

Dossier Projet



Titre RNCP (37674)

Niveau 5 - Développeur web et web mobile

Ori&Nori

The screenshot displays four windows of the Ori&Nori application:

- Homepage:** Shows the "Welcome to Ori&Nori" message, a "Join us today and give your pets a vibrant social life" call-to-action, and two registration buttons: "Create Your Account in a Snap" and "Register".
- Pet Profile - Ori ♂:** Features a photo of a Corgi, the name "Ori ♂", breed "Chien, Welsh Corgi Pembroke", and vaccination status "Vaccinated". It also shows the location "Toulouse" and owner "ThomasMaye".
- Pet Profile - Nori ♀:** Features a photo of a cat, the name "Nori ♀", breed "Chat, European", and vaccination status "Vaccinated". It also shows the location "Toulouse" and owner "MaelEzanic".
- Meetup Map:** A map of Toulouse showing various meetup locations with pins. One pin is highlighted with a blue marker and labeled "by Me".
- Meetup Details:** A modal window titled "My Upcomming Meetups" showing a meetup titled "Rencontre entre chats à la maison." It includes details like "by Me", "20/07/2025 16:00", and "Rue du chapeau Rouge - Toulouse". It also indicates "3 Pets have joined this Meetup: European Chat".

Mayé
Thomas

19 Juin 2025

Partie 1 : Présentation Générale du Projet.....	3
1.1. Compétences du référentiel couvertes par le projet.....	3
1.2. Contexte du projet.....	4
1.2.1. Constat et idée.....	4
1.2.2. Présentation du service.....	5
1.2.3. Cahier des charges et expression des besoins.....	5
1.2.4. Les contraintes du projet et les livrables attendus.....	6
1.2.4.1. Les contraintes du projet.....	6
1.2.4.2. Les livrables attendus.....	7
1.2.5. Les critères de réussite.....	7
1. Critères fonctionnels.....	8
2. Critères techniques.....	8
1.3. Environnement humain et technique.....	8
1.3.1. Environnement humain.....	8
1.3.2. Environnement techniques et choix technologiques.....	9
1.3.2.1. Architecture Back-End et base de données.....	9
1.3.2.2. Architecture Front-end.....	10
1.3.2.3. Écosystème de développement et DevOps.....	10
Partie 2 : Les Réalisations Techniques.....	12
2.1. Les réalisations côté Front-end.....	12
2.1.1. Réalisations significatives, choix techniques et sécurité.....	12
2.1.1.1. Identité visuelle et charte graphique.....	12
2.1.1.2. Approche mobile-first et responsive design avec Tailwind CSS.....	14
2.1.1.3. Approche par composants : cohérence et maintenabilité.....	15
2.1.1.4. Prise en compte de la sécurité côté client.....	15
2.1.2. Présentation des maquettes de l'application.....	17
2.1.3. Enchaînement des maquettes (Userflow - Diagramme de Séquence).....	18
2.1.3.1. Userflow.....	18
2.1.3.2. Diagramme de séquence.....	19
2.1.4. Interfaces utilisateur.....	19
2.1.4.1. Optimisation de l'éco-conception.....	19
2.1.4.2. Adaptation web et web mobile : interface responsive.....	20
1. Affichage adaptatif des "Meetups".....	20
2. Le Menu de navigation compressible.....	21
2.1.4.3. Prise en compte de l'accessibilité et conformité au RGAA.....	22
1. Utilisation des balises sémantiques HTML.....	22
2. Outils d'audit et directives d'accessibilité.....	22
3. Gestion des contrastes de couleurs.....	23
4. Navigabilité au clavier et tests basiques avec lecteurs d'écran.....	23
2.1.4.4. Optimisation pour les Moteurs de Recherche (SEO).....	24
1. Utilisation des balises sémantiques HTML.....	24

2. Meta-données et description.....	24
3. Inscription à Google Search Console.....	24
2.1.4.5. Interfaces utilisateur statiques.....	24
1. La card Meetup.....	25
2. Le formulaire de création des meetups.....	26
2.1.4.6. Interfaces utilisateur dynamique.....	27
1. Affichage Conditionnel sur la card "Meetup".....	27
2. Géocodage et affichage sur la carte interactive.....	29
2.2. Les réalisations côté Back-End.....	31
2.2.1. Architecture et fondations Back-End.....	31
2.2.2. Modélisation et implémentation de la base de données.....	32
2.2.3. Structuration des requêtes et logique d'accès aux données.....	36
2.2.4. Implémentation de la logique métier dans l'application.....	37
2.2.4.1. Gestion des utilisateurs et des animaux.....	37
2.2.4.2. Organisation des Meetups.....	39
2.2.4.3. Système d'évaluation.....	40
2.2.4.4. Règles de sécurité et d'accès.....	41
2.2.4.5. Upload et suppression de médias associés.....	42
2.2.5. Sécurité applicative côté back-end.....	43
2.2.6. Résolutions structuré des problèmes.....	45
2.3. Jeu d'Essai et validation des fonctionnalités.....	45
2.3.1. Approche de Test.....	45
2.3.2. Fonctionnalité Représentative : Gestion des Meetups.....	46
2.3.3. Scénarios de test et résultats.....	46
2.3.4. Analyse des écarts et pistes d'amélioration.....	47
Partie 3 : Veille sur les vulnérabilités de sécurité et identification des failles potentielles.....	48
3.1. Démarche de veille.....	48
3.1. Suivi technologique.....	48
3.3. Contre-mesures en place.....	49
3.4. Pistes d'amélioration et suites possibles.....	51
Partie 4 : Documenter le déploiement d'une application dynamique Web ou Web Mobile.....	52
4.1. Choix de l'Environnement d'Hébergement et de la Stratégie de Déploiement.....	52
4.2. Construction et orchestration des conteneurs.....	52
4.2.1. Construction des conteneurs (Dockerfile).....	52
4.2.2. Orchestration avec Docker Compose.....	53
4.3. Gestion des configurations et sécurité du déploiement.....	53
4.4. Résultats du déploiement.....	53

Partie 1 : Présentation Générale du Projet

1.1. Compétences du référentiel couvertes par le projet

Le projet Ori&Nori a permis de mettre en œuvre l'ensemble des compétences professionnelles requises pour l'obtention du titre de Développeur Web et Web Mobile (RNCP 37674), telles que définies dans le Référentiel d'Activités et de Compétences (REAC) et évaluées dans le Référentiel d'Évaluation (RE). Les compétences suivantes ont été spécifiquement développées à travers les réalisations du projet :

Bloc de Compétences	Compétence	Mise en œuvre dans le projet Ori&Nori
Développer la partie Front-end d'une application web ou web mobile sécurisée	Maquetter des interfaces utilisateur web ou web mobile	Maquettes réalisées avec Figma, en adoptant une approche "mobile-first" et en prenant en compte l'expérience utilisateur, l'accessibilité (conformité RGAA), et le respect de la charte graphique définie. L'Userflow a permis de modéliser l'enchaînement des maquettes.
	Réaliser des interfaces utilisateur statiques web ou web mobile	Intégration HTML/CSS avec Tailwind CSS pour un design responsive. Conformité aux maquettes, gestion des mentions légales (RGPD), prise en compte de l'accessibilité.
	Développer la partie dynamique des interfaces utilisateur web ou web mobile	Implémentation des fonctionnalités dynamiques avec Alpine.js et Leaflet.js pour les filtres interactifs, l'affichage de la carte et les interactions utilisateur. Utilisation d'API REST pour la géolocalisation. Le code est documenté.
Développer la partie back-end d'une application web ou web mobile sécurisée	Mettre en place une base de données relationnelle	Conception et implémentation d'une base de données PostgreSQL avec AdonisJS, incluant les tables pour les users, pets, meetups, species, breeds et reviews. Définition du schéma physique des données (ERD) et gestion des droits d'accès.
	Développer des composants d'accès aux données SQL et NoSQL	Mise en œuvre d'API REST sécurisées avec l'ORM Lucid d'AdonisJS pour les opérations CRUD (Création, Lecture, Mise à jour, Suppression). Contrôle et validation des entrées. Gestion de l'intégrité et de la confidentialité des données.
	Développer des composants métier côté serveur	Implémentation de la logique métier principale dans les contrôleurs selon l'architecture MVC d'AdonisJS. Exemples : gestion du système de filtrage d'animaux par critères, organisation des rencontres, gestion des évaluations, et contrôles d'autorisation stricts pour garantir que les utilisateurs ne peuvent modifier que leurs propres données. Le code est documenté.

	Documenter le déploiement d'une application dynamique web ou web mobile	Le projet est conteneurisé avec Docker pour un environnement de développement stable et reproductible, facilitant ainsi les procédures de déploiement. Un Makefile est utilisé pour automatiser les tâches de gestion de l'environnement, se rapprochant d'une démarche DevOps. Hébergement sur un serveur et nom de domaine chez Scaleway et utilisation de Traefik.
--	---	---

Le projet Ori&Nori a permis d'appliquer et de consolider l'ensemble des compétences requises pour le titre de Développeur Web et Web Mobile. Chaque compétence du référentiel a été traitée concrètement, via les choix technologiques et les fonctionnalités implémentées.

Côté Front-end, l'accent a été mis sur l'expérience utilisateur et l'adaptabilité. Le maquettage a défini une interface intuitive. L'utilisation de Tailwind CSS et Alpine.js a garanti un rendu responsive et dynamique. Les préoccupations de sécurité côté client, notamment la protection CSRF, ont été intégrées dès la conception des formulaires.

Pour le Back-End, la robustesse et la sécurité ont guidé les choix. AdonisJS avec son ORM Lucid et PostgreSQL a fourni une architecture solide pour la gestion des données et l'implémentation de la logique métier. La structuration des requêtes, la validation des données et les contrôles d'accès ont été des priorités constantes pour garantir l'intégrité et la confidentialité des informations.

Savoir installer et configurer son environnement de travail en fonction du projet a été crucial. L'utilisation de Docker dès le début du projet a permis d'établir un environnement de développement homogène et de se familiariser avec les pratiques essentielles au déploiement continu. Un Makefile ont été créés pour automatiser les tâches de gestion de cet environnement, améliorant la productivité et la reproductibilité du projet.

1.2. Contexte du projet

Cette section explique le contexte général du projet Ori&Nori, de l'idée initiale à la définition de ses objectifs et de son cadre de réalisation. Elle expose le "pourquoi" et le "comment" de cette application.

1.2.1. Constat et idée

L'idée d'Ori&Nori est née d'un constat quotidien : les animaux de compagnie ont un besoin fondamental de socialisation. Que ce soit pour les chiens ou les chats, les interactions avec leurs congénères sont essentielles à leur équilibre, leur bien-être émotionnel et le développement d'un comportement sain.

Cependant, les modes de vie modernes, particulièrement en milieu urbain, limitent souvent ces opportunités. Les propriétaires manquent de temps, ignorent les lieux adaptés ou craignent les

mauvaises rencontres. Ce manque de socialisation peut entraîner anxiété, comportements destructeurs ou simplement un animal ne pouvant s'épanouir pleinement.

De là est venue l'idée : utiliser la technologie pour briser ces barrières. L'objectif est devenu clair : créer un site web permettant aux propriétaires d'animaux de se connecter, de trouver des partenaires de jeu compatibles et d'organiser des rencontres simplement et en toute sécurité. Ori&Nori est donc une réponse concrète à un besoin réel : améliorer la vie des animaux et, par extension, celle de leurs propriétaires. Le nom Ori&Nori est un hommage à nos propres animaux ayant inspiré ce projet.

1.2.2. Présentation du service

Ori&Nori est un site web pour les propriétaires d'animaux souhaitant enrichir la vie sociale de leurs compagnons. Le service vise à rendre la socialisation animale plus accessible, plus sûre et mieux organisée.

Trois piliers principaux composent l'application :

- **Profils personnalisés** : Chaque animal dispose d'une page dédiée avec photos, race et caractère. Cela permet aux propriétaires d'évaluer la compatibilité avant une rencontre.
- **Géolocalisation** : L'application affiche sur une carte les "Meetups" organisés à proximité, facilitant les rencontres spontanées ou planifiées.
- **Organisation d'événements** : Les utilisateurs peuvent créer ou rejoindre des Meetups existants, qu'il s'agisse d'une promenade ou d'un atelier éducatif.

Le projet est fondé sur des valeurs de bienveillance, de soutien mutuel et de respect du bien-être animal. L'ambition est de créer une communauté de confiance où les propriétaires peuvent échanger, partager des conseils et offrir à leurs animaux les interactions qu'ils méritent.

1.2.3. Cahier des charges et expression des besoins

Pour concrétiser ce projet dans les délais, un Product Minimum Viable (MVP) a été défini. L'objectif est de livrer un produit fonctionnel et utile, répondant aux besoins essentiels de nos utilisateurs.

L'expression des besoins fonctionnels s'articule autour des points suivants :

- **Gestion des utilisateurs :**
 - Création de compte par un visiteur.
 - Connexion et déconnexion sécurisées.
 - Réinitialisation de mot de passe en cas d'oubli.
- **Gestion des animaux de compagnie ("Pets") :**
 - Création de profils détaillés pour chaque animal (nom, espèce, race, photos, traits de caractère).
 - Consultation et modification des profils d'animaux.

- **Gestion des "Meetups" :**
 - Création de Meetup (lieu, date, heure, nombre de participants).
 - Consultation des Meetups disponibles sur une carte interactive.
 - Participation ou retrait d'un Meetup.
 - Filtres pour rechercher un Meetup par type d'animal (espèce, race).
- **Retour d'expérience :**
 - Notation et commentaire de l'expérience et des animaux rencontrés après chaque Meetup.
 - Consultation des notes et commentaires par l'organisateur du Meetup et le propriétaire des animaux.

Ce cahier des charges a servi de feuille de route pour le développement du sites, assurant la construction d'une base solide et cohérente. Pour structurer le développement d'Ori&Nori et garantir que l'application réponde précisément aux besoins de ses futurs utilisateurs, les fonctionnalités ont été définies et détaillées sous forme d'User Stories, comme présenté en **Annexe 1** (p.55).

1.2.4. Les contraintes du projet et les livrables attendus

Tout projet de développement est soumis à un ensemble de contraintes qui façonnent sa réalisation. Pour Ori&Nori, celles-ci étaient de plusieurs ordres.

1.2.4.1. Les contraintes du projet

Contraintes techniques :

- **Apprentissage du framework** : Le choix d'Adonis, un framework Node.js moderne, a posé un défi. Sa documentation moins exhaustive a nécessité un temps important de recherche, d'expérimentation et de résolution de problèmes, impactant directement le planning de développement.
- **Intégration d'API externes** : L'intégration de l'API OpenCage pour la géolocalisation a été une contrainte majeure. Il a fallu maîtriser la communication avec ce service tiers (clés d'API, formats de requêtes et réponses) et traiter les données de géocodage pour un affichage pertinent sur carte. Cela impliquait la gestion des appels asynchrones et l'anticipation des erreurs.
- **Design responsive** : Assurer une expérience utilisateur de qualité sur tous les supports (ordinateur, tablette, mobile) était primordial. Le site a été conçu en "*mobile-first*", mais l'adaptation de l'affichage d'éléments complexes (cartes interactives, formulaires, listes filtrées) a exigé un travail d'intégration CSS conséquent avec Tailwind CSS.
- **Choix d'architecture** : La décision a été prise de privilégier le Server-Side Rendering (SSR) avec des pages légères, sans dépendre de lourdes bibliothèques JavaScript côté client comme celles utilisées dans les architectures Client-Side Rendering (CSR) ou Single Page Application (SPA). Cette approche

a permis d'optimiser les performances initiales et de réduire la complexité du front-end.

Contraintes temporelles : Le projet s'inscrivait dans un cadre académique avec une échéance fixe. Ce calendrier serré a contraint à des choix stratégiques, à la priorisation des fonctionnalités essentielles du MVP et à l'abandon de certaines idées pour garantir la livraison d'un produit fonctionnel à temps.

Contraintes de ressources : En tant qu'équipe de deux développeurs, les ressources humaines et connaissances étaient limitées. Cela a influencé les choix technologiques, orientant vers des solutions open-source et des services proposant des offres gratuites pour les API.

1.2.4.2. Les livrables attendus

À l'issue du projet, les livrables suivants ont été définis :

L'application web Ori&Nori : Un site web fonctionnel avec les capacités suivantes :

- **Gestion des utilisateurs** : Inscription, connexion, déconnexion et réinitialisation de mot de passe.
- **Gestion des animaux** : Création, consultation et modification des profils des animaux.
- **Création de Meetups** : Définition des détails de l'événement.
- **Visualisation et filtrage des Meetups** : Affichage sur carte interactive et filtrage par critères pertinents.
- **Participation aux rencontres** : Possibilité de rejoindre un Meetup.

La base de données : Une base de données PostgreSQL structurée et relationnelle. Elle contient les tables nécessaires au fonctionnement de l'application (utilisateurs, animaux, meetups, etc.) et assure l'intégrité et la sécurité des données.

Le code source complet : L'intégralité du code du projet (Front-end et Back-end). Le code est versionné avec Git, disponible sur un dépôt GitHub, inclut des commentaires et respecte les conventions de codage établies.

Le déploiement de l'application : Le projet est conçu pour être déployé sur un serveur d'hébergement, le rendant accessible publiquement via une URL.

1.2.5. Les critères de réussite

Le succès du projet se mesure à l'atteinte d'objectifs SMART et à la validation des User Stories du MVP. Nous confirmerons sa réussite en nous assurant que les fonctionnalités clés répondent aux besoins utilisateurs et que les performances techniques sont au rendez-vous.

1. Critères fonctionnels

- **Complétion du MVP** : Toutes les fonctionnalités définies dans le cahier des charges du MVP doivent être opérationnelles.
- **Fluidité du parcours utilisateur** : L'utilisateur doit pouvoir accomplir les actions principales (s'inscrire, ajouter un animal, créer/rejoindre un Meetup) simplement et intuitivement, sans blocage.
- **Fiabilité** : L'application doit fonctionner de manière stable, sans bugs critiques sur les fonctionnalités principales.

2. Critères techniques

- **Qualité du code** : Le code doit être lisible, maintenable et respecter les bonnes pratiques de développement (structure MVC claire, nommage cohérent, etc.).
- **Sécurité** : Les failles de sécurité de base (par exemple, injections SQL) doivent être évitées, et les données sensibles (comme les mots de passe) correctement protégées.
- **Performance** : Les temps de réponse de l'application doivent être acceptables pour garantir une bonne expérience utilisateur.

1.3. Environnement humain et technique

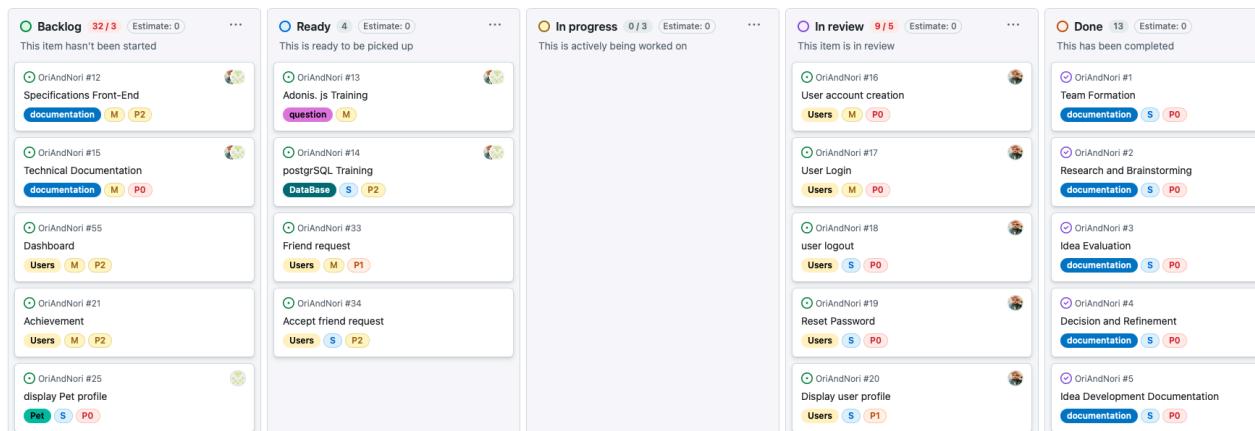
1.3.1. Environnement humain

Le projet a été réalisé par un binôme composé de Maël EZANIC et de moi-même. Une collaboration antérieure a permis un démarrage basé sur la confiance mutuelle et la compréhension des forces de chacun. Une méthodologie inspirée des principes AGILE a été adoptée.

