RETEX

Projet Annuel ESGI B3

Logiciel de gestion de salle de sport

https://github.com/guilliammrst/GymManagement

Du 24/10/2024 au 14/07/2025

Guilliam MORISSET

Sommaire

1. Analyse du projet

1.1. Contexte du projet

Ce projet s'inscrit comme projet annuel de troisième année de Bachelor Informatique à l'ESGI Nantes. Le but étant de développer notre capacité à gérer et mener à bien un projet. Ce projet a été réalisé en binôme, en collaboration avec Elouan LE BRAS. Nous avons choisi de développer un logiciel de gestion de salle de sport. Pourquoi ce choix ? Car le sport est une de nos passions communes et le sujet nous motivait.

1.2. Chronologie du projet

Le projet a été lancé le jeudi 24 octobre, c'est à ce moment précis que nous avons formé le groupe et choisi notre projet.

Une fois le projet lancé, nous sommes entrés en phase de préparation. Nous avons commencé par travailler sur la définition du besoin en faisant une AFB (Analyse Fonctionnelle du Besoin), pour ce faire nous avons utilisé Miro pour regrouper tous nos documents. Nous avons utilisé une mindmap (avec l'aide de mindmeister) pour noter toutes nos idées et clarifier notre projet. Puis, nous avons défini les risques et les fonctionnalités ainsi que le cahier des charges. Nous avons terminé par faire un POC (proof of concept) pour vérifier la faisabilité de notre projet.

Nous avons présenté notre projet à Nassim ZGA le vendredi 25 avril, pour pouvoir commencer la phase de mise en œuvre.

Suite à la validation du projet par Nassim, nous avons pu entrer en phase de mise en œuvre et avons pu mettre en œuvre notre management de projet ainsi que démarrer le développement du logiciel.

On peut constater plusieurs écarts de planification sur les diagrammes ci-dessous, le MVP devait être fini mi-mai mais il a été terminé fin mai, avec un écart de 2 semaines. Concernant la v1.0.0, elle devait être livrée début juin mais a été livrée fin juin. La v2.0.0 est elle toujours en développement à l'heure où j'écris ce RETEX. Après avoir constaté que notre estimation initiale était trop optimiste nous avons choisi de ne pas faire la v3.0.0 (qui était du bonus).

Le projet s'est terminé le mardi 22 juillet lors de la présentation finale faite devant Nassim. Quant à elle, la livraison des livrables s'est faite le dimanche 13 juillet.

