Projet Annuel ESGI B3

Logiciel de gestion de salle de sport

GymManagement

https://github.com/guilliammrst/GymManagement

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

Elouan LE BRAS

école supérieure de génie informatique

Contents

1. Analyse du projet

1.1. Contexte du projet

Ce projet s'inscrit dans le cadre du projet annuel de troisième année de Bachelor Informatique à l'ESGI Nantes. L'objectif principal était de développer nos compétences en gestion de projet et en développement logiciel dans un environnement collaboratif.

Le projet a été réalisé en binôme avec Guilliam MORISSET. Nous avons choisi de développer **GymManagement**, un logiciel de gestion de salle de sport. Ce choix s'explique par notre intérêt commun pour le domaine du sport et notre volonté de travailler sur une solution ayant une utilité concrète et moderne.

Le projet vise à créer une solution complète comprenant :

Une interface web d'administration pour les gestionnaires de salle

Une application mobile pour les utilisateurs finaux

Un système de gestion d'abonnements et d'accès par QR code

Une architecture moderne et sécurisée utilisant les technologies cloud

1.2. Chronologie du projet

Le projet a débuté le **jeudi 24 octobre 2024** avec la formation de l'équipe et la définition du sujet.

Phase de préparation (Octobre 2024 - Avril 2025) :

- Analyse Fonctionnelle du Besoin (AFB) avec utilisation de Miro
- Création d'une mindmap avec MindMeister pour clarifier la vision
- Définition des risques et des fonctionnalités
- Rédaction du cahier des charges
- Développement d'un POC (Proof of Concept) pour valider la faisabilité

Phase de validation : La présentation du projet à Nassim ZGA a eu lieu le **vendredi 25 avril 2025**, marquant le début officiel de la phase de mise en œuvre.

Phase de développement (Mai - Juillet 2025) :

- Mise en place du management de projet
- Développement itératif par versions successives
- Présentation finale le **mardi 22 juillet 2025**
- Livraison des livrables le **dimanche 13 juillet 2025**

Analyse des écarts de planification :

D'après les diagrammes de Gantt, plusieurs écarts significatifs ont été constatés :

- **MVP** : Initialement prévu pour mi-mai, livré fin mai (2 semaines de retard)
- **Version 1.0.0** : Prévue début juin, livrée fin juin (3 semaines de retard)
- **Version 2.0.0** : Encore en développement au moment de la rédaction de ce RETEX
- **Version 3.0.0** : Abandonnée par manque de temps

Ces retards s'expliquent principalement par :

- Notre exigence élevée sur la qualité architecturale du projet
- L'apprentissage en cours de projet de certaines technologies (.NET MAUI notamment)
- La complexité sous-estimée de certaines fonctionnalités

