la tentative d'inscription.

**const errorData = await response.json()** : Si la réponse n'est pas OK, cette ligne tente d'extraire les données d'erreur envoyées par le serveur au format JSON.

**throw new Error**: Cela lance une nouvelle erreur avec un message approprié. Si le serveur a renvoyé un message d'erreur, il sera utilisé ; sinon, un message générique "Erreur lors de l'inscription" sera utilisé.

**catch (error) :** attrape toute erreur qui pourrait survenir dans le bloc try.

**console.error("Erreur lors de l'inscription:", error) :** Cela enregistre l'erreur dans la console pour faciliter le débogage. Il affiche un message indiquant qu'une erreur s'est produite lors de l'inscription, suivi des détails de l'erreur.

**throw error :** Cela relance l'erreur après l'avoir enregistrée, permettant à d'autres parties du code qui appellent cette fonction de gérer également cette erreur si nécessaire.

## 11. Les Tests

Le Test Driven Development (TDD) est une approche de développement logiciel dans laquelle les tests sont écrits avant le code de production. Cela encourage un cycle de développement itératif où les tests guident la conception et la mise en œuvre du code.

Voici un exemple de la façon dont le TDD peut être utilisé :

Supposons que nous devons implémenter une fonction pour calculer la somme de deux nombres entiers. En utilisant TDD, nous commencerons par écrire un test qui vérifie si la fonction retourne la somme correcte pour un cas donné :

```
```java
import static org.junit.Assert.assertEquals;
import org.junit.Test;
public class SumTest {
    @Test
    public void testSum() {
        int result = sum(2, 3);
        assertEquals(5, result);
    }
    public int sum(int a, int b) {
        return a + b;
    }
}
````
```

Dans cet exemple, nous écrivons d'abord le test `testSum()` qui appelle la fonction `sum()` avec les paramètres 2 et 3, puis vérifie si le résultat est égal à 5 en utilisant l'assertion `assertEquals()`. Cependant, la fonction `sum()` n'est pas encore implémentée.