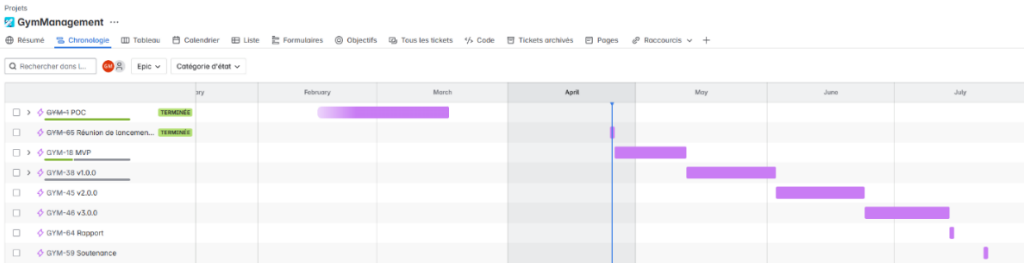

Figure 1 - Diagramme de Gantt Initial

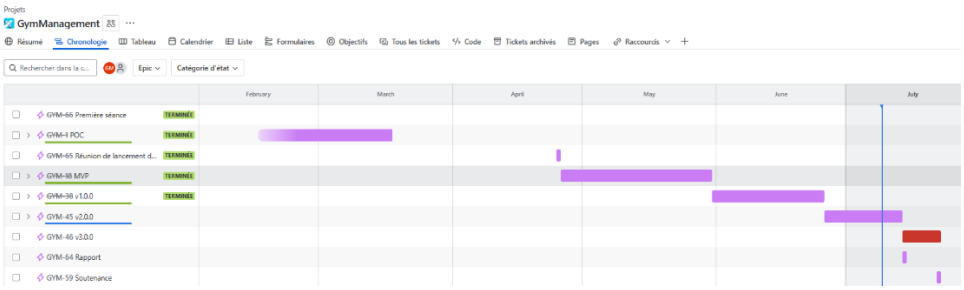

Figure 2 - Diagramme de Gantt Final

1.3. Travail prévu vs réalisé

Au début du projet, nous avions défini cette liste de versions :

- POC : Connexion sécurisée via token à une interface web admin (Staff + Manager) + intégration bdd et secrets via Azure + initialisation clean archi
- MVP : Interface web où le staff peut ajouter des membres/clubs/plans d'abonnement et inscrire un utilisateur à un plan (inscription uniquement en physique à la salle et pas d'accès automatique à la salle)
- v1.0.0 : Utilisateur peut s'inscrire, choisir un abonnement, payer, accéder à une salle via un qr code (via interface mobile), gérer son compte et admins peuvent gérer les salles/utilisateurs
- v2.0.0 : Interface web devient accessible aux coachs qui pourront ajouter des coachings (mais pas de gestion de salle/utilisateurs) et membre peuvent s'y inscrire (via interface mobile) + statistiques ? (Interface web)
- v3.0.0 (bonus) : Jobs/azure fonctions pour gérer les expirations des abonnements/renouvellements (tournent vers minuit pour invalider les abonnements) + statistiques (interface web) + déployer sur Azure et configurer les différents services avec Docker

Nous avons choisi de ne pas réaliser la v3.0.0 par manque de temps. On peut expliquer cet écart de planning par l'exigence que nous nous sommes donnés sur l'architecture de notre projet, rendant le projet très qualitatif mais augmentant le temps de développement. Concernant la matrice de conformité, comme on peut le voir ci-dessous, la plupart des exigences ont été validées, il reste quelques exigences en cours de développement au moment où j'écris ces lignes et la gestion des coachings qu'on aimerait finir avant la fin. Concernant les exigences avec le statut « Non prévu » ce sont les exigences de la v3.0.0 qui ne sera pas développée. Il y a quelques besoins qui n'avaient pas été définis dans le cahier des charges initial comme la « Refonte du coté front : UI/UX clean », qui nous a été conseillée par Nassim car notre interface web était très minimaliste ce qui impactait négativement le travail perçu.

ID	Description	Statut final
GYM-3	Définir le schéma de base de données	OK
GYM-4	Configurer la base de données via un cloud provider	OK
GYM-5	Implémenter Entity Framework pour pouvoir contacter la base de données	OK
GYM-7	Configurer le coffre de stockage via un cloud provider	OK
GYM-8	Implémenter un service pour récupérer les secrets contenu dans le coffre de stockage	OK
GYM-10	Liste des utilisateurs	OK
GYM-11	Création d'un utilisateur par admin	OK
GYM-14	Obtention d'un token utilisateur	OK
GYM-16	Interface de connexion	OK
GYM-17	Tableau de bord des utilisateurs	OK
GYM-20	Création de compte autonome	OK
GYM-21	Informations de l'utilisateur connecté	OK
GYM-23	Détails d'un utilisateur	OK
GYM-24	Mise à jour des informations d'un utilisateur	OK
GYM-26	Abonner un utilisateur	OK
GYM-27	Ajouter un club	OK
GYM-30	Ajouter un plan d'abonnement	OK
GYM-35	Liste des clubs	OK
GYM-36	Liste des plans d'abonnement	OK
GYM-33	Page utilisateurs	OK
GYM-37	Page détail d'un utilisateur	OK
GYM-40	Page clubs	OK
GYM-44	Page plans d'abonnement	OK
GYM-71	Refresh token	OK