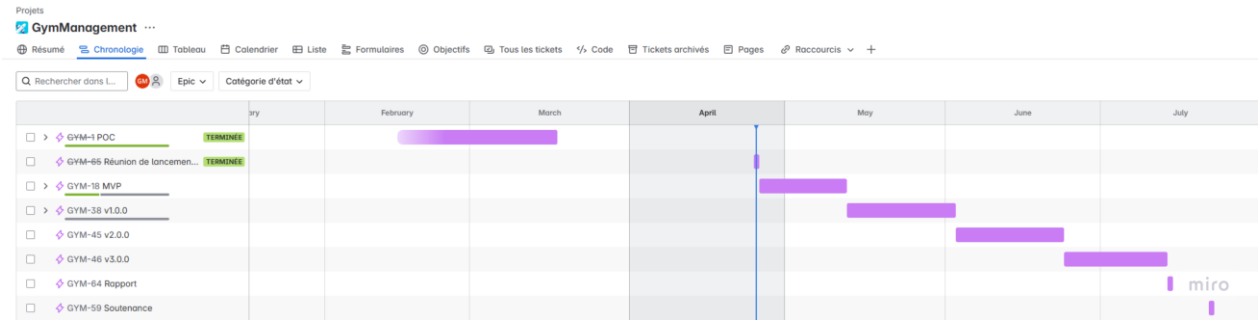

Figure 1 - Planification initiale

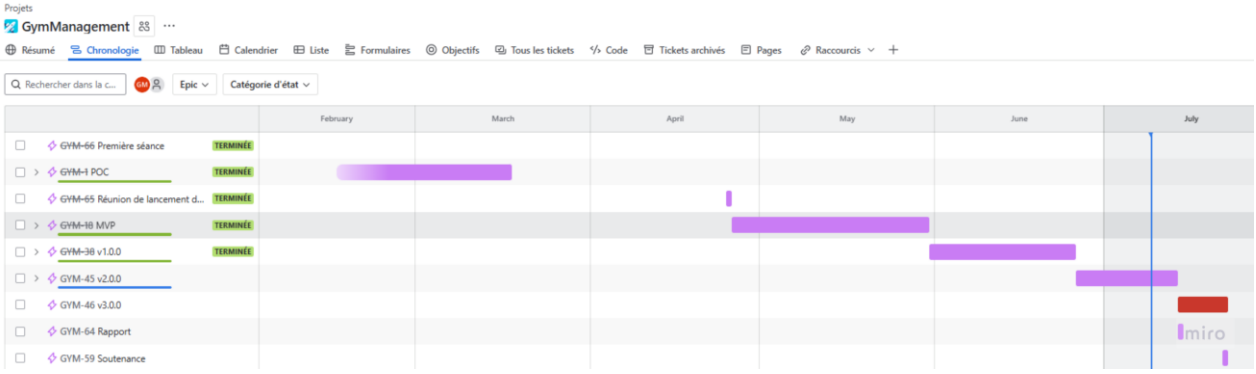

Figure 2 - Planification Finale

1.3. Travail prévu vs réalisé

Planification initiale des versions :

- **POC** : Connexion sécurisée via token à une interface web admin + intégration BDD et secrets via Azure + initialisation clean architecture

- **MVP** : Interface web permettant au staff d'ajouter des membres/clubs/plans d'abonnement et d'inscrire un utilisateur
- **Version 1.0.0** : Inscription utilisateur autonome, choix d'abonnement, paiement, accès par QR code via mobile, gestion de compte
- **Version 2.0.0** : Interface web accessible aux coachs, gestion des coachings, inscription aux coachings via mobile, statistiques
- **Version 3.0.0** (bonus) : Jobs Azure pour gestion des expirations, statistiques avancées, déploiement Azure avec Docker

Matrice de conformité finale :

Selon les données du tableau de bord Jira :

Statut	Nombre de tâches	Pourcentage
OK (Terminé)	22	~75%
En cours	5	~17%
NOK (Non réalisé)	4	~14%
Non prévu	3	~10%

Besoins ajoutés non prévus initialement :

- Refonte du côté front : UI/UX clean (conseillée par Nassim)
- Amélioration de l'expérience utilisateur mobile
- Optimisations de performance

Explications des écarts :

- **Abandon de la v3.0.0** : Choix stratégique pour se concentrer sur la qualité des versions précédentes
- **Retards de développement** : Complexité architecturale choisie (DDD + Clean Architecture) qui, bien que bénéfique long terme, a augmenté le temps de développement
- **Courbe d'apprentissage** : Apprentissage simultané de nouvelles technologies

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

L'organisation du projet s'est faite via Jira avec une méthodologie Kanban. Le tableau de bord était structuré en colonnes : **Backlog**, **À faire**, **En cours**, **À tester**, **Terminé**.

Répartition des responsabilités :

- **Guilliam** : Référent Jira et Lead développeur
- **Elouan** : Contributeur technique et rResponsable de la stratégie Git

Analyse de la répartition : Sur un total de 41 tickets créés, la répartition s'est faite de manière déséquilibrée avec Guilliam prenant en charge plus de 60% des tâches. Cette situation s'explique par :

- Des différences de disponibilité et de rythme de travail
- La complexité de certaines tâches nécessitant une expertise spécifique
- Des réajustements en cours de projet pour respecter les délais

Méthodologie de travail :

- Utilisation de **User Stories** pour exprimer les besoins métier
- Critères d'acceptation (CA) définis pour chaque ticket
- Suivi régulier de l'avancement via les rituels Agile