Notre flux de travail quotidien reposait sur des outils et des rituels précis :

- **Gestion de projet** : a servi de tableau Kanban. Chaque fonctionnalité était décomposée en tâches, permettant un suivi visuel de la progression ("À faire", "En cours", "Terminé"). L'outil a été exploité au mieux avec issues, jalons, labels, dates de début et de fin.

Exemple tableau Kanban sur GitHub Projects



- **Communication** : Slack était notre canal de communication principal pour les échanges rapides et le partage d'informations.
- **Synchronisation quotidienne** : Chaque journée commençait par un "stand-up" rapide. L'objectif était de répondre à trois questions : Qu'est ce qui a été fait hier ? Que sera-t-il fait aujourd'hui ? Quels sont les points bloquants ? Ce rituel a assuré un alignement constant, une résolution immédiate des problèmes et un ajustement des priorités en temps réel.

Initialement, les rôles étaient répartis : Maël sur le Back-End, Thomas sur le Front-end. Cependant, la collaboration s'est avérée plus fluide et polyvalente. En pratique, nous avons tous deux contribué à l'ensemble du projet, du Front-end au Back-end.

Cette polyvalence était un choix délibéré, motivé par plusieurs objectifs :

- **Partage de connaissances** : Du temps a été systématiquement pris pour expliquer mutuellement les choix techniques et la logique du code produit. Des sessions de "pair programming" ont été fréquentes pour développer ensemble les fonctionnalités complexes.
- **Montée en compétences mutuelle** : Cette approche a permis une formation réciproque. L'un a approfondi ses compétences back-end grâce à l'autre, et inversement pour le front-end. Le résultat est une compréhension de l'intégralité de l'application.
- **Efficacité et confiance** : Cette collaboration étroite et la confiance mutuelle ont fluidifié le processus de revue de code. La validation se faisait souvent en continu, expliquant le recours limité aux Pull Requests formelles, plus adaptées aux grandes équipes. Dans un contexte d'entreprise, un processus de revue par Pull Request ou Merge request est reconnu comme indispensable pour la qualité et la traçabilité des contributions.

En conclusion, notre duo a fonctionné comme une seule unité de développement, agile et adaptable. Cet environnement humain, basé sur la communication, la confiance et une volonté partagée d'apprendre, a été un facteur déterminant dans la réussite du projet Ori&Nori.

1.3.2 Environnement techniques et choix technologiques

L'architecture technique a été pensée pour former un écosystème cohérent, moderne et performant. Chaque outil a été choisi pour répondre à un besoin précis, de la base de données au déploiement, en passant par l'expérience utilisateur et celle du développeur.

1.3.2.1 Architecture Back-End et base de données

- **AdonisJS (Framework Node.js)** : Cœur de l'application. Choisis pour sa structure MVC robuste. Ses modules intégrés (authentification, ORM, etc.) ont fourni un cadre sécurisé et productif, permettant de se concentrer sur la logique métier.
- **PostgreSQL (Base de données)** : Nous l'avons choisie pour sa fiabilité et sa capacité à bien gérer les informations liées entre elles, ce qui est essentiel pour notre application.

Grâce à ses transactions sécurisées, nous sommes sûrs que les données de nos utilisateurs et de leurs animaux restent exactes et cohérentes.

- **Lucid (ORM d'AdonisJS)** : Pont entre l'application et la base de données PostgreSQL. Son implémentation a simplifié les requêtes (CRUD) et a offert une protection native contre les injections SQL.
- **OpenCage (API de Géocodage)** : Pour la géolocalisation, la conversion des adresses en coordonnées GPS a été déléguée à l'API OpenCage. Son modèle basé sur les données libres d'OpenStreetMap et sa simplicité d'intégration via une API REST en ont fait un choix évident.

1.3.2.2 Architecture Front-end

- **EdgeJS (Moteur de templates)** : Intégré à AdonisJS, EdgeJS a été utilisé pour le rendu des vues côté serveur. Sa syntaxe simple et ses fonctionnalités de layouts et de composants partiels ont permis de construire des pages HTML dynamiques et maintenables.
- **Tailwind CSS (Framework CSS)** : Son approche "utility-first" a été adoptée pour un contrôle total sur le design. Cela a permis de construire une interface sur mesure et responsive rapidement, sans être contraint par des composants préfabriqués.
- **Alpine.js (Framework JavaScript)** : Utilisé pour ajouter de l'interactivité côté client (comme l'ouverture de modales ou la gestion d'onglets) sans la complexité d'un framework plus lourd comme React ou Vue.js. Sa légèreté et sa syntaxe déclarative, s'intégrant directement dans le HTML, étaient parfaites pour cette architecture.
- **Leaflet.js (Bibliothèque de cartographie)** : Cette bibliothèque open-source a été utilisée pour afficher les cartes interactives. Tandis que le back-end utilise OpenCage pour obtenir les coordonnées, Leaflet.js a permis de les représenter visuellement sur le Front-end de manière performante et personnalisable.

1.3.2.3. Écosystème de développement et DevOps

Pour garantir un environnement de travail stable, reproductible et efficace, un point d'honneur a été mis sur l'utilisation d'outils professionnels de gestion et d'automatisation.

- **Docker et Docker Compose** : L'ensemble du projet (application AdonisJS, base de données PostgreSQL) a été conteneurisé avec Docker. Le fichier docker-compose.yml et un fichier Makefile ont permis de définir et de lancer tout l'environnement de développement avec une seule commande. Cette approche garantit que chaque membre de l'équipe travaille sur une configuration identique, éliminant les problèmes de "ça ne marche pas sur ma machine" et préparant le projet pour un déploiement simplifié.
- **DBeaver (Client de Base de Données)** : Durant le développement, DBeaver a été l'outil pour visualiser le schéma de la base de données, inspecter les données en temps réel et exécuter manuellement des requêtes de test.
- **Make (Automatisation des tâches)** : Pour simplifier les commandes Docker et autres tâches répétitives, un Makefile a été créé. Des commandes simples comme make up ou

make down ont servi de raccourcis, améliorant la productivité et rendant le projet plus simple d'utilisation.

- **Outils de développement et sécurité** : VS Code a été configuré comme l'environnement de développement principal, complété par Git pour la gestion des versions. La sécurité de l'environnement de travail a reçu une attention particulière, notamment via la génération et la gestion des clés SSH. Ces clés ont sécurisé les communications avec les dépôts et les serveurs, protégeant ainsi contre les accès non autorisés.

Pour maintenir cet environnement robuste et sécurisé, une veille technologique constante a été mise en place. Cela inclut la surveillance des mises à jour de sécurité de logiciels comme Docker et des extensions de VS Code. Cette pratique a permis de s'assurer que les outils restaient à jour et que les vulnérabilités connues étaient rapidement corrigées.

Partie 2 : Les Réalisations Techniques

2.1. Les réalisations côté Front-end

Le développement Front-end d'Ori&Nori s'est basé sur l'idée qu'une expérience utilisateur réussie est essentielle à l'adoption de l'application. Une approche "*mobile-first*" a été adoptée dès la conception pour garantir une accessibilité et une ergonomie optimales sur tous les appareils, du smartphone à l'ordinateur de bureau.

Nous avons choisi ces technologies parce qu'elles sont récentes, efficaces et faciles à entretenir.

- L'interface a été construite avec le moteur de template EdgeJS, natif du framework back-end AdonisJS, permettant une intégration fluide des données.
- Le design est géré par le framework Tailwind CSS, qui, par son approche "utility-first", a permis de créer une identité visuelle sur-mesure et parfaitement réactive.
- Pour les interactions dynamiques, Alpine.js a été utilisé comme une surcouche légère de JavaScript, notamment pour les composants plus complexes comme la carte interactive Leaflet.js et la dynamique du menu.

2.1.1. Réalisations significatives, choix techniques et sécurité

2.1.1.1. Identité visuelle et charte graphique

Une identité visuelle claire est importante pour établir la confiance avec l'utilisateur.

- **Création du logo :** Le logo d'Ori&Nori a été conçu de manière authentique. D'abord esquissé à la main pour représenter la connexion et la bienveillance entre animaux, le dessin a ensuite été numérisé sur tablette graphique, puis vectorisé avec Affinity Designer. Cette méthode assure un logo unique, personnel et adaptable à divers supports, du favicon aux supports imprimés.



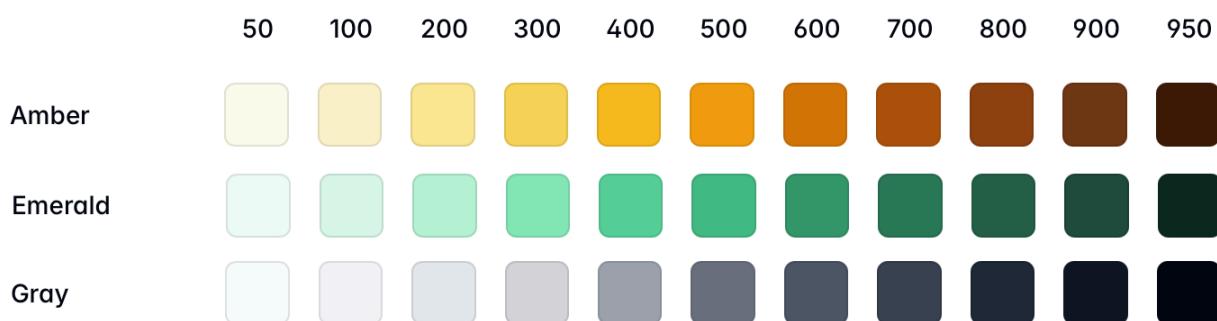
- **Charte graphique :** La palette de couleurs a été choisie pour évoquer la nature, la confiance et la chaleur.
 - **Vert (#047149)** : Couleur dominante, elle symbolise la nature, les promenades et l'équilibre.
 - **Jaune (#FFB317)** : Utilisé pour les appels à l'action principaux, il apporte une touche de chaleur, de joie et d'énergie.

- **Gris (nuances variées)** : Utilisé pour les textes et les fonds, il assure une excellente lisibilité et un contraste suffisant, conformément aux règles d'accessibilité du web (WCAG).



Palette de couleurs

Cette combinaison assure un confort visuel et une hiérarchie claire de l'information, guidant l'utilisateur intuitivement. La charte graphique est en accord avec la palette de couleurs de **Tailwind CSS**, utilisant les teintes Amber(400) et Emerald(700) pour une homogénéité parfaite du site.



Palette de couleurs Tailwind CSS

- **Mode Sombre (Dark Mode)** : Un mode sombre a été intégré pour améliorer le confort visuel en faible luminosité et répondre aux préférences des utilisateurs. L'implémentation utilise la stratégie de "classe" de Tailwind CSS, configurée dans le fichier tailwind.config.js. Un bouton permet d'ajouter ou de retirer la classe dark sur l'élément <html>, activant ainsi les styles préfixés par dark:.

```

17      <h2 class="text-3xl font-semibold text-gray-800 dark:text-white lg:text-4xl mt-4">
18        <div>
19          @if(auth.use('web').isAuthenticated)
20            @!component('components/badge-identity', { username : auth.use('web').user.username })
21          @end
22        </div>
23        Welcome to <span class="text-emerald-700 dark:text-emerald-700">Ori&Nori</span>
24      </h2>

```

Exemple d'utilisation de la class dark

2.1.1.2. Approche mobile-first et responsive design avec Tailwind CSS

L'application a été pensée pour être utilisée en situation de mobilité. Le choix de Tailwind CSS a été déterminant pour mettre en œuvre cette vision.

- **Tailwind CSS** : Son approche “utility-first” permet d'appliquer des styles directement dans le code HTML via des classes. Cela accélère le développement, garantit une cohérence visuelle et évite d'écrire du CSS personnalisé complexe. Le framework permet d'utiliser des composants mais propose des classes utilitaires pour construire un design sur mesure (contrairement à d'autres framework). Il gère nativement la suppression des styles inutilisés en production, ce qui assure des fichiers CSS finaux extrêmement légers et performants.

Exemple : La page affichant la liste des "meetups" est un excellent exemple. Sur un petit écran, les cartes d'événements s'affichent sur une seule colonne pour une lecture verticale naturelle. Sur des écrans plus larges, la même grille s'adapte pour afficher deux, puis trois colonnes.

```
77      {{-- Meetups list section --}}
78      <div class="lg:w-2/3">
79          <div class="grid gap-6 grid-cols-1 md:grid-cols-2 xl:grid-cols-2 2xl:grid-cols-3">
80              @each(meetup in meetups)
81                  @if(new Date(meetup.date) >= new Date())
82
83                  {{-- Meetup card --}}
84                  <div
```

Exemple de code et de page responsive

The screenshot displays the 'My Meetups History' page from a mobile device. The top navigation bar shows the title 'My Meetups History'. Below it, a map of Toulouse and surrounding areas shows several blue location markers. The main content area is a grid of meetup cards, which are displayed in a single column on the mobile screen. Each card includes a small map, the meetup name, the organizer's name, the date and time, and a 'View' button. As the screen size increases, the grid adapts to show two or three columns of cards. The cards contain details such as the number of pets joined, the names of the pets, and a 'View' button. The overall design is clean and modern, utilizing Tailwind CSS's utility-first approach for responsive layout.

Ce code démontre l'efficacité de l'approche “*mobile-first*”. La classe grid-cols-1 est le style par défaut (mobile). Les préfixes md: et lg: sont des “media queries” qui s'appliquent uniquement sur les écrans de taille moyenne et grande. Cette approche déclarative rend le code lisible et facile à maintenir.

2.1.1.3. Approche par composants : cohérence et maintenabilité

Pour une interface utilisateur cohérente et un développement rapide, Ori&Nori a été construite avec une approche par composants. Au lieu de redéfinir le style de chaque élément, des composants réutilisables (cartes de profil, boutons) ont été créés directement en HTML, en utilisant les classes utilitaires de Tailwind CSS. Cela a permis de constituer un ensemble de composants réutilisables partout sur le site.

Exemple : Création d'un bouton d'action

Un bouton n'est pas juste un rectangle cliquable ; il doit communiquer son état (normal, survolé, focus) et son importance (primaire, secondaire). Le code ci-dessous montre comment un bouton d'action principal a été stylisé.

```
1   <button class="inline-block rounded-lg bg-emerald-700 px-4 py-2.5 text-sm font-medium te
2   type="{{ type || 'submit' }}">>{{ text }}</button>
```

Exemple de création d'un composant

Ainsi, nous pouvons réemployer le même bouton en l'appelant de la manière suivante :

```
29           <div class="flex w-full justify-center">
30             @!component('components/button-simple/full', { text: "Register" })
31           </div>
```



Exemple d'insertion d'un composants et aperçu

Cette approche par composants garantit que tous les boutons d'action de l'application auront une apparence et un comportement identiques, renforçant ainsi la cohérence de l'interface. Bien entendu, l'exemple du bouton est le plus simple, mais tous les autres composants se comportent et s'utilisent de la même manière : les bannières, la carte interactive, etc ...

2.1.1.4. Prise en compte de la sécurité côté client

La sécurité de l'application est une priorité qui est traitée dès le Front-end. Bien que les validations les plus importantes soient effectuées côté serveur, l'interface utilisateur intègre deux mécanismes de protection essentiels.

- **Protection contre les attaques CSRF** : Pour se prémunir contre les failles de type “Cross-Site Request Forgery”, tous les formulaires envoyant des données (connexion,

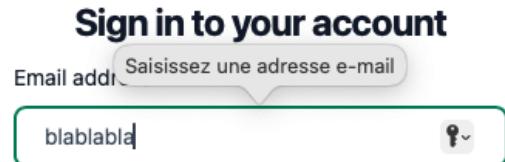
inscription, etc.) sont protégés. Nous utilisons pour cela le middleware shield d'AdonisJS. L'intégration est simple et robuste, comme le montre cet extrait du formulaire de connexion :

```
21      <form class="space-y-6" action="{{ route('auth.register') }}" method="POST">
22        {{ csrfField() }}
23
24        @!component('components/input', { name: 'username', label: 'Username', required: true })
25        @!component('components/input', { name: 'email', label: 'Email address', type: 'email', required: true })
26        @!component('components/input', { name: 'password', label: 'Password', type: 'password', required: true })
27
28        <div>
29          <div class="flex w-full justify-center">
30            @!component('components/button-simple/full', { text: "Register" })
31          </div>
32        </div>
33      </form>
```

Exemple d'un code pour un formulaire sécurisé

La directive {{ csrfField() }} ajoute automatiquement un jeton de sécurité unique et caché au formulaire. Ce jeton est ensuite vérifié par le serveur à chaque soumission, garantissant que la requête provient bien de notre application et non d'un site tiers malveillant.

- **Validation des entrées par le navigateur** : Pour améliorer l'expérience utilisateur et éviter l'envoi de formulaires invalides, la validation HTML5 native a été employée. Des attributs tels que required ou type="email" sur les champs de formulaire permettent au navigateur de vérifier les données avant leur envoi. Cette validation côté client sert de premier filtre et complète la validation principale, qui est effectuée côté back-end. Le formulaire de connexion est un bon exemple de cette approche de sécurité par la validation des données.

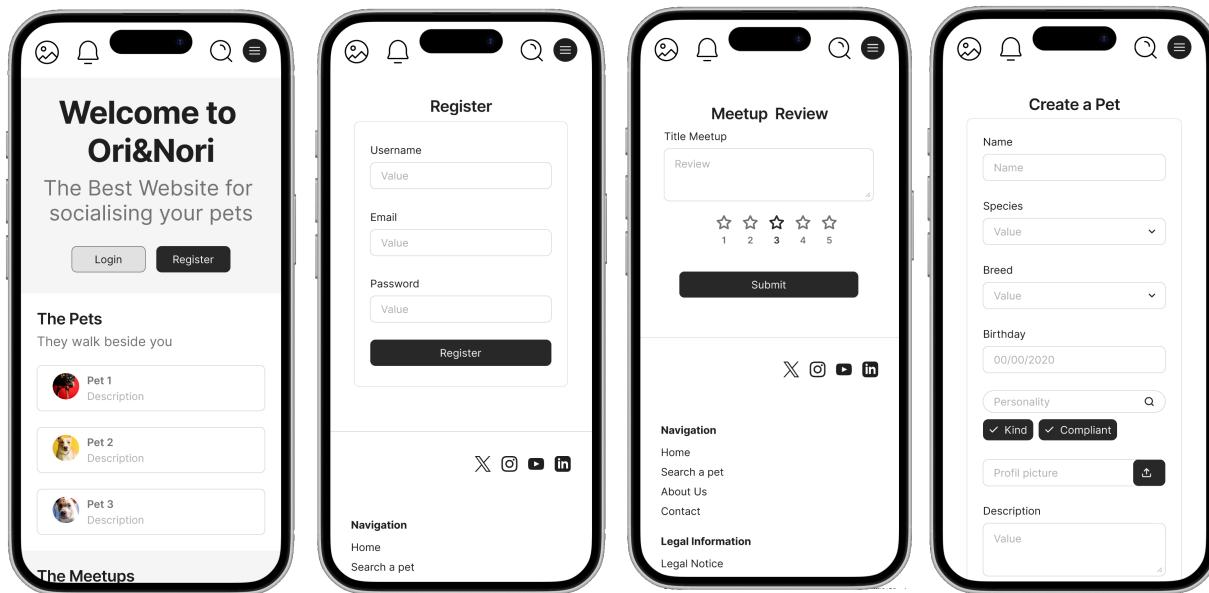


En résumé, nos choix technologiques se sont portés sur Tailwind CSS pour développer rapidement une interface sur-mesure, réactive et cohérente, et sur AdonisJS pour nous fournir un socle robuste et sécurisé, simplifiant l'intégration des données et la protection des formulaires.