ID	Description	Statut final
GYM-25	Suppression d'un utilisateur	OK
GYM-28	Modifier un club	OK
GYM-30	Modifier un plan d'abonnement	OK
GYM-31	Liste des utilisateurs avec filtres et pagination	OK
GYM-42	Utilisateur peut s'abonner	OK
GYM-43	Utilisateur peut créer un qr code temporaire pour accéder à une salle	OK
GYM-49	Utilisateur peut modifier ses informations de compte	OK
GYM-50	Utilisateur peut supprimer son compte	OK
GYM-51	Page utilisateurs => Ajout recherche et pagination	OK
GYM-52	Page utilisateur => Ajout bouton suppression	OK
GYM-53	Page clubs => Ajout bouton modification	OK
GYM-54	Page plans d'abonnement => Ajout bouton modification	OK
GYM-55	Connexion/inscription sur l'app mobile	OK
GYM-56	Page d'accueil sur l'app mobile	En cours
GYM-57	Page informations de compte sur l'app mobile	En cours
GYM-58	Page d'abonnement sur l'app mobile	En cours
GYM-77	Page QrCode sur l'app mobile	En cours
GYM-72	Refonte du coté front : UI/UX clean	En cours
GYM-79	Ajout d'un coaching	NOK
GYM-80	Page création d'un coaching	NOK
GYM-81	Inscrire un utilisateur à un coaching	NOK
GYM-82	Page d'inscription à un coaching sur l'app mobile	NOK
GYM-84	Job expiration et renouvellement d'abonnement	Non prévu
GYM-85	Déploiement des apps sur Azure	Non prévu
GYM-86	Statistiques	Non prévu

Figure 3 - Matrice de conformité

1.4. Kanban projet et répartition des tâches

Au sein de l'équipe, j'ai occupé le rôle de référent Jira ainsi que celui de lead développeur. À ce titre, j'ai participé activement à l'organisation du travail, à la création et à l'attribution des tickets, ainsi qu'au suivi de l'avancement du projet.

Sur un total de 41 tickets, j'en ai pris en charge 25, soit plus de 60 %. En parallèle, j'ai effectué 18 commits Git (hors commits de merge ou de pull requests), sur un total de 27 commits, ce qui témoigne d'une forte implication technique. J'ai majoritairement travaillé sur la partie back-end du projet, en développant l'API REST et la logique métier.

Je me suis senti très investi tout au long du projet. À plusieurs reprises, j'ai dû reprendre certaines tâches initialement attribuées à d'autres membres afin de ne pas accumuler trop de retard. Même si la répartition initiale des tâches était correcte, la motivation et la disponibilité ont varié selon les membres, ce qui a nécessité un certain réajustement en cours de route.

J'ai rédigé l'ensemble des tickets dans Jira en utilisant la méthode des "user stories", ce qui a permis de mieux exprimer les besoins métiers et les attentes fonctionnelles. Cette formulation a facilité le travail de priorisation, de développement et de validation fonctionnelle. J'ai aussi intégré des critères d'acceptation (CA) clairs pour guider le développement et les tests. Les tickets suivaient la structure :