2. Bilan du projet

2.1. Liste des outils utilisés

2.1.1. Technique

Outil	Description	Comment l'avons-nous utilisé ?	Justification du choix	Bilan
Visual Studio 2022 Community	IDE officiel Microsoft pour C#/.NET	Développement, compilation, debug, intégration Git et NuGet	Interface complète, outils de debug puissants, extensions nombreuses	Bon choix
C#	Langage de programmation orienté objet Microsoft	Développement complet : API backend, webapp Blazor, app mobile MAUI	Cohérence de l'écosystème .NET, syntaxe moderne, typage fort	Bon choix
ASP.NET Core	Framework web pour API RESTful	Exposition des services REST consommés par les applications	Framework moderne, performant, documentation riche	Bon choix
.NET MAUI	Framework cross-platform pour applications mobiles	Développement de l'application mobile Android	Développement partagé entre plateformes, code C# unique	Améliorable - Framework jeune, bugs mineurs
Blazor/Razor	Framework pour interfaces web interactives en C#	Développement de l'interface web d'administration	Cohérence full-stack C#, bonne intégration backend	Bon choix
Aspire	Outil d'orchestration et observabilité .NET	Organisation en services, configuration centralisée, monitoring	Solution native .NET, simplification multi-services	Bon choix
PostgreSQL (Azure)	SGBD relationnel hébergé sur Azure	Base de données principale pour tous les services	Robuste, open-source, compatible .NET, scalabilité cloud	Bon choix
Azure Key Vault	Service cloud de gestion des secrets	Stockage sécurisé des chaînes de connexion et	Sécurité, pas de secrets en dur dans le code	Bon choix

Outil	Description	Comment l'avons-nous utilisé ?	Justification du choix	Bilan
		secrets		

2.1.2. Projet

Outil	Description	Comment l'avons-nous utilisé ?	Justification du choix	Bilan
Jira	Outil de gestion de projet	Attribution des tâches, suivi d'avancement, user stories	Centralisation, méthodologie Agile, suivi précis	Bon choix
GitHub	Plateforme de versioning Git	Hébergement du code, pull requests, gestion des branches	Standard du marché, intégration Visual Studio	Bon choix
Discord	Plateforme de communication	Calls réguliers, partage d'écran, entraide technique	Facilité d'utilisation, fonctionnalités riches	Bon choix
Snapchat	Réseau social	Communication rapide et informelle	Plus réactif que Discord pour les messages courts	Bon choix
Miro	Dashboard de collaboration	Centralisation des documents, brainstorming, AFB	Collaboration visuelle, partage de ressources	Bon choix
MindMeister	Outil de mindmapping	Structuration des idées initiales du projet	Clarification de la vision, brainstorming libre	Bon choix

2.2. Points positifs à capitaliser

2.2.1. Partie technique

Architecture robuste : L'implémentation d'une architecture combinant Domain Driven Design (DDD) et Clean Architecture s'est révélée très bénéfique :

- Séparation claire des responsabilités
- Code centré sur le métier
- Maintenabilité et testabilité accrues
- Indépendance vis-à-vis des technologies externes

Intégration cloud native : L'utilisation d'Azure Key Vault et PostgreSQL Azure a apporté :

- Sécurité renforcée (pas de secrets en dur)
- Scalabilité native
- Monitoring intégré

Outils de développement :

- Visual Studio 2022 : stabilité et fonctionnalités complètes
- Aspire : visualisation des dépendances, configuration simplifiée
- GitHub : workflow de développement fluide avec pull requests

Transfert de compétences :

- **Backend et POO** : Expérience très enrichissante, parallèles intéressants avec mon travail en Python backend en alternance

2.2.2. Partie projet

Méthodologie Agile efficace :

- User Stories claires avec critères d'acceptation
- Kanban adapté à notre effectif réduit
- Suivi régulier via Jira permettant une bonne visibilité

Communication d'équipe :

- Points réguliers maintenant la cohésion
- Entraide technique constante
- Réactivité face aux blocages

Gestion des versions :

- Approche itérative avec livraisons successives
- Validation continue des fonctionnalités
- Ajustements possibles à chaque itération

2.3. Points de vigilance à améliorer

2.3.1. Partie technique

Courbe d'apprentissage sous-estimée : .NET MAUI étant un framework relativement jeune, nous avons rencontré :

- Bugs mineurs ralentissant le développement
- Documentation parfois insuffisante
- Nécessité d'adapter les solutions trouvées

Estimation initiale trop optimiste :

- Sous-estimation de la complexité architecturale
- Temps de développement UI mobile mal évalué
- Intégration des services cloud plus complexe que prévu

Écart de compétences techniques :

- **Manque d'expérience C# de mon coté** : Courbe d'apprentissage importante, d'où le lead technique naturel de Guilliam

2.3.2. Partie projet

Répartition des tâches déséquilibrée : Avec un effectif réduit (2 personnes), nous avons rencontré :