Avec du recul, cette synergie a permis un développement rapide, performant et sécurisé, répondant parfaitement aux exigences du projet Ori&Nori.

2.1.2. Présentation des maquettes de l'application

Avant toute ligne de code, une phase de maquettage a été réalisée sur l'outil Figma. Conformément à notre approche “mobile-first”, les maquettes ont d'abord été pensées pour un usage sur smartphone, qui représente la majorité des cas d'utilisation pour une application sociale comme Ori&Nori. Ces wireframes ont défini la structure, l'agencement des éléments et les parcours utilisateurs principaux.



Exemple de maquettes “mobile first” d’Ori&Nori monochrome réalisée avec Figma

La conception de l'interface utilisateur a suivi plusieurs principes clés :

- **Simplicité et accessibilité** : Un design épuré a été privilégié pour comprendre rapidement les fonctionnalités. Par exemple, l'écran d'inscription propose un formulaire clair et concis avec un nombre minimal de champs requis, juste l'Username, l'email et un mot de passe, pour encourager l'inscription.
- **Mise en évidence des éléments clés** : Les pages principales, comme celles d'inscription, de gestion des Meetups et des animaux, sont structurées pour rendre les informations importantes immédiatement accessibles. Elles sont toutes constituées avec un visuel identique.
- **Pratiques UX/UI** : Les boutons d'action sont visibles et placés stratégiquement pour éviter toute confusion. Par exemple, le bouton "Créer un Meetup" est rapidement accessible pour une interaction immédiate. De même, les boutons de validation sont positionnés à la fin de chaque formulaire, assurant une logique claire pour l'utilisateur.
- **Respect de l'accessibilité (RGAA)** : Les maquettes ont été conçues en tenant un maximum compte des exigences d'accessibilité, notamment celles du Référentiel Général d'Amélioration de l'Accessibilité (RGAA). Cela inclut des contrastes de couleurs

suffisants pour les textes et les éléments interactifs (par exemple, un texte sombre sur un fond clair) l'utilisation d'attributs alt pertinents pour les images, et l'intégration des attributs ARIA (Accessible Rich Internet Applications) pour améliorer la sémantique et l'interaction des composants dynamiques, afin que l'application soit utilisable par les personnes en situation de handicap.

- **Palette de Couleurs** : la palette de couleurs aide à améliorer la visibilité. Des couleurs distinctes sont utilisées pour les actions primaires et secondaires.

2.1.3. Enchaînement des maquettes (Userflow – Diagramme de Séquence)

2.1.3.1. Userflow

L'Userflow, présenté en **Annexe 2** (p.56), offre une visualisation synthétique de l'architecture de navigation et de l'enchaînement des écrans de l'application Ori&Nori. Cet outil a été essentiel pour valider la logique des parcours utilisateurs et garantir une expérience intuitive et cohérente avant le développement.

Il met en évidence deux parcours principaux :

1. **Le Parcours du Visiteur** : Ce flux représente l'expérience d'un utilisateur non authentifié.
 - **Point d'entrée** : Le parcours débute sur la page d'accueil ("Welcome to Ori&Nori").
 - **Actions possibles** : Le visiteur peut choisir de s'inscrire (Register) ou de se connecter (Login). Des pages comme "About" et "Contact", bien que accessibles, ne sont pas représentées sur l'Userflow.
 - **Objectif** : L'inscription ou la connexion mène à un état authentifié, donnant accès au reste de l'application (symbolisé par le retour à une page d'accueil connectée "Home" avec un message personnalisé par utilisateur).
2. **Le Parcours de l'utilisateur connecté** : Une fois authentifié, l'utilisateur accède à l'ensemble des fonctionnalités via le menu principal.
 - **Navigation centrale** : Le menu sert de hub, menant aux différentes sections : Profile, Meetup, Pet (incluant des actions comme "Search Pet", "Search Meetup", "My Meetups", etc.).
 - **Logique des fonctionnalités** : Le schéma illustre l'enchaînement des actions. Par exemple, une recherche de rencontre (Search Meetup) mène à une page de résultats, d'où l'utilisateur peut rejoindre (Join) un événement spécifique.
 - **Fonctionnalités avancées** : Le flux intègre également des parcours plus complexes, tel le système d'avis (Review), montrant la possibilité de laisser un avis sur un animal (Pet Review) ou une rencontre (Meetup Review) après participation.

En conclusion, cet Userflow a été un outil de conception essentiel. Il a permis de :

- Valider l'ergonomie et la logique de l'application avant de coder.
- Clarifier les liens entre les différentes maquettes et fonctionnalités.
- Fournir une vision d'ensemble claire du projet, facilitant la communication et la répartition des tâches de développement.

L'Userflow illustre une conception d'interface structurée, centrée sur l'expérience utilisateur.

2.1.3.2. Diagramme de séquence

En complément de cet Userflow qui présente le parcours utilisateur global, la conception a été approfondie avec des diagrammes de séquence. Ces diagrammes détaillent les interactions techniques entre les différentes couches de l'application (Front-end, Back-end, base de données) pour des fonctionnalités précises.

Afin d'illustrer cette démarche, deux diagrammes de séquence clés sont disponibles en annexe.

- **Diagramme de séquence pour la création d'un profil Pet (Annexe 3 - p.57)**: Il décrit les étapes de validation des données, de la requête POST jusqu'à l'insertion en base de données et la confirmation à l'utilisateur.
- **Diagramme de séquence pour la planification d'un Meetup (Annexe 4 - p.58)** : Il montre non seulement la création d'un meetup par un utilisateur, mais aussi comment un autre utilisateur peut consulter et rejoindre cet événement.

Ces documents techniques ont permis d'anticiper l'implémentation des routes de l'API et la logique des contrôleurs, assurant une construction du Back-end parfaitement alignée avec les besoins du Front-end.

2.1.4. Interfaces utilisateur

Dans cette partie, l'objectif est de montrer, à travers des exemples de code et des explications sur nos choix de conception, comment nous avons construit l'interface utilisateur, en la rendant à la fois fonctionnelle, dynamique et adaptable à différents supports.

2.1.4.1. Optimisation de l'éco-conception

Pour réduire la consommation de ressources de nos interfaces, qu'elles soient statiques ou dynamiques, nous avons mis en œuvre plusieurs choix techniques clés :

- **Rendu côté serveur (SSR) avec EdgeJS** : Cette approche minimise le JavaScript envoyé au client, allégeant les pages et accélérant les temps de chargement, particulièrement bénéfique pour les connexions ou appareils limités.
- **Interactivité légère avec Alpine.js** : Contrairement aux frameworks JavaScript plus lourds (React, Vue.js, Svelte), Alpine.js offre une réactivité ciblée avec une empreinte logicielle minimale.

- **CSS optimisé avec Tailwind CSS** : En production, Tailwind CSS élimine les classes inutilisées, garantissant des fichiers CSS très légers, même sans bundlers comme Vite.

Ces méthodes combinées réduisent la quantité de données transférées et traitées, diminuant ainsi significativement l'impact environnemental de l'application.

2.1.4.2. Adaptation web et web mobile : interface responsive

Pour garantir une expérience utilisateur optimale sur tous les appareils, du smartphone à l'ordinateur de bureau, nous avons adopté une approche "*mobile-first*" et utilisé le framework CSS Tailwind CSS. Cette méthode nous a permis de construire des interfaces qui s'adaptent nativement à la taille de l'écran.

1. Affichage adaptatif des "Meetups"

La page listant les "meetups" est un exemple de notre approche responsive. Sur un grand écran, les informations sont organisées en une grille multi-colonnes pour une vue d'ensemble efficace. Sur un appareil mobile, cette grille se transforme en une simple colonne, rendant chaque "meetup" facilement lisible et accessible.

En **Annexe 5** en p.59, se trouve une capture d'écran comparative : la page des "meetups" sur un grand écran (web) et sur un petit écran (mobile).

Ce comportement est obtenu grâce aux classes conditionnelles de Tailwind CSS. Voici un extrait de code illustrant cette logique :

```

77      {{-- Meetups list section --}}
78      <div class="lg:w-2/3">
79          <div class="grid gap-6 grid-cols-1 md:grid-cols-2 xl:grid-cols-2 2xl:grid-cols-3">
80              @each(meetup in meetups)
81              @if(new Date(meetup.date) >= new Date())
82
83                  {{-- Meetup card --}}

```

Exemple de code adaptatif

Explication :

- grid: Active l'affichage en grille.
- grid-cols-1: Par défaut (sur mobile), les éléments sont sur une seule colonne.
- md:grid-cols-2: Sur les écrans de taille moyenne (md) et plus, la grille passe à deux colonnes.
- lg:grid-cols-3: Sur les grands écrans (lg), elle s'étend à trois colonnes.

Non seulement cette approche a été appliquée sur tous les affichages des meetups (My meetups, My upcoming Meetups, My meetups History) mais aussi pour l'affichage de la liste

des Pets, garantissant une cohérence et une ergonomie sur l'ensemble du site (voir exemple en **Annexe 6** en p.60).

2. Le Menu de navigation compressible

Le menu de navigation est un autre composant clé de l'expérience utilisateur. Sur un grand écran, il est entièrement visible. Sur mobile, il se compresse en une icône "burger" pour économiser de l'espace.

Également sur les **Annexes 5 et 6** (en p.59 & 60), nous pouvons observer le menu déployé sur grand écran, et une avec l'icône "burger" sur mobile.

Le code ci-dessous montre comment les classes de Tailwind sont utilisées pour afficher ou masquer les liens de navigation en fonction de la taille de l'écran. Le design de notre menu utilise une combinaison de Tailwind CSS et Alpine.js pour gérer la responsivité en fonction de la taille de l'écran. Ici, Alpine.js nous permet de gérer les interactions dynamiques du menu comme l'ouverture/fermeture du menu mobile, et leurs animations sans avoir à écrire de JavaScript complexe. Pour installer Alpine.js ici, il suffit de charger la bibliothèque de cette manière.

```
<script defer src="https://cdn.jsdelivr.net/npm/alpinejs@3.x.x/dist/cdn.min.js"></script>
```

lg:hidden permet de masquer le menu hamburger lorsqu'on dépasse une largeur d'écran supérieur à 1024px.

```
64
65      <!-- Mobile menu button -->
66      <div class="lg:hidden flex justify-end">
67
```

Si la taille de l'écran est inférieur à la classe lg, on utilise Alpine.js pour afficher ou masquer le menu avec is open.

```
<button x-cloak @click="isOpen = !isOpen" type="button"
        class="text-gray-500 dark:text-gray-200 hover:text-gray-600 dark:hover:text-gray-400
        aria-label="toggle menu">
```

Et s'il est open (c'est à dire qu'on a cliqué sur l'hamburger) alors le menu passe en mode vertical par translation. Tant qu'on ne clique pas sur un élément du menu, une croix est affichée.

```
<!-- Mobile Menu open: "block", Menu closed: "hidden" -->
<div x-cloak :class="[isOpen ? 'translate-x-0 opacity-100' : 'opacity-0 -translate-x-full']"
        class="space-y-1 absolute inset-x-0 z-20 w-full px-6 py-4 transition-all duration-300 ease-in"
```

Et si l'écran est supérieur à la classe lg, alors le menu est aligné horizontalement grâce à lg:flex-row.

```
131      <div class="flex flex-col lg:flex-row items-center">
132
133      <div
```

2.1.4.3. Prise en compte de l'accessibilité et conformité au RGAA

La conception d'Ori&Nori a intégré dès le départ les principes d'accessibilité numérique afin de rendre l'application utilisable par le plus grand nombre, y compris les personnes en situation de handicap. Bien que notre expertise dans ce domaine soit en développement, nous avons veillé à appliquer les bonnes pratiques fondamentales et à utiliser les outils disponibles pour progresser.

1. Utilisation des balises sémantiques HTML

L'une des premières étapes a été l'utilisation rigoureuse des balises sémantiques HTML. Plutôt que de se limiter à des `<div>` génériques, nous avons structuré nos pages avec des éléments qui donnent du sens au contenu :

- `<header>` pour les en-têtes de section.
- `<nav>` pour les blocs de navigation.
- `<main>` pour le contenu principal et unique de la page.
- `<footer>` pour les pieds de page.
- Des titres hiérarchisés (`<h1>`, `<h2>`, etc.) pour structurer le contenu de manière logique, facilitant la navigation pour les lecteurs d'écran.
- Des balises `<button>` pour les éléments cliquables ayant une action, et `<form>` pour les formulaires, garantissant que ces éléments sont correctement identifiés et interagissent de manière attendue avec les technologies d'assistance.

Par exemple, le menu de navigation compressible sur mobile, bien que stylisé avec Tailwind CSS et Alpine.js, s'appuie sur une structure `<nav>` contenant des liens sémantiques. L'icône est implémentée comme un `<button>` avec un attribut `aria-label="toggle menu"`, permettant aux utilisateurs de lecteurs d'écran de comprendre son rôle.

2. Outils d'audit et directives d'accessibilité

Pour valider et améliorer l'accessibilité de nos interfaces, nous nous sommes appuyés sur les ressources du W3C (World Wide Web Consortium), notamment les Web Content Accessibility Guidelines (WCAG). Nous avons également utilisé des outils d'audit d'accessibilité comme l'extension "Wave Evaluation Tool" pour navigateur sous forme de plugin. Cet outil nous a permis d'identifier visuellement les problèmes potentiels (contrastes insuffisants, attributs alt manquants pour les images, structure de titres incorrecte) et de les corriger au fur et à mesure du développement.

Capture d'écran d'Ori&Nori analysée avec Wave, montrant des problèmes d'accessibilité

3. Gestion des contrastes de couleurs

Nous avons également porté attention aux contrastes de couleurs, un aspect important pour les personnes malvoyantes. La palette de couleurs d'Ori&Nori (vert émeraude, jaune ambre, et diverses nuances de gris) a été choisie en partie pour assurer une bonne lisibilité, et nous avons utilisé des outils de vérification de contraste pour nous assurer que les combinaisons de couleurs texte/arrière-plan respectaient les ratios recommandés par les WCAG. Dans l'inspecteur du navigateur de Firefox, dans l'onglet accessibilité, nous pouvons voir des alertes de type "by PierrePatiche" contraste avec un focus sur l'élément à modifier pour convenir.

4. Navigabilité au clavier et tests basiques avec lecteurs d'écran

Enfin, la navigabilité au clavier a été testée régulièrement. L'utilisation de la touche Tab pour parcourir les éléments interactifs et le recours à des lecteurs d'écran basiques (comme VoiceOver sur macOS, même si ce n'était pas un test exhaustif) nous ont permis de vérifier que l'ordre de tabulation était logique et que tous les éléments interactifs étaient accessibles sans souris.



Capture d'écran de l'ordre de tabulation pour test VoiceOver

2.1.4.4. Optimisation pour les Moteurs de Recherche (SEO)

La visibilité sur les moteurs de recherche est un facteur clé pour l'accessibilité et l'adoption d'Ori&Nori par son public cible. Nous avons intégré des pratiques d'optimisation pour les moteurs de recherche (SEO) dès la conception des interfaces statiques, en complément de nos efforts pour l'accessibilité.

1. Utilisation des balises sémantiques HTML

Conformément aux bonnes pratiques de SEO, l'utilisation rigoureuse des balises sémantiques HTML a été une priorité. Au-delà de l'accessibilité, ces balises (comme `<header>`, `<nav>`, `<main>`, `<footer>`, et les titres hiérarchisés `<h1>`, `<h2>`, etc.) structurent le contenu de manière logique, ce qui facilite l'indexation et la compréhension du site par les robots des moteurs de recherche. Chaque page est ainsi balisée pour indiquer clairement son contenu principal, ses sections de navigation, et d'autres éléments clés, permettant une meilleure interprétation thématique par Google et autres moteurs.

2. Meta-données et description

Des balises méta pertinentes, telles que les balises de titre (`<title>`) et les méta-descriptions, sont utilisées pour chaque page afin de fournir des résumés concis et informatifs de leur contenu. Cela permet aux moteurs de recherche de mieux comprendre le sujet de la page et d'afficher des extraits plus pertinents dans les résultats de recherche.

```
@layout.app({ title: "Create a New Meetup" })
@slot('meta')
<meta name="description" content="A page to create a new meetup made with EdgeJS">
@endslot
```

3. Inscription à Google Search Console

Pour un suivi et une optimisation continu de notre visibilité, Ori&Nori a été inscrit sur Google Search Console. Cet outil nous permet de surveiller la performance du site dans la recherche Google, d'identifier les éventuelles erreurs d'indexation, de soumettre des sitemaps et de recevoir des alertes sur les problèmes d'indexation. Cette intégration permet une analyse précise du référencement et un ajustement réactif de la stratégie SEO en fonction des données réelles. Nous nous assurons ainsi que le site reste visible et bien positionné pour le public ciblé dès son déploiement.

2.1.4.5. Interfaces utilisateur statiques

Les interfaces statiques sont les fondations de notre application. Elles présentent l'information sans interaction dynamique complexe. Nous avons mis un point d'honneur à créer des composants réutilisables et clairs.

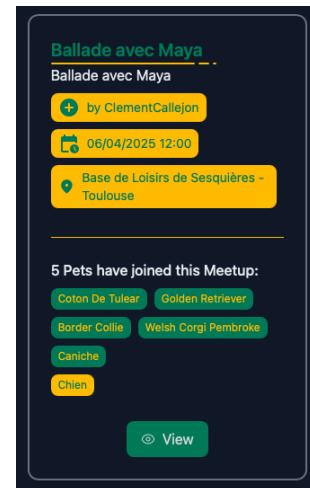
1. La card Meetup

La card "Meetup" est un composant central, affichant les informations essentielles d'un événement.

```
87      {{-- Existing meetup card content --}}
88      <h2 class="text-xl font-semibold text-emerald-700">{{ meetup.title }}</h2>
89      <div class="flex items">
90          <span class="inline-block w-40 h-0.5 bg-amber-400 rounded-full"></span>
91          <span class="inline-block w-3 h-0.5 mx-1 bg-amber-400 rounded-full"></span>
92          <span class="inline-block w-1 h-0.5 bg-amber-400 rounded-full"></span>
93      </div>
94
95      <div class="space-y-2 dark:text-gray-200 text-gray-600">
96          {{ meetup.description }}
97
98          <p class="mt-2 flex items-center">
99              <a href="{{ route('auth.display_user_profile', { id: meetup.userId }) }}">
100                 <span
101                     class="bg-amber-400 text-emerald-700 hover:bg-amber-500 hover:text-emerald-800 px-2 py-1 rounded-lg flex items-center mr-2 text-sm">
102                     <svg class="mr-2" xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24">
103                         <path fill="currentColor">
104                             d="M11 13v3q0 .425.288.713T12 17t.713-.288T13 16v-3h3q.425 0 .713-.288T17 12t-.288-.712T16 11h-3V8q0-.425-.288-.712T12 7t-.712.288"
105                     </svg>
106                     by {{ auth.user && meetup.userId === auth.user.id ? 'Me' : meetup.organizer }}>
107                 </span>
108             </a>
109         </p>
110
111         <p class="mt-2 flex items-center">
112             <span class=" bg-amber-400 text-emerald-700 px-2 py-1 rounded-lg flex items-center mr-2 text-sm">
113                 <svg class="mr-2" xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24">
114                     <path fill="currentColor">
115                         d="M5 22q-.825 0-1.412-.587T3 20V6q0-.825.588-1.412T5 4h1V2h2v2h8V2h2v2h1q.825 0 1.413.588T21 6v4.675q0 .425-.288.713t-.712.287t-.712.288"
116                     </svg>
117                     {{ meetup.formattedDate }}>
118                 </span>
119             </p>
```

Extrait et exemple de code statique pour les cards Meetups

Ce code est purement structurel (HTML) et stylisé avec Tailwind CSS. Il définit la disposition de l'image, du titre, de la description et des tags. Les données ({{ meetup.title }}, etc.) sont injectées par le moteur de template EdgeJS d'AdonisJS, mais à ce stade, la structure elle-même est statique.