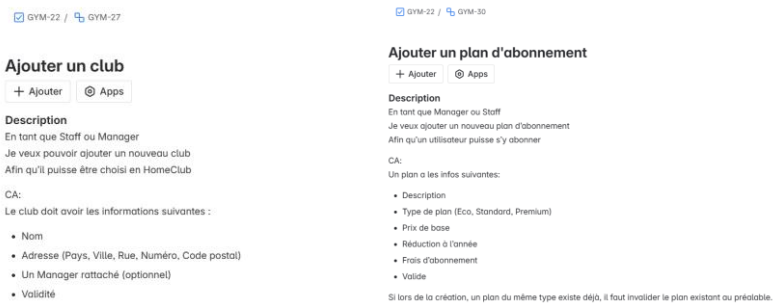

Figure 4 - Exemples user story

2. Bilan du projet

2.1. Liste des outils utilisés

2.1.1. Technique

Outils	Description	Comment l'avons-nous utilisé ?	Justification du choix	Bilan
Visual Studio 2022 Community	IDE officiel de Microsoft pour C#, .NET	Utilisé pour développer, compiler, exécuter et déboguer. Intégration avec Git et gestion des packages NuGet.	Interface complète, nombreuses extensions, outils de debug puissants.	Bon choix
C#	Langage de programmation orienté objet de Microsoft	Utilisé pour l'ensemble du back-end (API), la logique métier, la web app (Blazor) et l'app mobile (MAUI)	Langage central de l'écosystème .NET, parfaitement intégré à Visual Studio	Bon choix

ASP.NET Core	Framework web pour créer des API RESTful	Utilisé pour exposer des services REST consommés par l'app mobile et la web app	Framework moderne, robuste, performant	Bon choix
.NET MAUI	Framework cross-platform pour créer des applications mobiles et desktop en C#	Utilisé pour développer l'application mobile Android de notre solution (interface utilisateur)	Permet un développement partagé (code commun) entre plateformes, support natif Android/iOS	Améliorable : encore jeune, bugs mineurs rencontrés
Blazor/Razor	Framework pour construire des interfaces web interactives en C# / HTML	Utilisé pour développer la partie front-end de la web app (interface admin)	Permet de rester full-stack C#, bonne intégration avec ASP.NET et le back-end	Bon choix
Aspire	Outil de l'écosystème .NET pour la composition, l'orchestration et l'observabilité des applications distribuées	Utilisé pour organiser le projet en services, simplifier la configuration, centraliser le monitoring et visualiser les dépendances	Solution native .NET, très utile pour des projets multi-services. Gain de temps en configuration et supervision	Bon choix
PostgreSQL (Azure)	Système de gestion de base de données relationnelle, hébergé sur Microsoft Azure	Utilisé comme base de données principale pour stocker les utilisateurs, données métiers, etc. Accessible via les API ASP.NET	Choix open-source, robuste, compatible avec .NET. L'hébergement Azure assure la scalabilité et la sécurité	Bon choix
Azure Key Vault	Service cloud sécurisé de gestion de secrets, clés API, mots de passe	Utilisé pour stocker les chaînes de connexion à la BDD et secrets sensibles	Permet de ne pas exposer de secrets en dur dans le code source. Intégration native avec Azure	Bon choix

2.1.2. Projet

Outils	Description	Comment l'avons-nous utilisé ?	Justification du choix	Bilan
Snapchat	Réseau social	- Communication instantanée	- Plus actif sur Snapchat que sur Discord	Bon choix
Discord	Plateforme de VoIP et messagerie	- Partage des ressources - Call pour points réguliers - Entraide approfondie	- Partage d'écran - Rappel pour les points réguliers	Bon choix
Jira	Outil de gestion de projets	- Attribution des tâches - Ecrire les différents fixs à faire en plus des tâches	- Savoir où chacun est rendu - Centraliser	Bon choix

Mindmeister	Outil de création de mindmap	- Mettre toutes nos idées en forme	- Commencer sans limiter ses idées	Bon choix
Miro	Dashboard de collaboration	- Centraliser nos documents - Partage des ressources	- Collaboration active	Bon choix
GitHub	Plateforme de versionnage Git et collaboration en ligne	- Utilisé pour héberger le code, faire du versioning, des pull requests, et gérer les branches du projet	- Plateforme largement utilisée, intégrée à Visual Studio. Pratique pour le travail en équipe	Bon choix

2.2. Points positifs à capitaliser

2.2.1. Partie technique

Cohérence de la stack 100 % .NET → Utiliser C#, ASP.NET, Blazor, MAUI et PostgreSQL dans un écosystème unifié a permis une meilleure compréhension globale du projet et une réutilisation de code.