- Concentration de la charge sur un membre
- Difficulté à maintenir un rythme constant
- Nécessité de réajustements fréquents

Aléas projet :

- Retards dus à la complexité technique
- Changements de périmètre en cours de projet (refonte UI)
- Gestion des priorités parfois difficile

Contraintes d'alternance :

- **Cassures de rythme** : Rythme 3 semaines entreprise / 1 semaine école créant des interruptions dans la dynamique projet

3. Mon apport à l'équipe

3.1. Partie technique

3.1.1. Liste des fonctionnalités développées

Modules développés :

- **Gestion des clubs** : API complète + interface d'administration
- **Plans d'abonnement** : Système complet de création et gestion des plans
- **Abonnements utilisateurs** : Souscription, paiement et historique
- **Refonte UI/UX** : Interface plus clair et design

3.1.2. Activités techniques prises en charge

Stack technique :

- **Backend** : Architecture DDD, repositories, services, contrôleurs API REST
- **Frontend** : Composants Blazor, pages d'administration, système de modales
- **Intégration** : ApiClients, authentification, services Azure

3.2. Partie projet

Rôle dans l'équipe :

- **Contributeur technique** : Modules clubs, plans d'abonnement, refonte UI/UX
- **Gestion Git** : Stratégie de branches, résolution des conflits, revues de code
- **Coordination Miro** : Centralisation des documents, brainstorming, AFB

Initiatives :

- Refonte UI/UX complete (HTML/CSS pur)
- Système de notifications et modales custom
- Veille technologique et solutions alternatives

4. Les IA génératives (GenIA) et moi

4.1. GenIA en développement IT

4.1.1. Outils utilisés

ChatGPT (OpenAI) :

- Utilisation pour la résolution de problèmes techniques spécifiques
- Aide à la compréhension de concepts complexes
- Génération d'exemples de code pour des cas d'usage précis

GitHub Copilot :

- Assistant de codage intégré à Visual Studio
- Auto-complétion intelligente
- Suggestions de code contextuel

4.1.2. Cas d'utilisation

Avec ChatGPT :

- Résolution d'erreurs de compilation complexes
- Aide à la compréhension des patterns architecturaux (DDD, Clean Architecture)
- Génération de code pour les entités et services métier
- Optimisation des requêtes LINQ et Entity Framework

Avec GitHub Copilot :

- Génération de code répétitif (DTOs, mappers, validations)
- Suggestions pour les contrôleurs API REST
- Aide à l'écriture de tests unitaires
- Complétion de code XAML pour l'application mobile

4.1.3. Avantages

Gain de productivité :

- Réduction du temps passé sur le code répétitif
- Accélération du développement des fonctionnalités de base
- Aide immédiate sans recherche documentaire

Apprentissage accéléré :

- Compréhension plus rapide de nouvelles technologies
- Exemples concrets et adaptés au contexte
- Explications détaillées des concepts complexes

Résolution de problèmes :

- Assistance pour déboguer des erreurs complexes
- Suggestions d'optimisation du code
- Alternatives techniques proposées

4.1.4. Défis et limites

Qualité variable du code généré :

- Nécessité de relecture et validation systématique
- Code parfois non optimisé ou non conforme aux standards
- Risque d'introduction de bugs subtils

Dépendance excessive :

- Risque de ne pas développer ses propres compétences
- Tendance à accepter les solutions sans les comprendre
- Perte d'autonomie dans la résolution de problèmes

Limites contextuelles :

- Difficulté à comprendre l'architecture globale du projet
- Suggestions parfois hors-contexte
- Nécessité d'adapter les solutions au projet spécifique

4.1.5. Exemples concrets

Exemple 1 - Génération d'entités métier :

```
// Code généré par ChatGPT et adapté
public class User : Entity<UserId>
{
    public string Email { get; private set; }
    public string HashedPassword { get; private set; }
    public UserRole Role { get; private set; }

    // Méthodes métier ajoutées manuellement
    public void ChangePassword(string newPassword)
    {
        // Logique de validation et hashage
    }
}
```

Exemple 2 - Aide au débogage : Lors d'un problème de configuration Entity Framework, ChatGPT a fourni la solution pour configurer correctement les relations many-to-many avec les données de jonction.