Aperçu du card Meetup

2. Le formulaire de création des meetups

Les formulaires sont essentiels pour permettre aux utilisateurs de contribuer à la vie de l'application. Nous avons mis en place une approche basée sur des composants réutilisables pour garantir la cohérence visuelle et simplifier la maintenance. Le formulaire de création d'un "meetup" en est un exemple.

The screenshot shows the Ori&Nori website's user interface. At the top, there is a dark header with the logo 'ORI&NORI' featuring a cat and a dog, followed by a user profile for 'ThomasMaye' and navigation links for 'Home', 'About', 'Contact', 'Meetups', 'Pets', and 'Profile'. Below the header, the main content area has a dark background with light-colored text fields. The title of the page is 'Create A Meetup'. The form fields include:

- What is the name of your meetup?**: A text input field labeled 'Title'.
- Choose a type of meetup**: A dropdown menu labeled 'Select an option'.
- When is your meetup?**: A date and time input field showing '17/06/2025 12:30'.
- Select your pets for the walk:**: A checkbox list with options 'Chacha' and 'Ori'.
- Where did you want the meetup to take place?**: Address inputs for 'Address', 'Additional Address', 'Postal Code', and 'City'.
- Add a description of your meetup**: A text area labeled 'Description'.

A green 'Create' button is located at the bottom right of the form.

Capture d'écran du formulaire de création d'un "Meetup"

Le code de ce formulaire est géré par un template EdgeJS en **Annexe 7** en p.61 & 62. Plutôt que d'écrire des balises HTML <input> répétitives, nous appelons un composant personnalisé que nous avons créé : @component('components/input') voir en **Annexe 8** en p.63.

Pour ce qui est du code du formulaire :

- Approche par composant** : L'utilisation de @component('components/input', { ... }) est au cœur de notre formulaire. Cela nous permet de réutiliser le même code pour générer différents types de champs (text, datetime-local, select, etc.) en passant simplement des propriétés comme name, type ou options.
- Sécurité** : L'appel {{ csrfField() }} est une mesure de sécurité essentielle fournie par AdonisJS pour protéger le formulaire contre les attaques de type Cross-Site Request Forgery (CSRF).

- **Rendu dynamique** : La boucle @each(pet in pets) démontre la capacité du template à générer dynamiquement une liste de cases à cocher, une pour chaque animal appartenant à l'utilisateur.
- **Liaison au Backend** : L'attribut action="{{ route('createMeetup') }}" lie la soumission du formulaire à une route nommée dans notre backend, ce qui est une bonne pratique du modèle MVC (Modèle-Vue-Contrôleur).

Pour le code du composant input :

- **Logique conditionnelle** : Le composant utilise une série de @if/@elseif pour inspecter la propriété type et rendre le bon élément HTML (<select>, <textarea>, <input>). Cela le rend extrêmement flexible.
- **Gestion des erreurs de validation** : La directive @inputError(name) est une fonctionnalité puissante d'AdonisJS. Si le backend retourne une erreur de validation pour un champ spécifique (par exemple, "le titre est requis"), ce bloc l'affiche automatiquement juste en dessous du champ concerné, offrant un retour clair à l'utilisateur.
- **Repopulation du formulaire** : L'expression value="{{ old(name) ?? value }}" est une pratique essentielle pour une bonne expérience utilisateur. Si la soumission du formulaire échoue à cause d'une erreur de validation, old(name) recharge la valeur précédemment saisie par l'utilisateur. Cela lui évite de devoir remplir à nouveau tout le formulaire.

2.1.4.6. Interfaces utilisateur dynamique

La partie dynamique est ce qui donne vie à l'application. Elle gère les interactions des utilisateurs et l'affichage conditionnel des informations.

1. Affichage Conditionnel sur la card "Meetup"

Une page n'est plus un simple document statique, mais une interface qui s'adapte en temps réel aux données, à l'utilisateur connecté et à ses actions. La page "My Upcoming Meetups" est un exemple central de cette approche. Elle affiche une vue personnalisée pour chaque utilisateur, listant uniquement les événements qu'il a créés ou rejoints.

Son caractère dynamique est le résultat de plusieurs mécanismes mis en œuvre avec le moteur de template EdgeJS d'AdonisJS.

My Upcoming Meetups

Here you can find all the meetups you have joined or created.
Click on a meetup to view more details (or edit it if you are the creator).

Détente, Jeux dans un Parc
Un moment de détente avec Nala.
by PierrePatiche
15/07/2025 16:00
Parc Bidot - Fonsorbes

2 Pets have joined this Meetup:
Border Collie Welsh Corgi Pembroke Chien

Ballade avec Ori
Je vous propose une ballade dans mon quartier avec Ori.
by Me
31/07/2025 18:00
67 Avenue Maurice Bourgès-Maunoury – Toulouse

1 Pet has joined this Meetup:
Welsh Corgi Pembroke Chien

View Edit Delete

Capture d'écran de "My Upcoming Meetups"

En Annexe 9 en p.64, le code réalisé pour la page "My Upcoming Meetups". Cette page est dynamique car son contenu est généré et adapté à la volée en fonction de plusieurs facteurs :

- **Rendu conditionnel basé sur les données** : La structure même de la page change grâce à la condition @if(meetups.length === 0). Si le contrôleur backend ne renvoie aucun meetup pour l'utilisateur, la page affiche un message l'invitant à en créer un. Dans le cas contraire, elle affiche la mise en page principale avec la carte et la liste. C'est le dynamisme le plus fondamental : la vue s'adapte à la présence ou à l'absence de données.
- **Personnalisation de l'interface pour l'utilisateur** : L'interface n'est pas la même pour tout le monde.
 - **Propriétaire de l'événement** : La condition @if(auth.user && meetup.userId === auth.user.id) vérifie si l'utilisateur actuellement authentifié est le créateur du meetup. Si c'est le cas, les boutons "Edit" et "Delete" apparaissent, lui donnant des droits de gestion spécifiques. Ces boutons sont masqués pour tous les autres utilisateurs.
 - **Affichage personnalisé** : La ligne by {{ auth.user && meetup.userId === auth.user.id ? 'Me' : meetup.organizer }} utilise un opérateur ternaire pour afficher "Me" si l'utilisateur consulte son propre événement, ou le nom de l'organisateur sinon.

- **Filtrage et traitement des Données dans la Vue** : La page reçoit une liste de meetups, mais elle effectue un traitement supplémentaire : le Filtrage des événements passés. La condition @if(new Date(meetup.date) >= new Date()) à l'intérieur de la boucle garantit que seuls les meetups à venir sont affichés dans la liste, même si les données contenaient des événements passés.

Toutes les pages liées au meetups sont créées sur le même principe. Bien entendu, chaque pages ont des logiques d'affichage @if/else différentes.

2. Géocodage et affichage sur la carte interactive

Une des fonctionnalités dynamiques les plus importantes de Ori&Nori est l'affichage des "meetups" sur une carte interactive. Le processus complet combine une API externe pour le géocodage, une architecture de composants serveur avec EdgeJS, et une interaction côté client gérée par Alpine.js et Leaflet.js. Vous en trouverez un exemple sur la capture d'écran précédente "My Upcoming Meetups" et le code associé en **Annexe 9** en p.64.

Le processus peut être décomposé en quatre étapes clés :

Géocodage (Back-end) : Lors de la création ou de la modification d'un meetup, l'adresse postale est envoyée à notre contrôleur AdonisJS. Celui-ci communique avec l'API externe OpenCage pour convertir cette adresse en coordonnées GPS (latitude et longitude), qui sont ensuite enregistrées dans notre base de données PostgreSQL avec le meetup.

Préparation des Données (Server-side): Lorsque la page "My Upcoming Meetups" est affichée, le contrôleur transmet la liste des meetups (avec leurs coordonnées) au template EdgeJS. Nous utilisons alors notre système de composants pour préparer l'affichage de la carte. Dans l'extrait suivant, nous parcourons chaque meetup et, pour chacun, nous appelons notre composant map_marker, lui passant les informations nécessaires.

Extrait de code pour l'affichage de la carte interactive dans le HTML

```

55      {{-- Map section --}}
56      <div class="lg:w-1/3">
57          <div class="sticky top-20">
58              <div class="border-2 border-gray-300 rounded-lg">
59                  @component('components/map/display_map', { center: center, zoom: zoom, markers: markers, style: 'min-height: 300px;', class: 'lg:h-[500px]' })
60                  @each(meetup in upcomingMeetups)
61                  @component('components/map/map_marker', {
62                      latitude: meetup.latitude,
63                      longitude: meetup.longitude,
64                      label: meetup.title,
65                      description: meetup.description,
66                      date: meetup.date,
67                      address: meetup.address,
68                      city: meetup.city,
69                  })
70                  @end
71                  @end
72                  @endcomponent
73              </div>
74          </div>
75      </div>
76

```

Architecture de Composants (EdgeJS) : La communication entre le composant parent `display_map` et ses enfants `map_marker` est la clé de notre système.

- **Le composant parent `display_map`** : Il initialise un objet `map` qui contiendra les données de la carte et une liste vide de markers. Grâce à `@inject`, il rend cet objet accessible à tous ses enfants. Ensuite, via `@eval(await $slots.main())`, il exécute le code de ses enfants (la boucle `@each` ci-dessus) sans les afficher. Durant cette phase, chaque enfant `map_marker` va peupler la liste de marqueurs.

Code du composant `display_map`

```
1  @let(map = {
2    center,
3    zoom,
4    markers: [],
5  })
6
7  {{-- Share map object with children --}}
8  @inject({ map })
9
10 {{-- Execute children, but do not render them --}}
11 @eval(await $slots.main())
12
13 {{-- Render a div and bind it to an Alpine component --}}
14 <div style="{{ $props.get('style') }}" class="{{ $props.get('class') }}" x-data="map('{{ js.stringify(map) }}'" id="map"></div>
```

- **Le composant enfant `map_marker`** : Son rôle principal est d'agir comme un "fournisseur de données" pour son parent. Lors de la phase `@eval` du parent, chaque `map_marker` pousse ses propres informations (latitude, longitude, etc.) dans le tableau `map.markers` de l'objet partagé. Ce composant ne produit aucun HTML visible lui-même ; il ne sert qu'à configurer les données de la carte.

Code du composant `map_marker`

```
1  {{-- Make sure the marker component is a child of the map component --}}
2  @if(!$context.map)
3    @newError(
4      'The map.marker component should be nested within the map component',
5      $caller.filename,
6      $caller.line,
7      $caller.col
8    )
9  @end
10
11 {{-- Push props as a marker with the map --}}
12 @eval($context.map.markers.push({ latitude, longitude, label, description, date, adress, city}))
13
14 <div x-data="{
15   latitude: {{ latitude }},
16   longitude: {{ longitude }},
17   label: '{{ label }}',
18   description: '{{ description }}',
19   date: '{{ date }}',
20   adress: '{{ adress }}',
21   city: '{{ city }}'
22 }" x-init="addMarker(map, { latitude, longitude, label, description, date })">
23 </div>
```

Rendu et Interactivité (Client-Side) : La dernière étape se passe dans le navigateur.

- La div finale du composant `display_map` est générée.
- `x-data="map('{{ js.stringify(map) }}")` : C'est ici que la magie opère. L'objet `map`, maintenant rempli avec les données de tous les marqueurs, est converti en une chaîne JSON et passé au composant Alpine.js nommé `map`.
- Ce script Alpine.js reçoit alors toutes les données d'un seul coup. Il se charge d'initialiser Leaflet.js avec le bon centre et le bon niveau de zoom, puis il parcourt la liste des marqueurs pour les afficher sur la carte, en y attachant les popups interactives.

En résumé, nous utilisons la puissance du moteur de template EdgeJS pour agréger les données de manière structurée côté serveur, puis nous transmettons ces données à Alpine.js et Leaflet.js qui prennent le relais pour créer une expérience utilisateur riche et interactive côté client.

2.2. Les réalisations côté Back-End

2.2.1. Architecture et fondations Back-End

Le développement du Backend de l'application Ori&Nori a été conçu pour offrir une architecture robuste, sécurisée et maintenable. Les choix technologiques effectués répondent à des critères de cohérence avec notre stack full JavaScript, tout en assurant une courbe d'apprentissage adaptée à notre niveau de maîtrise.

Nous avons choisi AdonisJS, un framework Node.js, largement inspiré du framework PHP Laravel, il en reprend les grands principes, la structure MVC, le système de migration, un ORM intégré, une couche de validation et son système de routing claire. Cette décision s'est imposée rapidement pour mettre en application nos acquis, mais aussi explorer une technologie plus moderne et structurante.

AdonisJS s'est révélé être un choix pertinent pour Ori&Nori : son architecture MVC (Model, View, Controller), Lucid, son ORM performant, ainsi que les nombreux modules prêts à l'emploi pour la validation, l'authentification ou la gestion des rôles. AdonisJS offre de nombreux outils de sécurité, de déploiement, de test. Cela nous a définitivement convaincus.

La structure d'Ori&Nori repose sur l'architecture MVC, elle est utilisée pour organiser le code de façons clair:

Les Modèles: Gestion des données et des relations

Les modèles représentent le cœur d'Ori&Nori, User, Pet, Meetups. Chaque modèle est une classe qui interagit avec la base de données via Lucid par exemple:

- Le modèle User gère les utilisateurs, leur relations et inclut des fonctionnalités comme le hachage de mot de passe.
- Le modèle Species gère les informations liées à chaque espèces et établit la relation avec le modèle Breed pour associer les espèces aux races.

Les Vues : Présente les données

Les vues gèrent l'affichage des données de l'utilisateur, elles sont écrites en Edge.js, le moteur de template d'Adonis. Elles utilisent des composants réutilisables pour afficher les informations en récupérant les données des contrôleurs; par exemple :

- La vue login.edge affiche le formulaire de connexion et gère le message d'erreur ou de succès
- La vue delete_meetup.edge permet à l'utilisateur de confirmer la suppression du meetup

Les Contrôleurs: Gestion des requêtes

Les contrôleurs dirigent les interactions entre modèles et vue. Chaque contrôleur est une classe dédié à un domaine défini; par exemple :

- Le contrôleur AuthController gère l'authentification, l'inscription, la connexion et la déconnexion des utilisateurs.
- Le contrôleur MeetupsController gère les fonctionnalités liées aux meetups, leur création, suppression, mise à jour.

Ils réalisent les opérations sur la base de données en interagissant avec les modèles et renvoient leur résultats aux vues

Pour garantir une structure claire dans le code, nous avons appliqué des règles de nommage précises pour rendre le code plus facilement compréhensible. Le choix d'AdonisJS nous applique aussi par défaut certaines de ses règles notamment pour le nom des classes, des contrôleurs ou des vues. Les fichiers suivent la convention snake_case selon les normes d'AdonisJS; les classes utilisent la convention PascalCase; les méthodes et variables suivent la convention camelCase. Nous avons configuré ESLint pour automatiser le respect des règles de nommage, ainsi que détecter rapidement les erreurs et incohérences dans notre code. Nous avons également configuré un fichier EditorConfig avec l'extension VScode pour assurer une indentation et tabulation uniforme.

Ces choix nous ont permis de professionnaliser l'approche que nous avions, en nous confrontant à des problématiques de sécurité, de maintenabilité et de structuration.

2.2.2. Modélisation et implémentation de la base de données

Ori&Nori repose sur la gestion et l'interconnexion sociale : User, Pet, Meetup, Review. Le choix d'une base de données relationnelle s'est imposé comme le plus pertinent.

Nous avons opté pour PostgreSQL, en raison de sa stabilité, sa conformité aux standards professionnels et sa documentation fournie. Comme pour l'utilisation d'AdonisJS, nous voulions professionnaliser notre approche technique en nous appuyant sur un outil robuste avec de nombreuses fonctionnalités.

La modélisation des données a été directement réalisée sous forme d'un ERD (Entity Relationship Diagram) ou MCD en français, qui nous a servi de modèle conceptuel et logique. Nous avons, par le biais de ce schéma, identifié les entités principales ainsi que leurs relations (one-to-many, many-to-many), en lien avec les besoins métier.