Usage d'Aspire → Aspire a facilité la visualisation des dépendances techniques et la configuration multi-services.

Intégration d'Azure Key Vault → Un vrai plus pour la sécurité. Cela nous a évité de stocker des secrets en clair dans le code.

Utilisation efficace de Visual Studio 2022 → IDE stable et complet, très pratique pour le debug, les tests unitaires et l'intégration avec GitHub.

Combinaison de Domain Driven Design (DDD) et Clean Architecture → Pour structurer notre application, nous avons combiné les principes du Domain Driven Design (DDD) avec ceux de la Clean Architecture. Nous a permis d'organiser le code autour du métier, avec des entités riches et des règles métiers bien encapsulées (DDD), de séparer clairement les responsabilités grâce aux couches applicatives indépendantes (Clean Architecture), rendre le domaine totalement indépendant des technologies externes (framework, BDD, UI...), faciliter la testabilité et la maintenabilité du projet.

2.2.2. Partie projet

Organisation via Jira → L'attribution claire des tâches et le suivi de l'avancement ont permis une bonne anticipation des retards. L'utilisation de la méthode "user story" pour la rédaction des tickets Jira a permis une meilleure compréhension des besoins métier et une validation plus fluide des fonctionnalités.

Gestion de versions via GitHub → Travail par branches + pull requests bien respecté, ce qui a limité les conflits et permis une revue de code collaborative.

Communication fluide en équipe → Les réunions régulières et l'entraide entre membres ont permis de garder un bon rythme malgré quelques retards.

Réactivité face aux imprévus → Lors de retards ou blocages techniques, des solutions ont été trouvées rapidement grâce à une bonne coordination.

2.3. Points de vigilance à améliorer

2.3.1. Partie technique

Courbe d'apprentissage de .NET MAUI → Le framework étant encore jeune, nous avons rencontré des bugs et un manque de documentation, ce qui a ralenti le développement mobile. Il aurait été utile de prévoir plus de temps de veille technique au départ.

Manque de tests automatisés → Peu de tests unitaires ou d'intégration ont été mis en place, ce qui a rendu les phases de recette plus risquées.

2.3.2. Partie projet

Accumulation de retards dans l'avancement global du projet → En raison de notre effectif réduit (équipe de 2 personnes), la charge de travail était plus concentrée. Certaines tâches ont pris plus de temps que prévu.

Amélioration possible : établir un planning plus souple, avec des marges de sécurité réalistes, et prévoir des points de suivi réguliers, même à deux.

3. Mon apport à l'équipe

3.1. Partie technique

3.1.1. Liste des fonctionnalités développées

Voici les principales fonctionnalités que j'ai développées :

- Authentification (création de compte, connexion, refresh token)
- Gestion des utilisateurs : création, modification, suppression, informations de l'utilisateur connecté
- Pages fonctionnelles : page d'abonnement, page d'accueil, page informations de compte, page QR code
- QR code temporaire pour accéder à une salle
- Abonnement (s'abonner, modifier un abonnement)
- Gestion des clubs (modifier un club)
- Tableau de bord et liste des utilisateurs
- Interface d'administration et interface de connexion

3.1.2. Activités techniques prises en charge

En plus du développement fonctionnel, j'ai également :

- Implémenté la couche Entity Framework pour l'accès à la base de données
- Configuré la base de données PostgreSQL et son intégration avec Azure
- Mis en place le coffre de stockage Azure Key Vault, avec un service pour récupérer les secrets
- Intégré la gestion des tokens d'authentification (JWT)
- Travaillé à la sécurisation des communications entre services
- Été à l'initiative de la structuration du projet selon une Clean Architecture guidée par le DDD, avec une séparation stricte en 5 couches : Domain, Application, Presentation, Infrastructure et Shared (classes partagées, comme les enums…)

3.2. Partie projet

En termes de gestion de projet :

- J'ai joué un rôle central dans l'organisation technique, en définissant les priorités et l'ordre de développement des fonctionnalités.
- J'ai contribué activement à la gestion du dépôt GitHub, aux revues de code, et à la coordination avec mon binôme pour répartir équitablement la charge.
- J'ai pris des initiatives pour anticiper les problèmes techniques, notamment autour de l'intégration Azure et de la sécurité, et proposé des solutions (ex : usage de Key Vault, usage de Aspire, etc.).