4.2. Bilan personnel

Ressenti général : Mon expérience avec les IA génératives a été majoritairement positive. Elles ont constitué un véritable accélérateur de développement, particulièrement sur les aspects techniques que je maîtrisais moins au début du projet.

Attentes vs. réalité : Mes attentes initiales étaient peut-être trop élevées concernant la capacité de l'IA à comprendre le contexte global du projet. En réalité, l'IA excelle sur des tâches spécifiques et localisées, mais nécessite une supervision humaine pour l'architecture globale et la logique métier complexe.

Évolution de mon approche : J'ai appris à utiliser l'IA comme un **assistant**, pas comme un **remplaçant**. La clé est de savoir quand l'utiliser et surtout comment valider et adapter ses suggestions.

4.3. Mise en perspective professionnelle

Impact sur mes compétences :

- **Accélération** de l'apprentissage de nouvelles technologies
- **Amélioration** de ma capacité à résoudre des problèmes complexes
- **Développement** d'un esprit critique vis-à-vis des solutions proposées

Impact sur le travail : Les IA génératives transforment la façon de travailler en développement :

- Moins de temps sur les tâches répétitives
- Plus de temps sur la conception et l'architecture
- Nécessité de nouvelles compétences : savoir formuler des prompts efficaces

Réflexions pour l'avenir : Dans un environnement professionnel, maîtriser l'utilisation des IA génératives sera probablement un avantage concurrentiel. Cependant, cela nécessite de maintenir un équilibre entre assistance et autonomie technique.

5. Autres enseignements

L'importance de l'architecture dès le début : Investir du temps dans une architecture solide (DDD + Clean Architecture) s'est révélé payant malgré le ralentissement initial. Cette approche nous a évité de nombreux problèmes techniques en fin de projet et a facilité la maintenance du code.

La gestion du périmètre dans un projet à effectif réduit : Travailler à deux nous a appris l'importance de bien définir le périmètre et de savoir dire "non" à certaines fonctionnalités. La v3.0.0 abandonnée en est un exemple : mieux vaut livrer moins mais de qualité.

L'apprentissage continu comme nécessité : Ce projet m'a confirmé que le développement moderne nécessite un apprentissage constant. Partir avec peu d'expérience C# pour découvrir .NET MAUI, l'architecture DDD, les services Azure… La transition depuis Python backend m'a permis de voir les similitudes et différences entre les écosystèmes.

La valeur de la communication dans une petite équipe : Même à deux, maintenir une communication régulière et structurée est essentiel. Les points Discord réguliers ont permis de garder le cap malgré les difficultés.

L'équilibre entre innovation et pragmatisme : Nous avons parfois été trop ambitieux techniquement. Savoir équilibrer innovation (nouvelles technologies) et pragmatisme (solutions éprouvées) est un apprentissage clé.

6. Conclusion

6.1. Bilan personnel

Ce que je ne veux plus jamais revivre : La période de stress intense en fin de projet due à l'accumulation de retards. Cette expérience m'a appris l'importance cruciale de la planification réaliste et des marges de sécurité, même dans une petite équipe.

Mon meilleur souvenir : L'instant où toute la stack technique a fonctionné ensemble pour la première fois : l'utilisateur pouvaitse connecter, le système généraient un QR code, et l'admin pouvait valider l'accès via l'interface web. Ce moment d'accomplissement technique total reste un souvenir fort.

Découvertes sur moi-même :

- Capacité d'adaptation rapide à de nouvelles technologies
- Aptitude à maintenir la qualité technique même sous pression
- Goût pour l'architecture logicielle et les défis techniques complexes

6.2. Ce que j'ai vécu

Expérience technique : Ce projet m'a permis de découvrir l'écosystème .NET moderne dans sa globalité. De C# à Azure en passant par MAUI, j'ai acquis une vision complète du développement full-stack Microsoft.

Expérience humaine : Travailler en binôme intensif pendant 8 mois crée des liens forts et apprend la collaboration technique. J'ai appris à gérer les différences de rythme et à maintenir la cohésion d'équipe.

Expérience professionnelle : Ce projet m'a donné un aperçu concret du développement logiciel professionnel : gestion des deadlines, qualité du code, architecture évolutive, gestion des priorités. L'alternance 3 semaines/1 semaine m'a appris à gérer les projets par intermittence tout en maintenant la continuité technique.

Préparation à l'avenir : Je me sens désormais plus confiant pour aborder des projets complexes et j'ai acquis des compétences directement transférables en entreprise.