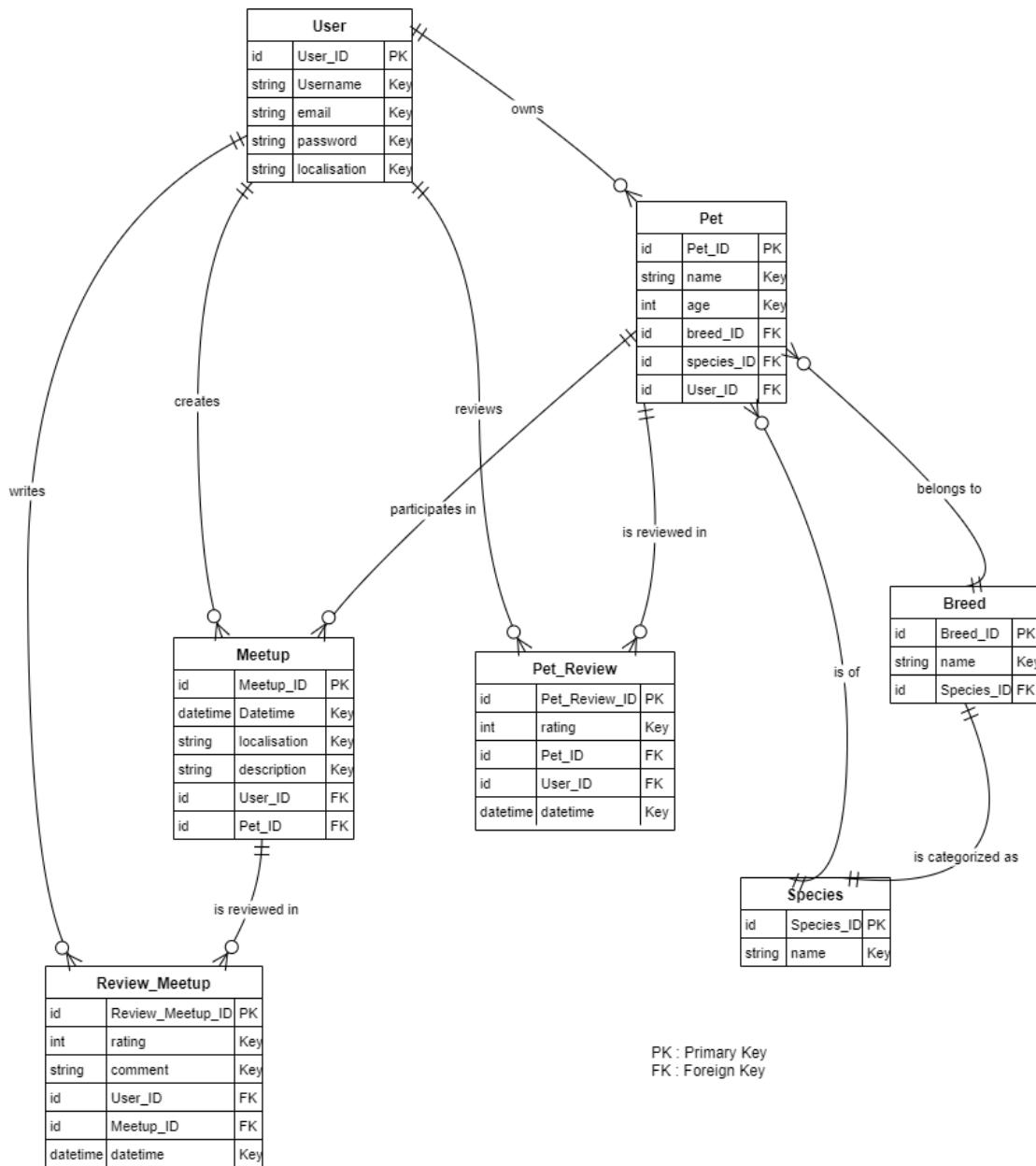


Diagramme ERD illustrant les relations au sein d'Ori&Nori

L'ensemble des tables a ensuite été mis en place via le système de migration d'AdonisJS, permettant de générer la structure de la base de données. Cette méthode permet une collaboration facile et une cohérence dans les données.

```
1 import { BaseSchema } from '@adonisjs/lucid/schema'
2
3 export default class extends BaseSchema {
4   protected tableName = 'user_meetups'
5
6   ▼ async up() {
7     this.schema.createTable(this.tableName, (table) => {
8       table.increments('id')
9
10      // Foreign keys
11      table.integer('user_id').unsigned().references('users.id').notNullable().onDelete('CASCADE')
12      table
13        .integer('meetup_id')
14        .unsigned()
15        .references('meetups.id')
16        .notNullable()
17        .onDelete('CASCADE')
18      table.string('user_name').notNullable()
19      table.string('sort_order').notNullable().defaultTo(0)
20      table.unique(['user_id', 'meetup_id'])
21
22      // timestamps
23      table.timestamp('created_at')
24      table.timestamp('updated_at')
25    })
26  }
27
28  async down() {
29    this.schema.dropTable(this.tableName)
30  }
31 }
```

Migration pour la création de la table user_meetups établissant les relations many-to-many entre user et meetup.

```

1   import { DateTime } from 'luxon'
2   import { BaseModel, column, belongsTo,hasMany } from '@adonisjs/lucid/orm'
3   import type { BelongsTo, HasMany } from '@adonisjs/lucid/types/relations'
4   import Species from './species.js'
5   import Pet from './pet.js'
6
7   export default class Breed extends BaseModel {
8     @column({ isPrimary: true })
9     declare id: number
10
11    @column()
12    declare name: string
13
14    @column()
15    declare speciesId: number
16
17    @belongsTo(() => Species)
18    declare species: BelongsTo<typeof Species>
19
20    @hasMany(() => Pet)
21    declare pet: HasMany<typeof Pet>
22
23    @column.dateTime({ autoCreate: true })
24    declare createdAt: DateTime
25
26    @column.dateTime({ autoCreate: true, autoUpdate: true })
27    declare updatedAt: DateTime
28  }

```

Modèle Breed définissant la structure des races d'animaux et leurs relations.

Pour éviter des pertes de données, nous avons configuré notre base de données PostgreSQL avec un stockage persistant via Docker. Cela assure la conservation des données en cas de redémarrage du serveur.

```

32   pgsql:
33     image: postgres:17
34     restart: always
35     environment:
36       POSTGRES_DB: ${DB_DATABASE}
37       POSTGRES_USER: ${DB_USER}
38       POSTGRES_PASSWORD: ${DB_PASSWORD}
39     volumes:
40       - oriandnori-pgsql:/var/lib/postgresql/data

```

Configuration PostgreSQL dans le docker compose.

2.2.3. Structuration des requêtes et logique d'accès aux données

Dans la version actuelle d'Ori&Nori, la logique métier est directement implémentée dans les contrôleurs (AuthController, UsersController, MeetupsController,...). Cette approche repose sur la structure standard MVC d'AdonisJS. Les modèles gèrent l'accès à la base de données et les contrôleurs assurent les traitements. Les contrôleurs sont organisés par domaine rendant le flux de données clair et direct avec une séparation nette entre modèles et vues. Cette structure suit les conventions d'AdonisJS et répond efficacement à nos besoins actuels et nous donne une bonne lisibilité du code.

Nous avons remarqué la duplication de certaines vérifications, notamment pour les autorisations ou les validations des données, ce qui montre les limites de notre approche si Ori&Nori évoluait en un projet de plus grande ampleur. Pour aller plus loin, ces logiques pourraient être regroupées dans une couche de services métier afin de les réutiliser plus facilement. Au fur et à mesure de l'avancement du projet, une meilleure compréhension de notre propre architecture ainsi que des solutions disponibles nous ont menés à envisager l'ajout d'une couche métier dédiée pour réduire la duplication du code, faciliter la maintenabilité.

```
13     try {
14       const user = auth.user
15       if (!user) {
16         return response.unauthorized({ message: 'User not authenticated' })
17     }
```

Exemple de vérification répétée dans chaque méthode

```
63     if (reviewMeetup.userId !== user.id) {
64       session.flash('error', 'You are not authorized to update this review')
65       return response.redirect().toRoute('displayMeetup', { id: meetupId })
66     }
```

Exemple de vérification répétée dans la méthode reviewMeetup

```
69     //update global rating
70     const meetup = await Meetup.findOrFail(meetupId)
71     await meetup.load('reviewMeetup')
72     await meetup.updateGlobalRating()
73
```

Exemple de logique métier répétée dans reviewMeetup

Notre structure répond pleinement à nos besoins actuels et nous a permis de réaliser dans les délais que nous avions une application fonctionnelle avec un code lisible.

2.2.4. Implémentation de la logique métier dans l'application

La logique métier, comme rappelé dans la section précédente, est principalement intégrée dans les contrôleurs. Nous avons bien compris les règles métier propres à Ori&Nori, et elles ont été appliquées aux contrôleurs. Un exemple essentiel de logique métier est la gestion de l'inscription à un meetup.

Lorsqu'un utilisateur souhaite rejoindre un meetup, plusieurs règles s'appliquent :

- L'utilisateur doit être authentifié et connecté
- Le meetup doit exister et être encore ouvert aux inscriptions
- L'utilisateur ne peut pas s'inscrire deux fois au même meetup
- La date du meetup ne doit pas être dépassée

Ces règles impliquent des vérifications importantes :

- Vérification d'authentification : S'assurer que l'utilisateur est bien connecté
- Validation d'existence : Confirmer que le meetup existe dans la base de données
- Contrôle de date : Vérifier que le meetup n'est pas dans le passé
- Prévention des doublons : S'assurer qu'aucune inscription n'existe déjà pour cet utilisateur

Comme autre exemple représentatif de composant métier, la gestion des avis:

Lorsqu'un utilisateur souhaite laisser un avis, plusieurs règles sont appliquées :

- Il doit avoir participé à cette rencontre
- Il ne peut évaluer qu'une seule fois chaque meetup
- Son évaluation (note + commentaire) sera visible sur le profil des animaux présents, et influencera leur réputation.

Ce traitement repose sur plusieurs vérifications métier :

- Existence de la participation
- Non-duplication de l'avis
- Attribution de la note au bon profil animal

Les règles métier intégrées directement aux contrôleurs témoignent de notre capacité à identifier les logiques propres au fonctionnement Ori&Nori.

2.2.4.1. Gestion des utilisateurs et des animaux

Chaque utilisateur peut créer un compte et gérer son profil.

Un utilisateur peut posséder plusieurs animaux (relation one-to-many).

Seul le propriétaire d'un animal peut modifier ou supprimer son profil (updatePet et deletePet). Les animaux sont catégorisés par espèce et race, avec des attributs comme leur statut vaccinal.

```
24      /**
25      * -----
26      * Create a Pet
27      * -----
28      */
29  ▼  async createPet({ auth, request, response, session }: HttpContext) {
30    try {
31      const validatedData = await request.validateUsing(createPetValidator)
32      const user = auth.user
33      let fileName = ''
34
35      if (!user) {
36        return response.unauthorized({ message: 'User not authenticated' })
37      }
38
39      if (validatedData.photo) {
40        await validatedData.photo.move(app.makePath('storage/uploads'), {
41          name: `${cuid()}.${validatedData.photo.extname}`,
42        })
43
44        if (!validatedData.photo.fileName) {
45          session.flash('error', 'Error uploading profile picture')
46          return response.redirect().back()
47        }
48
49        fileName = validatedData.photo.fileName
50      }
51
52      const vaccinated = request.input('vaccined') === '1'
53      const pet = new Pet()
54
55      pet.merge({
56        ...validatedData,
57        userId: user.id,
58        photo: fileName,
59        vaccinated: vaccinated,
60      })
61
62      await pet.save()
```

Exemple de composant métier et d'accès aux données méthode createPet

2.2.4.2. Organisation des Meetups

Les utilisateurs peuvent créer et planifier des rencontres entre animaux.
Un système de proposition de rencontre par e-mail est intégré (proposeMeetup).
Un filtrage est possible par espèce et race.

```
323
324     async proposeMeetup({ request, session, response, auth }: HttpContext) {
325         const petId = request.param('id')
326         const pet = await Pet.find(petId)
327
328         if (!pet) {
329             session.flash('error', 'Pet not found')
330             return response.redirect().back()
331         }
332
333         const owner = await User.find(pet.userId)
334         if (!owner) {
335             session.flash('error', 'Owner not found')
336             return response.redirect().back()
337         }
338
339         const user = auth.user
340         if (!user) {
341             session.flash('error', 'User not authenticated')
342             return response.redirect().back()
343         }
344         const profileUrl = `http://localhost:3333/users/${user.id}`
345
346         await mail.send(({message}) => {
347             message
348                 .to(owner.email)
349                 .from('no-reply@oriandnori.org')
350                 .subject('Proposition of Meetup')
351                 .htmlView('emails/propose_meetup_to_owner', { owner, user, profileUrl })
352         })
353
354         session.flash('success', 'An email has been sent to the owner')
355         return response.redirect().back()
356     }
357 }
```

Exemple de composant métier et d'accès aux données méthode proposeMeetup

2.2.4.3. Système d'évaluation

Les utilisateurs peuvent évaluer les rencontres et laisser des avis.

Les animaux peuvent recevoir des évaluations, visibles sur leur profil.

Ces évaluations participent à instaurer confiance et réputation au sein de la plateforme.

```
7      /**
8       * -----
9       * Create Review Meetup
10      * -----
11     */
12    async createReviewMeetup({ auth, request, response, session }: HttpContext) {
13      try {
14        const user = auth.user
15        if (!user) {
16          return response.unauthorized({ message: 'User not authenticated' })
17        }
18
19        const validatedData = await request.validateUsing(reviewMeetupValidator)
20        const meetupId: number = request.input('meetupId')
21
22        await ReviewMeetup.create({
23          ...validatedData,
24          userId: user.id,
25          meetupId: meetupId,
26        })
27
28        //update global rating
29        const meetup = await Meetup.findOrFail(meetupId)
30        await meetup.load('reviewMeetup')
31        await meetup.updateGlobalRating()
32
33        session.flash('success', 'Review created successfully')
34        return response.redirect().back()
35      } catch (error) {
36        const meetupId = request.input('meetupId')
37        session.flash('error', 'Review not found')
38        return response.redirect().toRoute('displayMeetup', { id: meetupId })
39      }
40    }
41  }
```

Code de la création d'un avis sur un Meetup

2.2.4.4. Règles de sécurité et d'accès

Vérification stricte que les utilisateurs ne peuvent modifier que leurs propres données.
Contrôle d'authentification pour toutes les actions sensibles.

Validation des données utilisateur pour garantir l'intégrité des informations saisies.

```
287 	async deletePet({ auth, response, params, session }: HttpContext) {
288  	const user = auth.user
289  	if (!user) {
290    	session.flash('error', 'You must be logged in to view this page')
291    	return response.redirect().toRoute('auth.login')
292   }
293  	const pet = await Pet.find(params.id)
294  	if (!pet) {
295    	session.flash('error', 'Pet not found')
296    	return response.redirect().toRoute('MyPets')
297   }
298
299  	if (pet.userId !== user.id) {
300    	session.flash('error', 'You are not authorized to update this pet')
301    	return response.redirect().toRoute('MyPets')
302   }
303
304  	if (pet.photo) {
305    	try {
306      	await drive.use().delete(`uploads/${pet.photo}`)
307     } catch (error) {
308      	session.flash('error', 'Error deleting profile picture')
309      	return response.redirect().back()
310     }
311   }
312
313  	await pet.delete()
314  	session.flash('success', 'Pet deleted successfully!')
315  	return response.redirect().toRoute('MyPets')
316 }
317 }
```

Exemple de composant métier et d'accès aux données méthode deletePet

2.2.4.5. Upload et suppression de médias associés

Upload et stockage des photos d'animaux lors de leur création ou mise à jour.
Suppression des fichiers associés lors de la suppression d'un profil animal.

Ces règles métier répondent à l'objectif principal d'Ori&Nori : faciliter les connexions sociales entre propriétaires d'animaux afin de favoriser la socialisation de leurs compagnons.

```
202
203     const pet = await Pet.find(params.id)
204     if (!pet) {
205         session.flash('error', 'Pet not found')
206         return response.redirect().toRoute('MyPets')
207     }
208
209     if (pet.userId !== auth.user.id) {
210         session.flash('error', 'You are not authorized to update this pet')
211         return response.redirect().toRoute('MyPets')
212     }
213
214     const updatePetData = await request.validateUsing(createPetValidator)
215     let fileName = ''
216
217     if (updatePetData.photo) {
218         if (pet.photo) {
219             try {
220                 await drive.use().delete(`uploads/${pet.photo}`)
221             } catch (error) {
222                 session.flash('error', 'Error deleting old photo')
223                 return response.redirect().back()
224             }
225         }
226
227         await updatePetData.photo.move(app.makePath('storage/uploads'), {
228             name: `${cuid()}.${updatePetData.photo.extname}`,
229         })
230
231         if (!updatePetData.photo.fileName) {
232             session.flash('error', 'Error uploading profile picture')
233             return response.redirect().back()
234         }
235
236         fileName = updatePetData.photo.fileName
237     }
238     //console.log('vaccined', request.input('vaccined'))
239     const vaccinated = request.input('vaccined') === '1'
240     //console.log('vaccined', vaccinated)
```

Partie de la fonction updatePet gestion de la photo

2.2.5. Sécurité applicative côté back-end

La sécurité des données et des accès est un enjeu pour toute application moderne ; elle l'est aussi pour Ori&Nori. Plusieurs mesures ont été mises en place pour garantir l'intégrité et la confidentialité, principalement grâce aux fonctionnalités intégrées à AdonisJS.

Nous avons mis en place un système d'authentification basé sur des sessions avec cookies, protégé par des middlewares pour restreindre l'accès aux routes sensibles. Des vérifications d'autorisation sont réalisées pour garantir qu'un utilisateur ne puisse gérer que ses données.

Lors de l'enregistrement, les mots de passe des utilisateurs sont stockés de manière sécurisée dans la base de données après avoir été hachés grâce au services de hachage d'AdonisJS. Les mot de passe ne sont pas enregistrés en clair mais haché, la vérification se fait en comparant le mot de passe saisie avec le hash stocké.

```
12  const AuthFinder = withAuthFinder(() => hash.use('scrypt'), {  
13    uids: ['email'],  
14    passwordColumnName: 'password',  
15  })
```

Hachage du mot de passe

	123 ➔ id	A-Z username	A-Z email	A-Z password
1	1	Mezanic9	Mezanic9@mail.com	\$scrypt\$n=16384,r=8,p=1\$Bmy3l/QVYTrIHs4JqhMkxA\$YX3/s09Gb+QgjxvER4255Ut5ovLhr
2	2	Mezanic	Mezanic@mail.com	\$scrypt\$n=16384,r=8,p=1\$7yMaTOEOsqD2BRmbD3twaQSIW4Uju8hjWPMip0ZeGAMYRCJ
3	3	Mezanic22	Mezanic22@mail.com	\$scrypt\$n=16384,r=8,p=1\$xrElOijUMGT4RJPMPvRQ\$HMePyujxs59fQbN9DFQCH0Fbl
4	4	User33	user3@mail.com	\$scrypt\$n=16384,r=8,p=1\$d90q+BFx82Qhu9MPs+G1A\$sPM7m3fGYw0vcVV8AfH3sU5nm
5	5	Mezanic10	Mezanic10@mail.com	\$scrypt\$n=16384,r=8,p=1\$M3h+Oha0FVKRsPnllqL8GA\$wyQt6tLnspz/OWICxRit3cnizR9ql
6	6	Mezanic11	Mezanic11@mail.com	\$scrypt\$n=16384,r=8,p=1\$frTzmjs6ZlscPCEIO/khw\$R/i7SQG9gFeovZPs6qUwvZdeIAh2wz4
7	7	Mezanic99	Mezanic99@mail.com	\$scrypt\$n=16384,r=8,p=1\$jyaAqK+2uOKx6ErWEML/7g\$oCtsxwtb0Y8Vgm8Sc9gcd+mQ+4k

Stockage des mots de passe dans la base de données

En cas d'oubli, un système de réinitialisation de mot de passe est mis en place. Les utilisateurs peuvent demander un lien de réinitialisation envoyé par email, généré à l'aide d'un jeton unique et temporaire, comme défini dans le modèle `resetPassword`. Ce lien permet de définir un nouveau mot de passe en toute sécurité.

```

81  ↘    async handleResetPassword({ request, session, response }: HttpContext) {
82      const { token, email, password } = await request.validateUsing(resetPasswordValidator)
83      const passwordResetToken = await ResetPassword.findBy('token', token)
84
85      if (
86          !passwordResetToken ||
87          !!passwordResetToken.isUsed === true ||
88          passwordResetToken.email !== email ||
89          passwordResetToken.expiresAt < DateTime.now()
90      ) {
91          session.flash('error', 'Link expired or invalid')
92          return response.redirect().toRoute('auth.forgot_password')
93      }
94
95      const user = await User.findBy('email', email)
96      if (!user) {
97          session.flash('error', 'User not found')
98          return response.redirect().toRoute('auth.forgot_password')
99      }
100     await passwordResetToken.merge({ isUsed: true }).save()
101     await user.merge({ password }).save()
102
103     await mail.send((message) => {
104         message
105             .to(user.email)
106             .from('no-reply@oriandnori.org')
107             .subject('Reset Password')
108             .htmlView('emails/reset_password', { user })
109     })
110     session.flash('success', 'Password reset successfully')
111     return response.redirect().toRoute('auth.login')
112 }
113 }
```

Méthode handleResetPassword en cas d'oubli de mot de passe.

Toute opération sur la base de données passent par l'ORM Lucid natif à AdonisJS, ce qui assure une sécurisation des requêtes et empêchent les injections SQL. La validation des données est assurée avec des validateurs natifs à AdonisJS, pour garantir l'intégrité des données.

Les limites de cette approche, ainsi que les vulnérabilités identifiées et les pistes d'amélioration pour l'ensemble du projet, côté Front-end et Back-end, sont détaillées dans la partie 3 à propos de la veille sur les vulnérabilités et la sécurité.

2.2.6. Résolutions structuré des problèmes

Chaque dysfonctionnement a été traité de manière structurée. Dès qu'une erreur survenait, je commençais par examiner le message d'erreur affiché sur la page ou dans la console. Si cela ne suffisait pas, j'examinais les logs Docker et l'état des conteneurs pour localiser l'origine du problème au niveau des conteneurs et de la base de données. Si je ne trouvais pas d'erreur, je poursuivais mes vérifications dans le code, en ajoutant des vérifications pour trouver l'origine du problème.

