



MÉMOIRE DE PROJET DE FIN D'ÉTUDES POUR L'OBTENTION DU TITRE  
D'INGÉNIEUR D'ÉTAT EN INFORMATIQUE

FILIERE GÉNIE LOGICIEL

---

# Mise en place d'un Système de gestion de temps d'activités

---

*Réalisé par :*  
Oussama BAKRI

*Encadré par :*  
Pr. Mahmoud NASSAR  
Pr. Mahmoud EL HAMLAOUI  
Pr. Bouchaib BOUNABAT  
M. Amine BENAZZOUZ

Année Universitaire 2023/2024

# Dédicaces

## **A mes chers Parents,**

Nulle dédicace ne peut décrire ce que je ressens pour vous. Merci, pour votre confiance, votre amour, votre patience, vos encouragements et vos sacrifices tout au long de mon parcours, que Dieu vous garde.

## **A mon frère et mes sœurs,**

Je vous remercie pour votre amour inconditionnel. Que dieu vous garde, Je vous aime et je vous souhaite une vie pleine de succès et de prospérité.

## **A toute ma famille,**

Source d'inspiration et de tendresse.

## **A tous mes amis de l'ENSIAS,**

Pour tout le soutien que vous m'avez offert, je vous dis MERCI.

A tous ceux qui ont contribué de près ou de loin à la réussite de ce projet, veuillez trouver ici le respect et la reconnaissance que j'éprouve pour vous.

# Remerciement

Je profite par le biais de ce rapport, pour exprimer mes vifs remerciements à toute personne contribuant de près ou de loin à l'élaboration de cet humble travail.

Je tiens à remercier vivement le corps professoral de l'École Nationale Supérieure d'Informatique et d'Analyse des Systèmes, et spécialement Monsieur Bouchaib BOUNABAT, mon professeur encadrant, qui a contribué à la réalisation de ce modeste projet, qui m'a encadré et aidé tout au long de mon parcours.

Un merci bien particulier adressé également à Monsieur Amine BENAZZOUZ, mon encadrant au sein de DataTeam, pour ses remarques et ses directives. Je tiens à lui exprimer mes sincères remerciements pour son suivi et ses orientations.

Je tiens à remercier aussi tout le personnel de l'entreprise DataTeam, qui m'ont accueilli, conseillé et soutenu le long de mon stage. Veuillez trouver ici l'expression de notre parfaite considération.

# Résumé

Ce rapport présente le travail réalisé dans le cadre du projet de fin d'études pour l'obtention du titre d'Ingénieur d'état en Génie logiciel. Le stage s'est déroulé au sein de l'entreprise DataTeam et avait pour objectif principal la conception et la mise en œuvre d'un Système de gestion du temps et des activités.

Le projet visait à développer une solution permettant aux entreprises de gérer efficacement le temps et les activités de leurs employés. L'objectif était de créer un outil convivial et performant, capable de s'adapter aux besoins spécifiques de différentes organisations.

La démarche adoptée a commencé par une analyse approfondie des besoins, permettant de comprendre les attentes des utilisateurs et de définir les fonctionnalités essentielles du système. Une attention particulière a été portée à l'architecture technique, afin de garantir la scalabilité et la performance de la solution. Le développement de l'application a été réalisé en utilisant des technologies modernes, assurant ainsi sa pérennité et sa facilité de maintenance.

Tout au long du projet, l'accent a été mis sur la création d'une interface utilisateur intuitive et sur l'implémentation de fonctionnalités répondant aux exigences des entreprises en matière de gestion du temps et des activités.

Ce stage a offert une opportunité précieuse de mettre en pratique les connaissances théoriques acquises au cours de la formation d'ingénieur, tout en développant de nouvelles compétences techniques et professionnelles. L'expérience acquise dans la conception et le développement d'une application d'entreprise complexe constitue une base solide pour une future carrière dans le domaine du développement logiciel.