4. Les IA génératives (GenIA) et moi

4.1. GenIA en développement IT

4.1.1. Outils utilisés

ChatGPT (OpenAI) → pour trouver des solutions concrètes, notamment sur la partie front-end Blazor (ex. composants, interactivité, gestion d'événements).

GitHub Copilot → pour l'aide à la saisie de code (auto-complétion), surtout en C# (back-end), Razor (front-end) et XAML (MAUI).

4.1.2. Cas d'utilisation

Utilisation de **ChatGPT** pour :

- Créer des pop-ups personnalisés dans Blazor, avec gestion d'ouverture/fermeture conditionnelle.
- Améliorer l'expérience utilisateur dans la web app (animations, transitions).
- Résoudre des erreurs de compilation ou de logique côté Razor.

Utilisation de **GitHub Copilot** pour :

- Générer du code C# dans les contrôleurs et services back-end.
- Suggérer des modèles de Data Annotations pour la validation.
- Accélérer l'écriture des vues XAML pour l'application MAUI.

4.1.3. Avantages

- Gain de temps sur les composants UI réutilisables et la logique répétitive.
- Assistance immédiate sans passer par la documentation ou StackOverflow.
- Qualité correcte pour des petits modules (ex. pop-up, alertes, messages conditionnels).
- **Copilot** a permis de réduire les erreurs de syntaxe et d'accélérer le développement sur les couches métiers.

4.1.4. Défis et limites

- Le code généré par **ChatGPT** pour Blazor était parfois non fonctionnel tel quel : nécessitait adaptation à notre structure.
- **Copilot** propose parfois du code hors-contexte, ou dupliquer des erreurs déjà présentes dans le fichier.
- Aucune IA ne remplace la compréhension du code : il faut toujours tester, corriger et adapter.
- Risque de se reposer trop dessus sans comprendre ce qu'on intègre.

4.1.5. Exemples concrets

- **Copilot** m'a aidé à compléter rapidement mon code lors d'actions répétitives.

- Pour afficher une pop-up de confirmation lors de la suppression d'un utilisateur dans la web app, j'ai demandé à **ChatGPT** un exemple de composant modal réutilisable en Blazor. Après quelques ajustements, j'ai intégré ce composant dans plusieurs parties de l'application (abonnement, clubs).

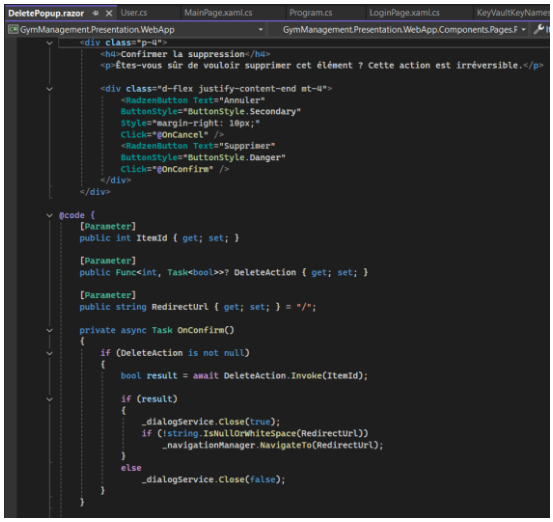

Figure 5 - Delete popup

4.2. Bilan personnel

Mon utilisation des IA génératives a été très bénéfique, surtout sur les parties que je maîtrisais moins, comme l'UI Blazor. Cela m'a permis de progresser plus vite, tout en gardant un contrôle total sur le code.