Pour comprendre le problèmes, je reproduisais le scénario plusieurs fois en testant différentes entrées ou en ajoutant temporairement des lignes de vérification dans le code afin de contrôler la validité des données

Une fois le problème bien isolé et documenté grâce aux logs, je cherchais une solution appropriée; cela pouvait passer par la correction de contrôleur, la modification d'un type de données, d'un ajustement dans le formatage ou la mise à jour d'un validateur.

Après chaque correction, je testais de nouveau le scénario initial puis l'ensemble des scénarios qui pouvait reproduire une erreur similaire ou continuer d'altérer les données. Cette vérification plus globale autour de mon problème me garantissait généralement un correctif robuste de mon problème.

Enfin j'informais systématiquement mon binôme de l'origine du problème ainsi que la solution mise en place. Lorsqu'un bug touchait une partie de code en dehors de mon travail j'en informais mon collègue, lui proposant mon aide. En cas de problème persistant nous mutualisons nos connaissances, recherche et piste de réflexion pour trouver le meilleur correctif possible.

2.3. Jeu d'Essai et validation des fonctionnalités

2.3.1. Approche de Test

Pour Ori&Nori, l'ensemble des tests a été réalisé manuellement. Les tests manuels ont constitué une part importante de notre temps de développement. N'étant pas à l'aise avec les outils de test automatisé tels que Jest ou Japa (intégré à AdonisJS), nous avons décidé de tester manuellement l'ensemble de nos fonctions. Cette approche nous a semblé la plus pertinente, compte tenu du volume important de nouvelles connaissances à acquérir avec les outils que nous avons choisis. Notre vision future, si le projet Ori&Nori devait évoluer, serait d'intégrer des tests unitaires, fonctionnels et d'intégration.

Afin de réaliser nos tests, nous avons créé un jeu de données varié. Les espèces et races d'animaux sont déjà implantées dans l'application. Nous avons aussi créé de nombreux

utilisateurs, certains possédant un ou plusieurs animaux, d'autres non. Selon les scénarios à tester, nous avons également préparé des meetups, des utilisateurs et des animaux préexistants.

2.3.2. Fonctionnalité Représentative : Gestion des Meetups

La fonctionnalité choisie pour illustrer notre jeu d'essai est la gestion des meetups. Cette fonctionnalité est centrale dans le projet Ori&Nori. Elle inclut plusieurs sous-fonctionnalités, telles que la création de meetups, l'ajout de participants et la géolocalisation des événements.

2.3.3. Scénarios de test et résultats

Pour tester cette fonctionnalité, plusieurs scénarios ont été définis avec des données d'entrée variées :

Scénario 1 : Création d'un meetup valide (Annexe 10 en p.60 et p.61)

- **Données en entrée :**
 - Titre : "Corginade"
 - Date : "19/06/2025 16:00"
 - Adresse : "Prairie des filtres, Toulouse"
 - Animaux sélectionnés : Ori, Nori
- **Résultat attendu :**
 - Le meetup est enregistré en base de données avec les informations fournies.
 - Les animaux sélectionnés sont associés au meetup.
 - L'utilisateur est ajouté comme organisateur.
 - Le Meetup est correctement géolocalisé.
- **Résultat obtenu :**
 - Le meetup est enregistré en base de données avec les informations fournies.
 - Les animaux sélectionnés sont associés au meetup.
 - L'utilisateur est ajouté comme organisateur.
 - Le Meetup est correctement géolocalisé.

Scénario 2 : Tentative de création d'un meetup sans animaux sélectionnés (Annexe 11 en p. 67)

- **Données en entrée :**
 - Titre : "Corginade au lac"
 - Date : "20/06/2025 16:00"
 - Adresse : "Prairie des filtres, Toulouse"
 - Animaux sélectionnés : aucun
- **Résultat attendu :**
 - Message d'erreur affiché : "You must select at least one pet for the meetup."
 - Redirection vers le formulaire de création.

- **Résultat obtenu :**
 - Message d'erreur affiché : "You must select at least one pet for the meetup."
 - Redirection vers le formulaire de création.

Scénario 3 : Rejoindre un meetup existant (Annexe 12 en p.68 et p.69)

- **Données en entrée :**
 - Meetup : Corginade
 - Animaux sélectionnés : Marin
 - Utilisateur connecté : User33
- **Résultat attendu :**
 - Confirmation affichée : "You have successfully joined the meetup."
 - L'utilisateur et ses animaux sont ajoutés au meetup.
- **Résultat obtenu :**
 - Confirmation affichée : "You have successfully joined the meetup."
 - L'utilisateur et ses animaux sont ajoutés au meetup.

Les tests manuels ont permis de valider la fonctionnalité principale de gestion des meetups dans Ori&Nori.

2.3.4. Analyse des écarts et pistes d'amélioration

Bien que les résultats aient été globalement satisfaisants durant les tests effectués tout au long du développement d'Ori&Nori, certains écarts ont mis en évidence des axes d'amélioration, notamment en termes de validation des données pour l'API OpenCage (adresse mal orthographiée ou non prise en compte) et de gestion des erreurs. Ces enseignements serviront de base pour la mise en place de tests dans les futures itérations du projet.

Partie 3 : Veille sur les vulnérabilités de sécurité et identification des failles potentielles

La sécurité des données et des accès est une composante importante d'une application, et Ori&Nori n'échappe pas à cette règle. Même si la sécurité n'a pas été un axe structurant dès les premières phases du projet, plusieurs mesures ont été mises en place de manière progressive, souvent en s'appuyant sur les fonctionnalités natives proposées par AdonisJS. Ainsi, des pratiques sécurisantes comme l'utilisation de l'ORM Lucid, des middlewares d'authentification , ou encore la protection CSRF ont été intégrées par défaut à notre stack.

Même sans être des experts en cybersécurité au début du projet, l'utilisation du framework a mis en place une base de sécurité solide. Cependant, en développant l'application, nous avons vite compris à quel point la sécurité était cruciale. Cela nous a poussés à nous informer au fur et à mesure. Par exemple, nous avons appris comment protéger les mots de passe des utilisateurs ou comment sécuriser les échanges entre le navigateur et le serveur pour éviter que des informations sensibles ne soient interceptées. Ce processus de veille nous a permis d'intégrer progressivement des pratiques de sécurité, comme la validation des entrées pour empêcher les injections SQL et XSS, afin de mieux protéger les données de nos utilisateurs et l'accès à notre application.

3.1. Démarche de veille

Pendant le développement du projet, la veille sur les aspects de sécurité s'est faite de manière ponctuelle et orientée par les outils fournis par AdonisJS. Les différents aspects liés à la sécurité ont été vus au fur et à mesure que le projet avançait, suivant les fonctionnalités et les besoins techniques. Notamment lors de la mise en place de l'authentification, de la gestion des sessions et de la validation des entrées utilisateur.

Les principales ressources consultées ont été la documentation officielle d'AdonisJS, notamment sur l'utilisation de l'ORM Lucid, des middlewares d'authentification et de la protection CSRF. Nous avons également pris connaissance du guide OWASP Top 10 afin de mieux comprendre les principales failles de sécurité web. Un pentest devra être réalisé pour évaluer concrètement la résistance de notre application à ces vulnérabilités.

3.1. Suivi technologique

Un suivi régulier de notre environnement est nécessaire pour assurer la sécurité, la stabilité et la continuité d'Ori&Nori. Cela implique de surveiller les mises à jour du framework AdonisJS, de Node.js, des dépendances principales (Traefik, Docker, PostgreSQL). Ce suivi nous permet d'appliquer rapidement les correctifs de sécurité liés, de bénéficier des optimisations de performance et d'anticiper les changements majeurs.

3.2. Vulnérabilités identifiées dans Ori&Nori

Voici les principales vulnérabilités identifiées ou anticipées dans notre projet :

- **Injection SQL** : Cette vulnérabilité est écartée grâce à l'utilisation de l'ORM Lucid, qui empêche la construction de requêtes SQL manuelles et limite ainsi les attaques par injection.
- **Cross-Site Scripting (XSS)** : L'utilisation d'AdonisJS avec EdgeJS nous protège contre les balises HTML ou Javascript qui sont insérées par un utilisateur, celle-ci ne s'exécute pas et son échappe automatiquement. De plus, l'utilisation de validateurs nous permet de contrôler les données entrées. Cependant, aucune sanitization n'est faite, il est important de noter que ce code est quand même enregistré dans notre base de données.(Annexe 14 en p.70)
- **Accès non autorisé aux ressources** : Nous avons sécurisé l'accès aux ressources via des contrôles manuels dans les contrôleurs (ex. : un utilisateur ne peut modifier que ses propres animaux).
- **Absence de politique de mot de passe fort** : Aucune exigence spécifique n'est imposée lors de la création d'un compte, ce qui laisse la possibilité aux utilisateurs de choisir des mots de passe faibles.
- **Attaques par force brute** : Aucune mesure n'est en place pour limiter le nombre de tentatives de connexion (ex. : captcha, délai d'attente, blocage temporaire).

3.3. Contre-mesures en place

Malgré un temps limité, plusieurs bonnes pratiques ont été mises en place pour renforcer la sécurité de l'application Ori&Nori :

- **Validation systématique des données** : Utilisation des validateurs natifs d'AdonisJS (par exemple createPetValidator) pour garantir l'intégrité des données saisies.

Validateur pour modifier ses information user

```
55  export const updateUserValidator = vine.compile(  
56    vine.object({  
57      first_name: vine.string().trim().minLength(2).optional(),  
58      last_name: vine.string().trim().minLength(2).optional(),  
59      address_1: vine.string().trim().minLength(5).optional(),  
60      address_2: vine.string().trim().minLength(5).optional(),  
61      postal_code: vine.string().trim().minLength(2).optional(),  
62      city: vine.string().trim().alpha().minLength(2).optional(),  
63      phone: vine.string().trim().minLength(10).optional(),  
64      description: vine.string().trim().minLength(10).maxLength(150).optional(),  
65      profile_picture: vine  
66        .file({  
67          size: '4mb',  
68          extnames: ['jpg', 'png', 'jpeg'],  
69        })  
70        .optional(),  
71    })  
72  )
```

```

134     /**
135      * -----
136      * Update my user's profile
137      * -----
138     */
139     async updateMyProfile({ request, auth, session, response }: HttpContext) {
140         if (!auth.user) {
141             session.flash('error', 'You must be logged in to view this page')
142             return response.redirect().toRoute('auth.login')
143         }
144
145         const updateUser = await request.validateUsing(updateUserValidator)
146         let fileName = ''
147

```

Utilisation du validateur pour vérifier les données à la modification des informations utilisateur

- **Utilisation de l'ORM Lucid** : Toutes les opérations sur la base de données passent par Lucid, ce qui supprime le risque d'injections SQL liées à des requêtes manuelles.

```

215
216     const meetup = await Meetup.findOrFail(meetupId)
217
218     const reviewMeetup = await meetup
219         .related('reviewMeetup')
220         .query()
221         .where('userId', user.id)
222         .firstOrFail()
223

```

Accès aux données d'un Meetup et de ses avis via Lucid ORM

- **Contrôles d'accès dans les contrôleurs** : Vérification explicite de l'authentification et de l'autorisation pour chaque ressource critique (profil utilisateur, animaux, inscriptions à des meetups).

```

209     if (pet.userId !== auth.user.id) {
210         session.flash('error', 'You are not authorized to update this pet')
211         return response.redirect().toRoute('MyPets')
212     }
213

```

Autorisation pour qu'un utilisateur ne peut modifier que ses propres animaux

- **Middlewares d'authentification** : Mise en place de middlewares pour protéger les routes sensibles et restreindre l'accès aux utilisateurs connectés.

```

137 // My Pets
138
139 router
140   .group(() => {
141     router.get('/my-pets', [UsersController, 'listMyPet']).as('MyPets')
142     router.put('/my-pets/updatePet/:id', [UsersController, 'updatePet']).as('updatePet')
143     router.get('/my-pets/updatePet/:id', [UsersController, 'updatePetView']).as('updatePetview')
144     router.delete('/my-pets/delete/:id', [UsersController, 'deletePet']).as('deletePet')
145     router.get('/my-pets/deletePetView/:id', [UsersController, 'deletePetView']).as('deletePetview')
146   })
147   .prefix('pets')
148   .use(middleware.auth())
149
150 // Display All Pets Profiles
151 router.get('/pets', [UsersController, 'displayPetList']).as('petList').use(middleware.auth())
152
153 // Display Pet Profile by ID
154 router
155   .get('/pets/:id', [UsersController, 'displayPetProfile'])
156   .as('PetProfile')
157   .use(middleware.auth())
158
159 // Propose a Meetup
160 router
161   .post('/pets/:id/meetup', [UsersController, 'proposeMeetup'])
162   .as('proposeMeetup')
163   .use(middleware.auth())
164

```

Exemple de routes avec utilisation de middleware

3.4. Pistes d'amélioration et suites possibles

Conscients des limites actuelles, voici les actions envisagées pour renforcer la sécurité si Ori&Nori devait évoluer vers une mise en ligne publique :

- Mettre en place une politique de mot de passe fort, avec des contraintes sur la longueur et la complexité.
- Mettre en place une sanitization automatique avant l'enregistrement des données afin de supprimer les balise indésirable et de garantir les données stockées.
- Intégrer une couche de protection contre les attaques par force brute, par exemple en limitant les tentatives de connexion.
- Effectuer un audit de sécurité afin de tester l'application contre un panel d'attaques courantes.

Ces pistes d'amélioration guideront nos choix pour toute version future d'Ori&Nori.

Partie 4 : Documenter le déploiement d'une application dynamique Web ou Web Mobile

Le déploiement de l'application Ori&Nori a été pensé pour garantir une mise en production en s'appuyant sur la conteneurisation. Cette section détaille les choix techniques et la démarche adoptée pour rendre l'application accessible publiquement.

4.1. Choix de l'Environnement d'Hébergement et de la Stratégie de Déploiement

Pour héberger Ori&Nori, nous avons choisi Scaleway. Ce choix s'explique par la flexibilité de ses services, son rapport qualité-prix compétitif pour un projet étudiant et sa facilité d'utilisation, même pour une offre amenée à évoluer. Scaleway propose également un service de mailing, ce qui nous permet de centraliser notre hébergeur et l'envoi des mails transactionnels. Enfin, le fait que Scaleway soit une entreprise française car cela permet de garder les données de nos utilisateurs en France et de nous assurer que Scaleway respecte bien les règles européennes sur la protection des données, comme le RGPD.

Le déploiement de l'application Ori&Nori repose sur une architecture conteneurisée. Nous nous sommes appuyés sur la documentation d'AdonisJS, que nous avons adaptée à l'utilisation de Dockerfile et Docker Compose. Pour diriger notre trafic, nous avons choisi Traefik comme reverse proxy. Traefik a été sélectionné pour sa capacité à gérer les certificats SSL automatiquement, à router les requêtes HTTP/HTTPS vers le bon conteneur et à offrir une configuration claire grâce aux labels Docker. Ces choix permettent à Ori&Nori d'être déployé rapidement tout en assurant une connexion sécurisée.

4.2. Construction et orchestration des conteneurs

Le déploiement s'articule autour de la construction d'images Docker et de leur orchestration :

4.2.1. Construction des conteneurs (Dockerfile)

Le fichier Dockerfile définit plusieurs étapes pour construire l'application: voir en **Annexe 15** en p.71)

- Une étape avec l'image officielle Node:24 pour l'environnement de base.
- Une étape pour installer les dépendances avec npm ci.
- Une étape de build où le framework AdonisJS compile le code source dans un dossier de production.
- Une étape finale pour exécuter l'application en mode production.

4.2.2. Orchestration avec Docker Compose

Le fichier docker-compose.prod.yml orchestre les différents services nécessaires au fonctionnement de l'application en production: voir en **Annexe 16** en p.72.

- **Traefik** : Un reverse proxy configuré pour gérer les certificats SSL/TLS via Let's Encrypt et rediriger les requêtes vers le service Node.js.
- **Node.js** : Le service principal qui exécute l'application Ori&Nori.
- **PostgreSQL** : Notre base de données relationnelle pour stocker les données des utilisateurs, des animaux et des événements.

Les services sont connectés via des réseaux Docker (un réseau Traefik pour la communication externe et un réseau internal pour les communications internes entre conteneurs) pour une communication sécurisée.

4.3. Gestion des configurations et sécurité du déploiement

Plusieurs aspects ont été pris en compte pour sécuriser et optimiser le déploiement :

- **Configuration des variables d'environnement** : Les variables sensibles, comme les identifiants de la base de données ou les clés d'API, sont définies dans un fichier .env (ignoré par Git grâce au fichier .gitignore) pour des raisons de sécurité.
- **Gestion des certificats SSL/TLS** : Traefik est configuré pour utiliser Let's Encrypt et générer automatiquement des certificats SSL/TLS, garantissant une connexion HTTPS sécurisée pour les utilisateurs.
- **Volumes persistants** : Les données de PostgreSQL sont stockées dans un volume Docker nommé oriandnori-pgsql pour garantir leur persistance même après un redémarrage des conteneurs.

4.4. Résultats du déploiement

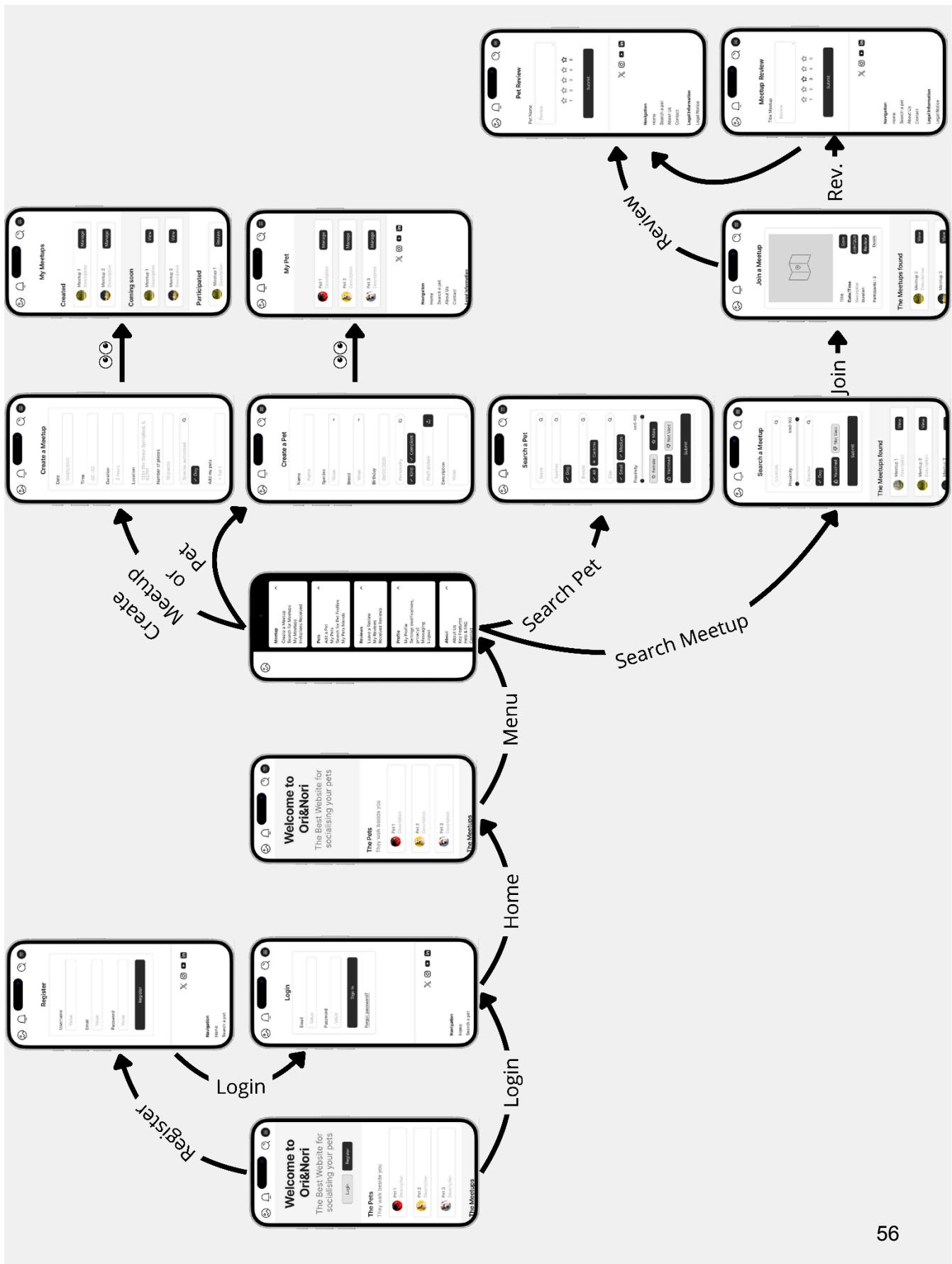
Une fois déployée, l'application est accessible via une URL sécurisée (par exemple, <https://oriandnori.com>), avec une infrastructure prête à gérer les utilisateurs et à évoluer en fonction des besoins futurs. Ce déploiement conteneurisé assure une reproductibilité de l'environnement de production et une facilité de maintenance.