**Mots-clés :** Système de gestion de temps d'activités, Méthodologie, architecture technique, Interface Utilisateur

# Abstract

This report presents the work carried out as part of the final year project for obtaining the State Engineering degree in Software Engineering. The internship took place at DataTeam company and had as its main objective the design and implementation of a Time and Activity Management System.

The project aimed to develop a solution allowing companies to efficiently manage their employees' time and activities. The goal was to create a user-friendly and high-performance tool, capable of adapting to the specific needs of different organizations.

The adopted approach began with an in-depth analysis of needs, allowing for understanding user expectations and defining the essential functionalities of the system. Particular attention was paid to the technical architecture to ensure the scalability and performance of the solution. The application development was carried out using modern technologies, thus ensuring its durability and ease of maintenance.

Throughout the project, emphasis was placed on creating an intuitive user interface and implementing features that meet business requirements for time and activity management.

This internship offered a valuable opportunity to put into practice the theoretical knowledge acquired during the engineering training, while developing new technical and professional skills. The experience gained in designing and developing a complex enterprise application provides a solid foundation for a future career in software development.

**Keywords :** Activity Time Management System, Methodology, Technical Architecture, User Interface

# Liste des abréviations

GTA	Gestion de Temps d'Activités
API	Application Programming Interface
HTTPS	HyperText Transfer Protocol Secure
IHM	Interface Homme Machine
IDE	Integrated Development Environment
CRUD	Create, Read, Update, Delete
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
JSON	JavaScript Object Notation
LINQ	Language Integrated Query
ORM	Object-Relational Mapping
REST	Representational State Transfer
UML	Unified Modeling Language
DI	Dependency Injection
DDD	Domain-Driven Design
SSR	Server-Side Rendering
SQL	Structured Query Language
SGDB	Système de gestion de base de données

# Table des matières

<b>Introduction Générale</b>	<b>12</b>
<b>1 Contexte général du projet</b>	<b>13</b>
1.1 Introduction . . . . .	13
1.2 Présentation de l'organisme . . . . .	13
1.2.1 Informations générales . . . . .	13
1.2.2 Secteur d'activité . . . . .	14
1.2.3 Partenariats et Clients . . . . .	15
1.3 Présentation du projet . . . . .	16
1.3.1 Problématique . . . . .	16
1.3.2 Description générale du projet . . . . .	16
1.4 Conduite du projet . . . . .	17
1.4.1 Équipe du projet . . . . .	17
1.4.2 Méthodologie adoptée : Développement en Y . . . . .	18
1.4.3 Planification du projet . . . . .	19
1.5 Conclusion . . . . .	20
<b>2 Analyse et Conception</b>	<b>21</b>
2.1 Introduction . . . . .	21
2.2 Authentication/Authorization . . . . .	21
2.2.1 Structure des Rôles . . . . .	22
2.2.2 Processus d'Authentification . . . . .	22
2.2.3 Gestion des Autorisations . . . . .	22
2.2.4 Intégration avec l'Architecture Multi-Tenant . . . . .	23
2.3 les cas d'utilisation . . . . .	23

2.3.1	Gestion du profil . . . . .	23
2.3.2	Cas d'utilisation « CRUD sur les entités » . . . . .	26
2.3.3	Cas d'utilisation « Calculer la synthèse » . . . . .	26
2.3.4	Cas d'utilisation « exécuter des Jobs » . . . . .	27
2.3.5	Diagramme de Cas d'utilisation . . . . .	27
2.4	Diagramme de classes . . . . .	29
2.5	Diagramme de séquence . . . . .	31
2.6	Conclusion . . . . .	32
<b>3</b>	<b>Etude technique</b>	<b>34</b>
3.1	Introduction . . . . .	34
3.2	Contraintes techniques . . . . .	34
3.3	Architecture physique : Multi-Tenant . . . . .	35
3.3.1	Introduction au Multi-Tenancy . . . . .	35
3.3.2	Avantages du Multi-Tenancy . . . . .	36
3.3.3	Multi-Tenant Architecture Design . . . . .	36
3.4	Architecture logicielle . . . . .	39
3.4.1	Clean Architecture . . . . .	39
3.4.2	Domain-Driven Design (DDD) . . . . .	41
3.4.3	Generic Repository . . . . .	42
3.5	Conclusion . . . . .	43
<b>4</b>	<b>Mise en Œuvre</b>	<b>45</b>
4.1	Introduction . . . . .	45
4.2	Environnement et outils de développement . . . . .	46
4.2.1	Langage C# . . . . .	46
4.2.2	Visual Studio 2022 . . . . .	47
4.2.3	La plateforme .NET . . . . .	48
4.2.4	Frameworks utilisés . . . . .	49
4.2.5	Microsoft SQL Server . . . . .	54
4.2.6	Git/Github . . . . .	55
4.3	Les IHM de L'application . . . . .	56