Maintenant, si nous exécutons ce test, il échouera car la fonction `sum()` n'est pas encore définie. Cela nous motive à écrire le code de production pour satisfaire le test :

```
```java
public int sum(int a, int b) {
    return a + b;
}
```

Une fois le code de production écrit, nous exécutons à nouveau le test. S'il réussit, cela signifie que notre fonction est correcte. Sinon, nous apportons les modifications nécessaires jusqu'à ce que le test passe.

Ce cycle itératif de rédaction de tests, implémentation du code de production et vérification des tests nous permet d'améliorer la qualité de notre code, de détecter les erreurs plus tôt et de faciliter la maintenance à long terme.

En résumé, le Test Driven Development (TDD) est une approche de développement qui consiste à écrire les tests avant le code de production. Cela favorise un cycle de développement itératif où les tests guident la conception et l'implémentation du code. Le TDD permet d'améliorer la qualité du code et de faciliter la maintenance en détectant les erreurs

plus tôt dans le processus de développement.

VI. Perspectives Futures

Pour améliorer la sécurité de l'authentification de notre application, plusieurs mesures seront mises en place. Tout d'abord, nous utiliserons des variables d'environnement pour stocker le secret **JWT**. Cela implique la création d'un fichier **.env** à la racine du projet, où nous insérerons notre code secret. Nous installerons ensuite la bibliothèque **dotenv** pour charger ces variables dans notre fichier serveur, garantissant ainsi que les informations sensibles ne sont pas codées en dur dans le code source.

Nous allons également implémenter une limitation des tentatives de connexion en utilisant **express-rate-limit**. Cette fonctionnalité permettra de restreindre le nombre de tentatives de connexion à cinq par période de quinze minutes, ce qui contribuera à prévenir les attaques par force brute. En parallèle, nous ajouterons une validation plus stricte des entrées utilisateur grâce à la bibliothèque **joi**, afin de garantir que les données fournies lors de la connexion respectent des critères spécifiques.

Pour faciliter l'interaction avec notre base de données et améliorer la cohérence du code, nous considérerons l'**utilisation d'un ORM**, en particulier **Sequelize**. En installant Sequelize et mysql2, nous pourrons gérer nos modèles de données de manière plus efficace, tout en nous permettant d'implémenter facilement des transactions complexes.

En termes de perspectives futures pour notre application, plusieurs améliorations sont envisagées. Nous prévoyons d'intégrer une **authentification à deux facteurs (2FA)** pour renforcer la sécurité des connexions. De plus, un système de gestion des sessions sera développé pour permettre aux utilisateurs de gérer leurs connexions sur différents appareils. La mise en place d'un audit et d'une journalisation détaillée des actions critiques liées à l'authentification sera également envisagée.

Un système de gestion des rôles et des permissions sera également développé pour un meilleur contrôle d'accès. Enfin, nous établirons un processus régulier de mise à jour des dépendances et intégrerons des **tests de sécurité automatisés** dans notre **pipeline CI/CD** afin de détecter les vulnérabilités avant le déploiement. Ces améliorations visent à renforcer la sécurité et la robustesse de notre application tout en offrant une expérience utilisateur optimisée.

Le déploiement représente une étape cruciale qui mérite une attention particulière. Notre objectif est de mettre en place une stratégie de déploiement robuste et efficace, garantissant une transition fluide de l'environnement de développement vers la production.

Nous envisageons d'adopter une approche de déploiement continu (CD), en synergie avec notre pipeline d'intégration continue (CI) existant. Cette méthode nous permettra d'automatiser le processus de déploiement, réduisant ainsi les erreurs humaines et accélérant la mise en production de nouvelles fonctionnalités. Pour ce faire, nous prévoyons d'utiliser des outils tels que Jenkins, GitLab CI/CD ou GitHub Actions, qui s'intégreront parfaitement à notre flux de travail actuel.

Le déploiement sera structuré en plusieurs environnements distincts : développement, test, staging et production. Cette approche permettra de tester rigoureusement chaque nouvelle version de l'application dans des conditions proches de la production avant son lancement officiel. Nous mettrons en place des tests automatisés à chaque étape, incluant des tests de charge et de performance, pour garantir la stabilité et la fiabilité de l'application.

En termes d'infrastructure, nous explorerons les options de déploiement sur le cloud, en considérant des fournisseurs tels qu'AWS, Google Cloud Platform ou Microsoft Azure. L'utilisation de conteneurs Docker sera également envisagée pour assurer la cohérence entre les environnements de développement et de production, facilitant ainsi le déploiement et la scalabilité de l'application.

La gestion des configurations sera un aspect crucial de notre stratégie de déploiement. Nous utiliserons des outils de gestion de configuration comme Ansible ou Terraform pour automatiser la configuration de nos serveurs et services, garantissant ainsi une cohérence entre les différents environnements.

Un plan de rollback sera également élaboré pour permettre un retour rapide à une version stable en cas de problèmes imprévus lors du déploiement. Cela inclura la mise en place de sauvegardes régulières et de points de restauration.

Enfin, nous implémenterons un système de monitoring et d'alerting robuste, utilisant des outils comme Prometheus et Grafana, pour surveiller en temps réel les performances de l'application et détecter rapidement tout problème potentiel après le déploiement.

Cette approche globale du déploiement vise à assurer une mise en production fluide et sécurisée de notre application, tout en permettant une évolution continue et maîtrisée du produit. Elle s'inscrit dans notre démarche d'amélioration constante de la qualité et de la fiabilité de notre service.

VII. Conclusion

Ce projet m'a permis d'en apprendre beaucoup sur le métier de développeur d'applications, que ce soit d'un point de vue structurel, d'un point de vue organisationnel (en groupe), et bien sûr technique. Il m'a permis de me confronter à de nouveaux langages (nextjs, mais surtout react-native et nodejs pour le back-end). J'aurais pu choisir de réaliser mon projet avec un framework plus abordé en cours (tel que Symfony), il m'apparaissait important d'être dans cette optique d'apprentissage, et de 'mise en danger' représenté par la réalisation de ce projet dans des langages qui me semblaient de prime abord difficiles à appréhender. J'ai ainsi l'impression d'avoir énormément appris avec la réalisation de ce projet et je suis fier d'avoir relevé ce défi.

Concernant le projet en lui-même, la réalité rappelle parfois les ambitions de départ. Nous pensions pouvoir réaliser l'ensemble des fonctionnalités prévues en amont, mais certains impondérables personnels pour les uns, ainsi que le temps de réalisation de certaines tâches pour l'ensemble des collaborateurs nous a rappelé au pragmatisme. Cependant, nous avons énormément travaillé sur ce projet, et il est difficile de renoncer à certaines fonctionnalités lorsque l'on est complètement impliqué. J'avais ainsi l'envie perpétuelle d'ajouter de nouvelles fonctionnalités (et ainsi d'apprendre la manière de le faire).

Il s'agit d'un domaine extrêmement vaste, et en constante évolution. Il développe cette quête de recherche infinie et cette envie d'en apprendre toujours plus.