ANNEXE

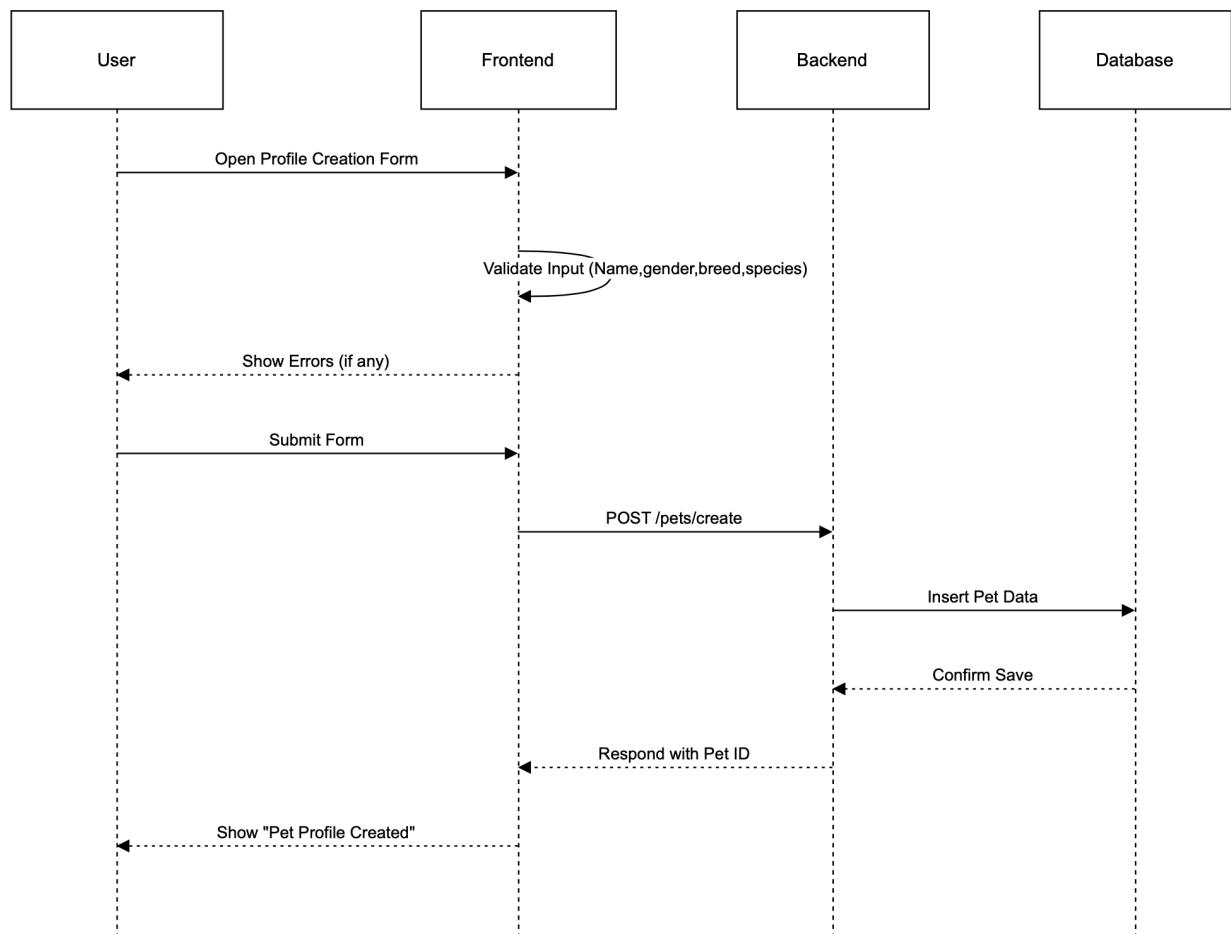
ANNEXE 1 : User Stories

CLASSES	Nom de la feature	En tant que <qui>	Je veux <quoi>	Afin de <pourquoi>	URL	Methode HTTP
USER	Création de compte utilisateur	Visiteur	Créer un compte avec mon email et un mot de passe	Accéder à la plateforme et utiliser ses fonctionnalités	/register	POST
	Connexion utilisateur	Utilisateur	Me connecter à mon compte avec mes identifiants	Gérer mes informations et accéder à mes données personnelles	/login	POST
	Déconnexion utilisateur	Utilisateur connecté	Me déconnecter de mon compte	Protéger mes informations personnelles	/logout	POST
	Réinitialisation de mot de passe	Utilisateur	Réinitialiser mon mot de passe via un lien reçu par email	Pouvoir accéder à mon compte en cas d'oubli	/password-reset	POST
	Affiche le profil d'un utilisateur	Utilisateur connecté	Voir le profil d'un utilisateur	Voir ses Meetups, ses Pets	/<ID>/profile	GET
	Gestion du profil	Utilisateur connecté	Modifier mes informations personnelles	Mettre mon profil à jour	/profile	GET, PUT, DELETE
	Téléchargement de photo de profil	Utilisateur connecté	Ajouter une photo à mon profil	Rendre mon profil plus engageant	/profile/photo	POST
PET	Création d'un Pet	Propriétaire d'un Pet	Ajouter mon Pet avec ses détails (personality, breed, species)	Rendre visible mon Pet pour les autres utilisateurs	/pets/create	POST
	Affiche la fiche d'un Pet	Utilisateur	Afficher la fiche d'un Pet (personality, breed, species, nombre de rencontre, la liste et le nombre de ses amis)	Lire la fiche d'un animal	/pets/<ID>	GET
	Ajout de photos pour le Pet	Propriétaire d'un Pet	Ajouter des photos pour la fiche de mon animal	Rendre la fiche plus attrayante et informative	/pets/<ID>/photo	POST
	Supprimer un Pet	Propriétaire d'un Pet	Supprimer un Pet ainsi que toutes ses détails	Supprimer ce Pet qui n'est plus accessible	/pets/<ID>/delete	DELETE
	Consultation des fiches Pet	Utilisateur	Voir les fiches des autres animaux aux environs	Voir la liste des Pets	/pets	GET
	Filtrage/Recherche avancée Pet	Utilisateur	Filtrer les Pets selon mes critères (âge, espèces, race, lieux ...)	Trouver un animal avec qui mon animal pourrait socialiser	/pets/search	GET
	Accepte demande d'amis	Utilisateur	Accepter une proposition d'amis	Pouvoir choisir si amis ou pas	/pets/<ID>/personality/accept	POST
	Proposer un Meetup à un animal	Utilisateur connecté	Proposer à un autre Pet une sortie	Sortir avec un animal de mon choix	/pets/<ID>/meetup	POST
	Gérer mes Pets	Propriétaire d'un Pet	Voir, éditer, ou supprimer mes Pets	Gérer tout mes animaux enregistrés	/pets/my-pets	GET, PUT, DELETE
	Système de notation pour les animaux et émettre un commentaire	Utilisateur	Donner une note à un Pet, et laisser mon commentaire sur l'animal (à voir si on peut faire la notation hors meetup)	Partager mon expérience pour aider les autres dans leur choix de meetup	/pets/<ID>/review	POST
MEETUP	Créer un Meetup	Utilisateur connecté	Créer une proposition de meetup	Rencontrer des animaux, de sociabilité notre animal	/meetups/create	POST
	Lister les Meetup	Utilisateur connecté	Lister tous les Meetup créer	Voir la liste des Meetup	/meetups	GET
	Filter les Meetup	Utilisateur connecté	Filtrer les Meetup selon mes critères (lieux, pets, durée, etc...)	Trouver un Meetup qui nous convient	/meetups/search	GET
	Rejoindre un Meetup	Utilisateur connecté	M'inscrire à un Meetup	Participer au Meetup et prévenir de mon adhésion à l'organisateur	/meetups/join	POST
	Se désinscrire du Meetup	Utilisateur connecté	Me désinscrire d'un Meetup	Ne plus participer et avertir l'organisateur du Meetup	/meetups/quit	POST
	Gérer mes Meetup	Utilisateur connecté	Gérer mes Meetup	D'organiser mes Meetup, ceux auxquels je participe	/meetups/manage	GET, PUT, DELETE
	Affiche un Meetup	Utilisateur connecté	Afficher le Meetup	Connaitre les caractéristiques du Meetup	/meetups/<ID>	GET
	Affiche les Pets d'un Meetup	Utilisateur connecté	Affiche la liste des Pets participants aux Meetup	Connaitre la liste des animaux d'un Meetup, et pouvoir accéder à la fiche de l'un d'entre eux	/meetups/<ID>/pets	GET
	Noter et commenter un Meetup	Utilisateur connecté	Donner une note à un Meetup et la commenter	Partager mon expérience pour aider les autres	/meetups/<ID>/review	POST
	Noter et commenter un animal par rapport à un Meetup	Utilisateur connecté	Donner une note à un pet du Meetup et la commenter	Partager mon expérience pour aider les autres	/meetups/<ID>/pets/<ID>/review	POST
REVIEW (Pet ou Meetup)	Affiche la liste des review	Utilisateur connecté	Lister les Reviews		/reviews	GET
	Gérer mes Review	Utilisateur connecté	Gérer mes review	Gérer mes reviews	/reviews/my-reviews	GET, PUT, DELETE
	Affiche une review	Utilisateur connecté	Affiche une review	Voir la review	/reviews/<ID>	GET
	Editer une Review	Utilisateur connecté	Modifier ma review	Changer mon avis et ma note	/reviews/<ID>/edit	PUT
	Supprimer une Review	Utilisateur connecté	Supprimer ma review	Supprimer mon avis	/reviews/<ID>/delete	DELETE

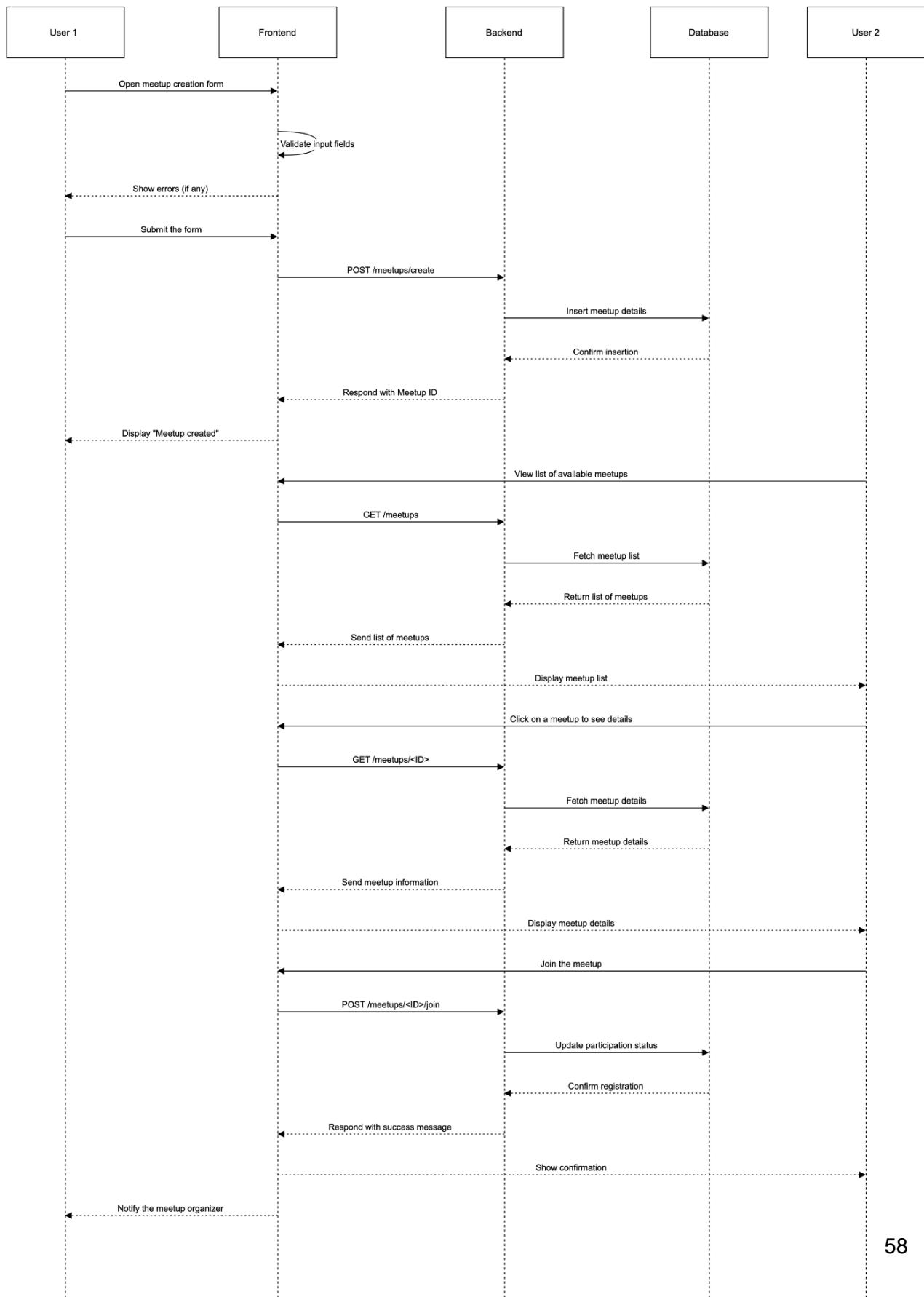
ANNEXE 2 : Userflow



ANNEXE 3 : Diagramme de séquence pour la création d'un profil “Pet”



ANNEXE 4 : Diagramme de séquence pour la planification d'un “Meetup”



ANNEXE 5 : Affichage adaptatif / responsive des “Meetups”

Taille personnalisée ▾ 1600 x 900 (100%) | x 3 ▾ | Ouvrir avec un simulateur ▾

ThomasMaye

Home About Contact Meetups Pets Profile

All Meetups

Here is the list of all meetups you can join.
Click on a meetup to view more details and join it.

Rencontre entre chats à la maison.
Je vous invite à partager un moment à la maison pour une socialisation de nos animaux.

+ by MaelEzanic
20/07/2025 16:00
Rue du chapeau Rouge - Toulouse

3 Pets have joined this Meetup:
European Chat

Socialisation de chat à la maison
Une socialisation à la maison pour voir si Sushi aime les autres chats.

+ by MarineLetalon
22/07/2025 10:00
Avenue Maréchal Foch - Toulouse

3 Pets have joined this Meetup:
European Chat

Ballade avec Loki
Une ballade tranquille autour du lac de la Ramée avec Loki.

+ by FlorenciaDesusa
27/07/2025 16:00
Lac de la Ramée - Tournefeuille

2 Pets have joined this Meetup:
Golden Retriever Border Collie Chien

View

Taille personnalisée ▾ 480 x 900 (100%) | x 3 ▾ | Ouvrir avec un simulateur ▾

ThomasMaye

All Meetups

Here is the list of all meetups you can join.
Click on a meetup to view more details and join it.

Rencontre entre chats à la maison.
Je vous invite à partager un moment à la maison pour une socialisation de nos animaux.

+ by MaelEzanic
20/07/2025 16:00
Rue du chapeau Rouge - Toulouse

ANNEXE 6 : Affichage adaptatif / responsive des “Pets”

Taille personnalisée ▾ 768 x 900 (100 %) | x 3 ▾ | Ouvrir avec un simulateur ▾

Pets List

Here you can see all the pets that are registered in the system.
You can view his pet's profile, and on the profile, you can propose him for a meetup, if you are interested by him.
You can also see the owner's profile and contact him.

Select an option ▾ Select an option ▾ Filter

Species: All Species Breed: All Breeds



Taille personnalisée ▾ 1800 x 900 (100 %) | x 3 ▾ | Ouvrir avec un simulateur ▾

ThomasMaye Home About Contact Meetups ▾ Pets ▾ Profile ▾

Pets List

Here you can see all the pets that are registered in the system.
You can view his pet's profile, and on the profile, you can propose him for a meetup, if you are interested by him.
You can also see the owner's profile and contact him.