4.4	Conclusion . . . . .	64
-----	----------------------	----

# Table des figures

1.1	Diagramme de structure de l'organisme d'accueil . . . . .	14
1.2	Queleques Partenaires et Clients de DataTeam . . . . .	15
1.3	Les membres de l'équipe de développement . . . . .	17
1.4	Cycle de développement en Y . . . . .	19
1.5	Diagramme de Gantt . . . . .	20
2.1	Diagramme de Cas d'utilisation . . . . .	28
2.2	Diagramme de classes . . . . .	30
2.3	Diagramme de séquence de la création et de l'initialisation d'un Tenant . . . . .	32
3.1	Architecture multitenant . . . . .	35
3.2	Clean Architecture . . . . .	40
3.3	Generic Repository . . . . .	43
4.1	Logo du Langage de programmation C# . . . . .	47
4.2	Logo de Visual Studio 2022 . . . . .	48
4.3	Logo de la plateforme .NET . . . . .	49
4.4	Logo de Blazor . . . . .	50
4.5	Logo de Entity Framework Core . . . . .	52
4.6	Logo de Serilog . . . . .	53
4.7	Logo de Postman . . . . .	53
4.8	Logo de Microsoft SQL Server . . . . .	55
4.9	Logo de Git et Github . . . . .	56
4.10	Page de connexion . . . . .	56
4.11	Liste des tenants . . . . .	57
4.12	Création d'un tenant . . . . .	57

4.13	Liste des organisation . . . . .	58
4.14	Création d'un organisation . . . . .	58
4.15	List des utilisateurs . . . . .	59
4.16	Création des utilisateurs . . . . .	59
4.17	List des Shifts . . . . .	60
4.18	Création d'un Shift . . . . .	60
4.19	Détails de Shift . . . . .	61
4.20	Création d'un Employé . . . . .	61
4.21	Liste des Employé . . . . .	62
4.22	Détails d'un Employé . . . . .	62
4.23	Création de clock device . . . . .	63
4.24	Liste des clocks device . . . . .	63
4.25	Liste des Time Logs . . . . .	64

# Liste des tableaux

2.1	Description du Cas d'utilisation « Créer un compte » . . . . .	24
2.2	Description du Cas d'utilisation « S'authentifier » . . . . .	24
2.3	Description du Cas d'utilisation « Modifier un compte utilisateur » . . . . .	25
2.4	Description du Cas d'utilisation « Afficher le profil de l'utilisateur » . . . . .	25
2.5	Description du Cas d'utilisation « Les opérations CRUD » . . . . .	26
2.6	Description du Cas d'utilisation « Calculer la synthèse » . . . . .	26
2.7	Description du Cas d'utilisation « exécuter des Jobs » . . . . .	27

# Introduction Générale

Ce rapport présente le fruit d'un projet de fin d'études réalisé au sein de Datateam.

Ce stage s'inscrit dans la continuité de la formation académique en Génie logiciel, offrant une opportunité unique de mettre en pratique les connaissances théoriques acquises et de les confronter aux réalités du monde professionnel. L'objectif principal de ce stage est de mettre en place un système de gestion de temps d'activités, un défi qui a nécessité l'application de compétences variées en développement logiciel et en gestion de projet.