J'ai appris à travailler avec l'IA sans lui déléguer la logique métier, ce qui est pour moi un usage sain et professionnel.

4.3. Mise en perspective professionnelle

Ce projet m'a montré que l'IA peut devenir un assistant de développement efficace, à condition de garder une posture critique. Dans un environnement professionnel, cela peut accélérer le développement, mais aussi introduire des erreurs si elle est utilisée sans contrôle.

Je pense que savoir utiliser et remettre en question les suggestions d'IA sera une compétence clé pour les développeurs de demain.

5. Autres enseignements

L'importance de la structuration dès le début → démarrer un projet avec une architecture claire (DDD + Clean Architecture) nous a évité bien des erreurs en fin de développement. Même à deux, cette rigueur structurelle a été un vrai levier de productivité.

Apprendre sur le tas, c'est bien — anticiper, c'est mieux → apprendre MAUI en cours de projet a ralenti le développement de l'app mobile. Il aurait été plus productif de bloquer du temps en amont pour se former, même brièvement, sur les technos utilisées.

Les outils d'IA sont utiles… mais pas infaillibles → ChatGPT et Copilot m'ont réellement aidé, mais uniquement quand je savais quoi leur demander. Le code généré nécessite une relecture attentive, et ne remplace jamais la compréhension métier.

6. Conclusion

6.1. Bilan personnel

Ce que je ne veux plus jamais revivre dans un projet → travailler sous pression en fin de projet à cause de retards accumulés. Même si cela fait partie de la réalité du développement, cette période a été stressante. J'ai compris que prévoir des marges, même dans une petite équipe, est indispensable pour préserver la qualité du travail et la santé mentale.

Mon meilleur souvenir du projet → mon meilleur souvenir reste le moment où tout le système a fonctionné ensemble : l'API, la web app Blazor, l'app mobile MAUI, la base PostgreSQL sur Azure… Ce sentiment d'avoir construit une solution cohérente, interconnectée, et pleinement fonctionnelle était très gratifiant. C'est dans ces instants qu'on mesure tout ce qu'on a appris.

Ce que j'ai appris sur moi-même → j'ai réalisé que j'étais capable de prendre le lead technique sur un projet complet, même en effectif réduit. J'ai également renforcé ma capacité à résoudre des problèmes de manière autonome, et à m'adapter rapidement à de nouvelles technologies (comme MAUI).

6.2. Ce que j'ai vécu

Ce projet a été à la fois stimulant, formateur et exigeant. Travailler dans une équipe réduite (deux personnes) m'a confronté à une réalité proche de celle d'un développeur en startup ou en freelance : il faut tout gérer, du fonctionnel à l'architecture en passant par le déploiement et la sécurité.

J'ai beaucoup progressé dans des domaines que je ne maîtrisais pas au départ : développement mobile avec MAUI ou encore l'architecture logicielle avancée (Clean Architecture, DDD). Ce projet m'a permis de consolider mes acquis et de me projeter plus concrètement dans des situations proches du monde professionnel.

Je termine ce projet avec plus de confiance en mes compétences, mais aussi avec une meilleure conscience des efforts à fournir pour devenir un développeur professionnel complet.