Select an option ▾ Select an option ▾ Filter

Species: All Species Breed: All Breeds



Mavis ▾

Marey ▾

Nicky ▾

Luki ▾

ANNEXE 7.1 : Code du formulaire de création d'un "Meetup"

```
22 <div class="container mx-auto p-4">
23   <form method="POST" action="{{ route('createMeetup') }}">
24     {{ csrfField() }}
25     <div class="px-4 py-4 border border-gray-200 dark:border-gray-700 rounded-lg shadow-lg bg-white dark:bg-gray-800">
26       <div class="px-4 text-emerald-700 text-lg grid grid-cols-1 md:grid-cols-2 gap-6 ">
27         <div>
28
29           <div class="mb-4">
30             What is the name of your meetup?
31             <div class="">
32               @!component('components/input', { name: 'title', placeholder: 'Title', required: true})
33             </div>
34           </div>
35
36           <div class="mb-4">
37             When is your meetup?
38             <div class="">
39               @!component('components/input', {
40                 type: 'datetime-local',
41                 id: 'date',
42                 name: 'date',
43                 class: 'mt-1 block w-full',
44                 required: true,
45                 min: new Date().toISOString().slice(0, 16)
46               })
47             </div>
48           </div>
49
50           <div class="mb-4">
51             Where did you want the meetup to take place?
52             <div class="">
53               @!component('components/input', { name: 'adress', placeholder: 'Address', required: true})
54               @!component('components/input', { name: 'additional_address', placeholder: 'Additional Address'})
55               <div class="grid grid-cols-2 gap-2 -my-2">
56                 @!component('components/input', { name: 'postal_code', placeholder: 'Postal Code', required: true})
57                 @!component('components/input', { name: 'city', placeholder: 'City', required: true})
58               </div>
59             </div>
60           </div>
61         </div>
62
63         <div>
64           <div class="mb-4">
65             Choose a type of meetup
66             <div class="">
```

ANNEXE 7.2 : Code du formulaire de création d'un "Meetup" (suite)

```
66     Choose a type of meetup
67     <div class="">
68         @!component('components/input', {
69             type: 'select',
70             id: 'type',
71             name: 'type',
72             required: true,
73             options: [
74                 { value: 'Walk', text: 'Walk' },
75                 { value: 'Sociabilization', text: 'Sociabilization' }
76             ]
77         })
78     </div>
79 </div>
80
81     <div class="mb-4">
82         <label class="block text-emerald-700 mb-2">Select your pets for the walk: </label>
83         <div class="text-sm space-x-2 border px-2 py-2 rounded-lg bg-white text-gray-700 outline outline-1 -outline-offset-1 outline-gray-300 flex flex-wrap">
84             @each(pet in pets)
85             <div class="flex items-center">
86                 <input types="checkbox" id="pet-{{ pet.id }}" name="petIds[]" value="{{ pet.id }}"
87                         class="w-4 h-4 text-emerald-700 border-gray-300 rounded focus:ring-emerald-700">
88                 <label for="pet-{{ pet.id }}" class="ml-2 text-gray-700">
89                     {{ pet.name }}
90                 </label>
91             </div>
92             @end
93         </div>
94     </div>
95
96     <div class="mb-4">
97         Add a description of your meetup
98         <div class="">
99             @!component('components/input', { name: 'description', placeholder: 'Description', type: 'textarea' })
100            </div>
101        </div>
102
103    </div>
104
105    </div>
106 </div>
107 </div>
108
109 <div class="flex justify-center mt-2">
110     @!component('components/button-simple/full', { text: "Create" })
111 </div>
```

ANNEXE 8 : Code du composant personnalisé “Input”

```

1  <div class="text-sm text-gray-900 dark:text-white mt-2">
2    <label for="{{name}}>{{label || ''}}</label>
3    <div class="mt-1">
4      @if(type === 'select')
5        <select name="{{name}}" id="{{name}}" {{ required ? 'required' : '' }}>
6          class="w-full rounded-lg bg-white px-5 py-2.5 text-gray-700 outline outline-1 -outline-offset-1 outline-gray-300
7            focus:outline focus:outline-2 focus:-outline-offset-2 focus:outline-emerald-700 sm:text-sm" style="height: 2.5rem;">
8
9          <option value="">Select an option</option>
10         @each(option in options)
11           <option value="{{ option.value.toString() }}>
12             {{ old(name) ?? value == option.value.toString() ? 'selected' : '' }}>
13             {{ option.text }}
14           </option>
15         @end
16
17       </select>
18
19     @elseif(type === 'textarea')
20       <textarea name="{{name}}" id="{{name}}" placeholder="{{ placeholder || 'Enter your ${label}'}}>{{ required ? 'required' : '' }}>
21       rows="5"
22       class="block px-4 py-2 mt-2 w-full rounded-lg bg-white text-gray-700 outline outline-1 -outline-offset-1 outline-gray-300
23         placeholder:text-gray-400 focus:outline focus:outline-2 focus:-outline-offset-2 focus:outline-emerald-700 sm:text-sm">{{old(name) || value || ''}}</textarea>
24
25     @elseif(type === 'checkbox')
26       <input type="checkbox" name="{{name}}" id="{{name}}"
27         {{ old(name) ?? value ? 'checked' : '' }} {{ required ? 'required' : '' }}>
28
29     @elseif(type === 'file')
30       <input type="file" name="{{name}}" id="{{name}}" accept="{{accept}}>{{ required ? 'required' : '' }}>
31       class="block w-full px-3 py-2 mt-2 text-sm text-gray-600 bg-white border outline outline-1 -outline-offset-1 outline-gray-300
32         rounded-lg file:bg-emerald-600 file:text-white file:text-sm file:px-4 file:py-2 file:border-none file:rounded-lg
33         dark:file:text-gray-200 dark:text-gray-300 placeholder-gray-400/70 dark:placeholder-gray-500 focus:border-blue-400
34         focus:outline-none focus:ring focus:ring-blue-300 focus:ring-opacity-40 dark:border-gray-600 dark:bg-gray-900
35         dark:focus:border-blue-300">
36
37     @elseif(type === 'range')
38       <input type="range" id="{{ name }}" name="{{ name }}" min="{{ min }}" max="{{ max }}" value="{{ old(name) ?? value }}"
39         {{ required ? 'required' : '' }}>
40       class="form-range block w-full border-gray-300 rounded-lg shadow-sm focus:ring-emerald-500 focus:border-emerald-500">
41
42     @elseif(type === 'date')
43       <input type="date" name="{{name}}" id="{{name}}" value="{{ old(name) ?? value }}>{{ required ? 'required' : '' }}>
44       class="block mt-2 w-full rounded-lg bg-white px-5 py-2.5 text-gray-700 outline outline-1 -outline-offset-1 outline-gray-300
45         placeholder:text-gray-400 focus:border-emerald-700 focus:outline-emerald-700 focus:outline focus:outline-2 focus:-outline-offset-2">
46
47     @else
48       <input type="{{type || 'text'}}" name="{{name}}" id="{{name}}" autocomplete="{{name}}"
49         placeholder="{{ placeholder || 'Enter your ${label}'}}>{{ required ? 'required' : '' }}>
50         value="{{ old(name) || value || '' }}>
51       class="block mt-2 w-full rounded-lg bg-white px-5 py-2.5 text-gray-700 outline outline-1 -outline-offset-1 outline-gray-300
52         placeholder:text-gray-400 focus:outline focus:outline-2 focus:-outline-offset-2 focus:outline-emerald-700 sm:text-sm">
53       @endif
54     </div>
55   </div>
56
57   @inputError(name)
58   @each(message in $messages)
59   <p class="text-red-500">{{ message }}</p>
60   @end
61   @end

```

ANNEXE 9 : Extrait Code réalisé pour la page “My Upcoming Meetups”

```

34
35  {{--No meetups--}}
36  @if(meetups.length === 0)
37  <div class="px-4 mt-10">
38    <h1 class="text-2xl font-semibold text-emerald-700 flex items-center">
39      <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24" class="mr-2">
40        <path fill="currentColor" d="M12 11.5q1.625 0 2.8 1.1t1.4 2.725q.05 3.275 4.88T17 16t.537-.2t.188-.5q-.25-2.25-1.863-3.775T12 18t-3.863 1.525T6.275 15.3q-.05 3.188.5T7 16t.525-187t.275-.488Q8.025 13.7 9.2 12.6t2.8-1.1m-1.45-3.975q.225-.2.213-.513T10.5 6.5t-.513-1.45m-3.975q.225-.2.213-.513T10.5 6.5t-.513-1.45"/>
41    You haven't created or joined any meetups yet
42  </h1>
43  </div>
44  <div class="mt-8">
45    @component('components/button-link/full', { text: 'Create a Meetup', link: route('createMeetupForm') })
46  </div>
47  @endif
48

76
77  {{-- Meetups list section -->}
78  <div class="lg:w-2/3">
79    <div class="grid gap-6 grid-cols-1 md:grid-cols-2 xl:grid-cols-2 2xl:grid-cols-3">
80      @each(meetup in meetups)
81      @if(new Date(meetup.date) >= new Date())
82
83        {{-- Meetup card -->}}
84        <div
85          class="shadow-lg hover:shadow-xl flex flex-col p-4 border-2 border-gray-200 hover:border-gray-400 sm:p-6 rounded-xl dark:border-gray-500 text-gray-500 dark:text-gray-300 transition duration-150 ease-in-out hover:-translate-y-px"
86
87        {{-- Existing meetup card content -->}}
88        <h2 class="text-xl font-semibold text-emerald-700">{{ meetup.title }}</h2>
89        <div class="flex items">
90          <span class="inline-block w-40 h-0.5 bg-amber-400 rounded-full"></span>
91          <span class="inline-block w-3 h-0.5 mx-1 bg-amber-400 rounded-full"></span>
92          <span class="inline-block w-1 h-0.5 bg-amber-400 rounded-full"></span>
93        </div>
94
95        <div class="space-y-2 dark:text-gray-200 text-gray-600">
96          {{ meetup.description }}
97
98          <p class="mt-2 flex items-center">
99            <a href="{{ route('auth.display_user_profile', { id: meetup.userId }) }}>
100              <span
101                class="bg-amber-400 text-emerald-700 hover:bg-amber-500 hover:text-emerald-800 px-2 py-1 rounded-lg flex items-center mr-2 text-sm">
102                <svg class="mr-2" xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24">
103                  <path fill="currentColor" d="M11 13v3q0 .425.288.713T12 17t.713-.288T13 16v-3h3q.425 0 .713-.288T17 12t-.288-.712T16 11h-3V8q0-.425-.288-.712T12 7t-.712.288T11 8V3H8q-.425 0 -.712.288T7 12t.288.713T8 13zm1 9q-2.075 0-3.9-.788t-3.175-2.137T2.788
104                </svg>
105                by {{ auth.user && meetup.userId === auth.user.id ? 'Me' : meetup.organizer }}
106              </span>
107            </a>
108          </p>
109
110        <p class="mt-2 flex items-center">
111          <span class="bg-amber-400 text-emerald-700 px-2 py-1 rounded-lg flex items-center mr-2 text-sm">
112            <svg class="mr-2" xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24">
113              <path fill="currentColor" d="M5 22q-.825 0-1.412-.587T3 20V6q0-.825.588-1.412T5 4h1V2h2v2h8V2h2v1q.825 0 1.413.588T21 6V4.675q0 .425-.288.713t-.712.287t-.288-.712V18H5v10h5.8q.425 0 .713.288T11.8 21t-.288.713T10.8 22zm1 1q-2.075 0-3.9-.788t-3.175-2.137T2.788
114            </svg>
115            {{ meetup.formattedDate }}
116          </span>
117        </p>
118
119      </div>
120
121      @if(auth.user && meetup.userId === auth.user.id && new Date(meetup.date) >= new Date())
122      <a href="{{ route('updateMeetup', { id: meetup.id }) }}>
123        <button
124          class="inline-flex items-center gap-2 rounded-lg px-4 py-2 text-gray-100 dark:text-gray-300 hover:text-white hover:background-emerald-800 focus:outline">
125            <svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" stroke-width="1.5"
126              stroke="currentColor" class="size-4">
127              <path stroke-linecap="round" stroke-linejoin="round"
128                d="M16.862 4.48711.6881.875 1.875 0 112.652 2.652L10.582 16.07a4.5 4.5 0 01-1.897 1.13L6 181.8-2.685a4.5 4.5 0 011.13-1.897 18.932-8.931z0 0L19.5 7.125M18 14V4.75A2.25 2.25 0 0115.75 21H5.25A2.25 2.25 0 01
129            </svg>
130
131          Edit
132        </button>
133      </a>
134    @end
135
136    @if(auth.user && meetup.userId === auth.user.id && new Date(meetup.date) >= new Date())
137    <a href="{{ route('deleteMeetupView', { id: meetup.id }) }}>
138      <button
139        class="inline-flex items-center gap-2 rounded-lg px-4 py-2 text-gray-100 dark:text-gray-300 hover:text-white hover:background-emerald-800 focus:outline">
140          <svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24" stroke-width="1.5"
141            stroke="currentColor" class="size-4">
142            <path stroke-linecap="round" stroke-linejoin="round"
143              d="M14.74 91.346 9m-4.788 0 9.26 9m9.968-3.212.342.052.682.107 1.022.166m-1.022-1.051L18.16 19.673a2.25 2.25 0 01-2.244 2.077H8.084a2.25 2.25 0 01-2.244-2.077L4.772 5.79m14.456 0a48.108 48.108 0 00-3.478-.397m-12
144        </svg>
145
146      Delete
147    </button>
148  </div>
149
150
```

ANNEXE 10 : Test Crédit Meetup (Scénario 1)

The screenshot shows the 'Create A Meetup' page. At the top, there's a logo for 'ORI&NORI' featuring two cartoon animals. The navigation bar includes links for 'Home', 'About', 'Contact', 'Meetups', 'Pets', and 'Profile'. The main form fields are: 'What is the name of your meetup?' (input: 'Corginade'), 'Choose a type of meetup' (dropdown: 'Walk'), 'When is your meetup?' (input: '19/06/2025 16:00'), 'Select your pets for the walk' (checkboxes: 'Noni' and 'Ori' both checked), 'Where did you want the meetup to take place?' (input: 'Prarie des Filtres'), 'Additional Address' (input: '31300'), 'Toulouse' (input: 'Toulouse'), and 'Add a description of your meetup' (text area: 'Balade dans Toulouse entre corgi et chat'). A 'Create' button is at the bottom right.

Création du meetup

The screenshot shows the 'My Upcomming Meetups' section. It displays a success message: 'Success You have successfully created a Meetup'. Below it is a map of Toulouse with a blue marker indicating the meetup location. To the right is a card for the 'Corginade' meetup, which includes details: 'by Me', '19/06/2025 16:00', 'Prairie des Filtres - Toulouse', '2 Pets have joined this Meetup: Chien de Berger des Pyrénées à face rase, European Shorthair', and buttons for 'View', 'Edit', and 'Delete'.

Meetup validé je suis le créateur du meetup

The screenshot shows the Ori&Nori website interface. At the top, there is a navigation bar with links for Home, About, Contact, Meetups, Pets, and Profile. The user 'Mezanic' is logged in. The main content area displays a map of Toulouse with a blue marker indicating the meetup location. To the right of the map, the event details are shown:

- Corginade**
- Balade Dans Toulouse Entre Corgi Et Chat
- By Me
- 19/06/2025 16:00
- Prairie Des Filtres, Toulouse
- Walk
- Edit

Below these details, there are two sections: 'Pets Participants' (Ori, Nodi) and 'Participants' (Mezanic). The entire content is enclosed in a light gray rounded rectangle.

Mes animaux sont bien dans le meetup

ANNEXE 11 : Tentative de création d'un Meetup sans animaux sélectionnés (Scénario 2)

Create A Meetup

Here you can create a new meetup for your pets. Fill in the details below and select the pets you want to include.

What is the name of your meetup?

Choose a type of meetup

When is your meetup?

Select your pets for the walk: Nori Ori

Where did you want the meetup to take place?

Add a description of your meetup

Balade entre Corgi au bord de l'eau

Create

Formulaire de création rempli sans animaux

Create A Meetup

Here you can create a new meetup for your pets. Fill in the details below and select the pets you want to include.

Error
You must select at least one pet for the meetup

What is the name of your meetup?

Choose a type of meetup

When is your meetup?

Select your pets for the walk: Nori Ori

Where did you want the meetup to take place?

Add a description of your meetup

Description

Redirection vers le formulaire de création avec le message d'erreur.

ANNEXE 12 : Test rejoindre un Meetup (Scénario 3)

The screenshot shows the Ori&Nori website's 'Meetups' section. At the top, there is a navigation bar with links for Home, About, Contact, Meetups, Pets, and Profile. On the left, there is a logo featuring two cartoon animals and the text 'ORI&NORI'. Below the navigation, a heading says 'All Meetups'. A map of Toulouse and its surroundings is displayed, with a blue marker indicating the location of the meetup. To the right of the map, a specific meetup titled 'Corginade' is shown. The details for this meetup include: 'Balade dans Toulouse entre corgi et chat', organized by 'Mezanic' on '19/06/2025 16:00' at 'Prairie des Filtres - Toulouse'. It notes that '2 Pets have joined this Meetup:' with examples like 'Chien de Berger des Pyrénées à face rose' and 'European Shorthair'. There is also a 'View' button.

Liste des Meetup que je peux rejoindre

The screenshot shows the details of the 'Corginade' meetup from the previous screen. The heading is 'Details of Corginade'. It reiterates the details: 'Balade Dans Toulouse Entre Corgi Et Chat', organized by 'Mezanic' on '19/06/2025 16:00' at 'Prairie Des Filtres - Toulouse'. It includes a map of Toulouse. On the right, there are sections for adding pets ('Add One Or More Of Your Pets') and participants ('Participants'). Under 'Participants', it shows 'Marin' by 'Mezanic'. Below that, under 'Pets Participants', there are buttons for 'Corgi' and 'Non'. There is also a 'Join with' button.

Vue de la page du Meetup en cochant Marin comme animal à ajouté.

My Upcomming Meetups

Here you can find all the meetups you have joined or created.
Click on a meetup to view more details (or edit it if you are the creator).

Corginade
Balade dans Toulouse entre corgi et chat
by Mezanic
19/06/2025 16:00
Prairie des Filtres - Toulouse

3 Pets have joined this Meetup:
Chien de Berger des Pyrénées à face rase
European Shorthair
Rottweiler

Redirection vers les Meetup de l'utilisateur on y voit bien le meetup et le message de succès.

Details of Corginade

Here is the details of the meetup.
You can join or leave the meetup.
You can also add or remove your pets from the meetup. And you can also leave a review for the meetup after the meetup.

Corginade
Balade Dans Toulouse Entre Corgi Et Chat
by Mezanic
19/06/2025 16:00
Prairie Des Filtres - Toulouse
Walk
Leave

Pets Participants: Dri, Nori, Marin, User33

Participants: Mezanic, User33

Vue des détails du Meetup on l'on voit bien Marin et l'user33 comme participant au Meetup

ANNEXE 13: Test XSS

Details of test2

Here is the details of the meetup.
You can join or leave the meetup.
You can also add or remove your pets from the meetup. And you can also leave a review for the meetup after the meetup.

Test2
Eaz ★ 4/5
By Me
19/06/2025 08:46
Prairie Des Filtres
Toulouse
Walk

Mezanic99 ★ 4/5
<Script>Alert('XSS')</Script>
19/06/2025 08:52
Edit Delete

Exemple de balise html écrit dans un avis, pour tester les failles XSS

Tableau	id	rating	description	meetup_id	user_id	created_at	updated_at
Tableau	1	1	4 <script>alert('XSS')</script>	5	7	2025-06-19 10:52:24.944 +0200	2025-06-19 10:52:24.944 +0200
Texte							

Stockage de la review dans la base de données

ANNEXE 14 : Dockerfile

```
⚡ Dockerfile > ...
1  FROM node:24 AS base
2
3  # All deps stage
4  FROM base AS deps
5  WORKDIR /app
6  ADD package.json package-lock.json ./
7  RUN npm ci
8
9  # Production only deps stage
10 FROM base AS production-deps
11 WORKDIR /app
12 ADD package.json package-lock.json ./
13 RUN npm ci --omit=dev
14
15 # Build stage
16 FROM base AS build
17 WORKDIR /app
18 COPY --from=deps /app/node_modules /app/node_modules
19 ADD . .
20 RUN node ace build
21
22 # Production stage
23 FROM base
24 ENV NODE_ENV=production
25 WORKDIR /app
26 COPY --from=production-deps /app/node_modules /app/node_modules
27 COPY --from=build /app/build /app
28 COPY .env .env
29 EXPOSE 8080
30 CMD ["node", "./bin/server.js"]
31
```

Dockerfile

ANNEXE 15 : Docker compose

```
1   services:
2     traefik:
3       image: traefik:3
4       restart: always
5       ports:
6         - 80:80
7         - 443:443
8       volumes:
9         - ./docker/traefik/traefik.yml:/etc/traefik/traefik.yml:ro
10      - ./docker/traefik/acme.json:/acme.json
11      - /var/run/docker.sock:/var/run/docker.sock
12     labels:
13       - traefik.enable=true
14     networks:
15       - traefik
16       - internal
17   node:
18     build:
19       context: ../
20     restart: always
21     depends_on:
22       - pgsql
23     labels:
24       - traefik.enable=true
25       - traefik.http.routers.node.rule=Host(`oriandnori.com`)
26       - traefik.http.routers.node.entrypoints=websecure
27       - traefik.http.routers.node.tls=true
28       - traefik.http.routers.node.tls.certresolver=letsencrypt
29     networks:
30       - traefik
31       - internal
32   pgsql:
33     image: postgres:17
34     restart: always
35     environment:
36       POSTGRES_DB: ${DB_DATABASE}
37       POSTGRES_USER: ${DB_USER}
38       POSTGRES_PASSWORD: ${DB_PASSWORD}
39     volumes:
40       - oriandnori-pgsql:/var/lib/postgresql/data
41     networks:
42       - internal
43     volumes:
44       oriandnori-pgsql:
45         name: oriandnori-pgsql
46         driver: local
47   networks:
48     traefik:
49       external: false
50     internal:
51       external: false
```

Docker-compose.prod.yml