Au cours de cette expérience, le travail a porté sur un projet concret, depuis sa phase d'analyse et de conception jusqu'à sa mise en œuvre technique. Ce processus a impliqué l'utilisation de méthodologies modernes de développement, l'exploration de concepts avancés en architecture logicielle et l'application de technologies de pointe dans le domaine du développement .NET.

Ce document vise à offrir un aperçu complet du travail effectué, des défis rencontrés et des solutions apportées. Ce rapport souligne l'importance de cette expérience pratique dans le parcours professionnel, tout en mettant en lumière la synergie entre formation académique et application concrète dans un environnement professionnel.

Ce stage de fin d'études représente non seulement l'aboutissement de la formation, mais aussi le point de départ de la carrière dans le monde du développement logiciel, offrant une base solide pour la future évolution professionnelle.

# Chapitre 1

## Contexte général du projet

### 1.1 Introduction

Ce chapitre vise à mieux cerner les exigences du système à développer, ainsi qu'à situer le projet dans son environnement organisationnel et contextuel. Il présente un aperçu de l'organisme d'accueil, DataTeam, ainsi que du thème général du projet, en introduisant ses objectifs et son cahier des charges. Il décrit ensuite les différentes démarches appliquées, notamment la méthodologie préconisée et la planification qui en résulte.

### 1.2 Présentation de l'organisme

#### 1.2.1 Informations générales

Depuis plus de 20 ans, le Groupe DATALINK déploie son expertise dans l'intégration des solutions de gestion, avec une parfaite maîtrise des enjeux business grâce à sa double compétence métier et technologique.

La société DATATEAM a été créée en 2012 spécialement pour intégrer toutes les activités en solutions de gestion du groupe DATALINK. Dans le cadre du développement des ressources de prestations de service, toutes les compétences sur les logiciels de gestion ont été concentrées dans la société DATATEAM Sarl, filiale à 100% du groupe DATALINK.

DATATEAM contribue à l'amélioration de la performance des opérations de ses clients en les aidant à choisir et à mettre en place les technologies qui leur conviennent le mieux. L'entreprise propose une gamme diversifiée de solutions performantes, spécialement conçues pour répondre aux besoins uniques de ses clients. Ces solutions sont mises en œuvre par une équipe d'ingénieurs et de techniciens hautement compétents, tous certifiés par des éditeurs et des fabricants

de renommée mondiale.