7. Annexes

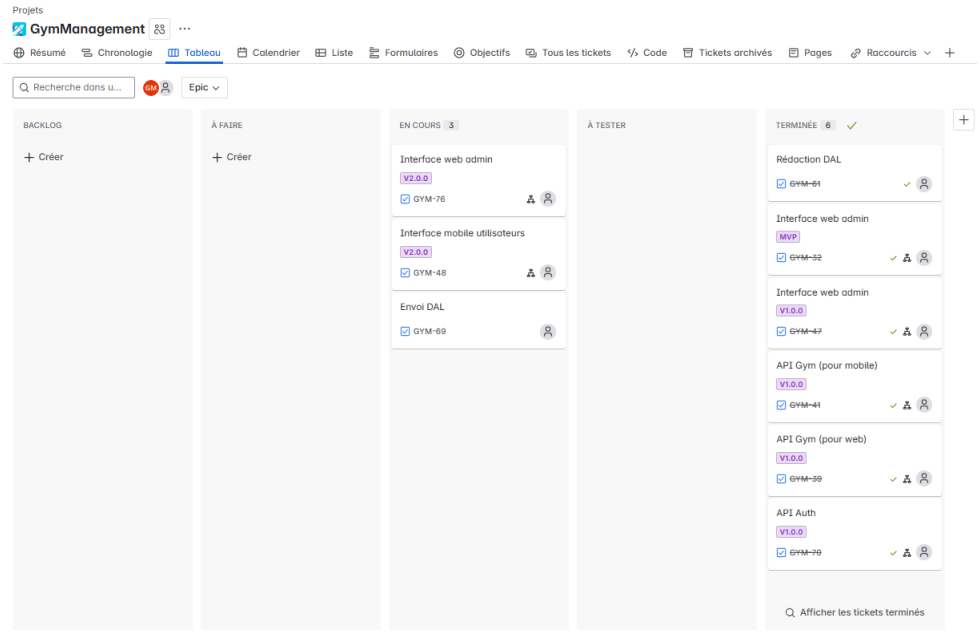

Figure 6 - Kanban (tâches)

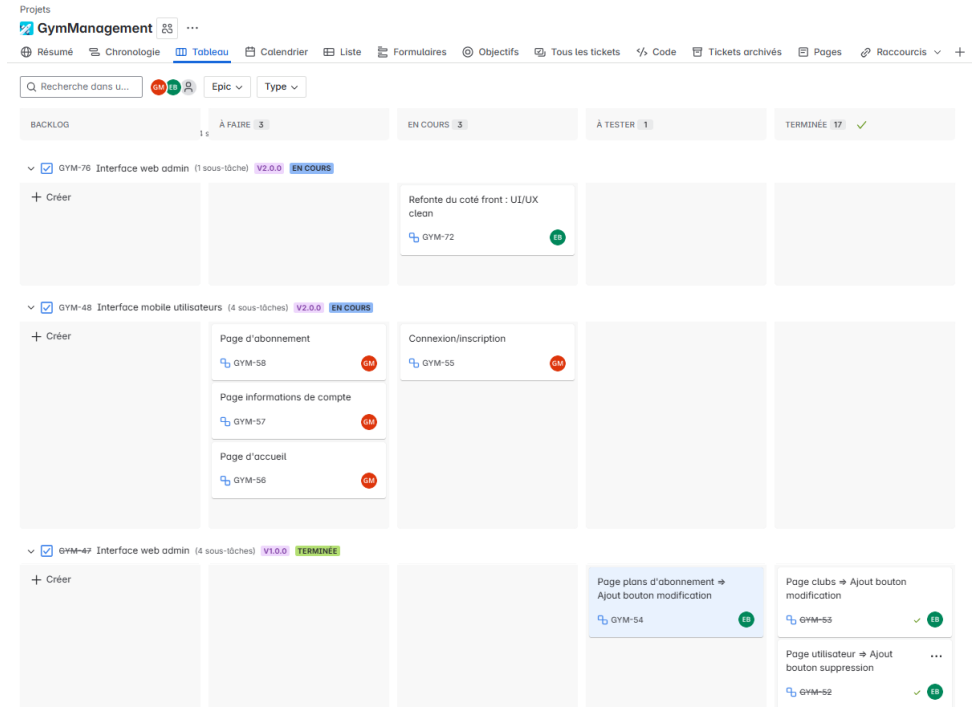

Figure 7 - Kanban (sous-tâches)