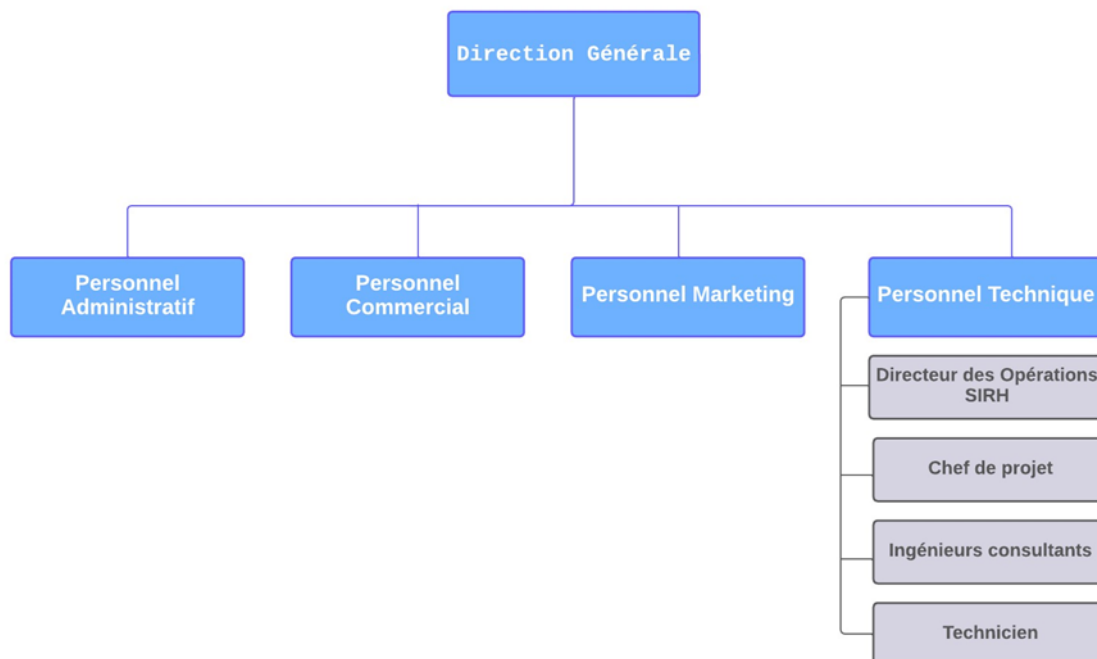


FIGURE 1.1 – Diagramme de structure de l'organisme d'accueil

### 1.2.2 Secteur d'activité

DATATEAM est une entreprise spécialisée dans l'intégration de solutions informatiques. Dotée d'une expertise avancée dans les systèmes HP, Microsoft, Oracle, etc, DATATEAM propose des services de consultation, déploiement, et support technique dans divers domaines, tels que l'infrastructure informatique, l'architecture réseau, les systèmes d'exploitation et les bases de données.

Son engagement envers l'excellence se reflète également dans ses missions d'audit et d'évaluation des systèmes d'informations. De l'audit de sécurité à l'évaluation de la performance du système, DATATEAM assure la mise en place de systèmes de gestion d'incidents et la rédaction de cahiers des charges pour des projets informatiques variés.

En tant que fournisseur de solutions professionnelles, DATATEAM propose une gamme complète de formations certifiées, couvrant les solutions Microsoft, les logiciels de gestion tels que SAGE et SUN, ainsi que des solutions spécialisées comme DORMAKABA pour la gestion du

temps et des activités.

Par ailleurs, DATATEAM se distingue par son expertise en intégration, services, et conseils en solutions de gestion intégrées et d'ERP. Que ce soit dans les domaines de la finance avec SAGE et ERP X3, de la gestion des ressources humaines avec HR ACCESS, ou de la gestion hôtelière avec MICROS et FIDELIO, DATATEAM assure une prise en charge complète des besoins de ses clients.

En outre, l'entreprise s'engage également dans le domaine du développement, offrant des solutions sur mesure pour répondre aux besoins spécifiques de ses clients et garantir leur succès dans un environnement informatique en constante évolution.

### 1.2.3 Partenariats et Clients

DataTeam, durant ses années d'existence, a su bâtir un réseau solide de clients prestigieux et de partenariats stratégiques. Parmi ses clients les plus notables, on compte La Mamounia, Accor, RISMA, IBIS, et SOFITEL. Ces collaborations témoignent de la capacité de DataTeam à répondre aux exigences variées de ses clients et à fournir des services de haute qualité.



FIGURE 1.2 – Quelques Partenaires et Clients de DataTeam



## **1.3 Présentation du projet**

Ce travail s'inscrit dans le cadre du Projet de Fin d'Études pour l'obtention du titre d'Ingénieur d'État en Génie Logiciel. Réalisé dans le cadre d'un stage au sein de l'entreprise Datateam à Casablanca, le projet GTA (Gestion du Temps d'Activité) vise à concevoir et à mettre en œuvre un système de gestion du temps d'activité. Ce système, destiné à répondre aux besoins spécifiques des clients, offre une solution efficace et adaptée à leurs exigences.

### **1.3.1 Problématique**

L'objectif principal du projet GTA est de développer des fonctionnalités avancées et sécurisées de gestion du temps d'activité, permettant aux utilisateurs de gérer efficacement les plannings et les dossiers, tout en garantissant une expérience utilisateur fluide et adaptée. En exploitant les dernières technologies, le projet GTA vise à simplifier les processus de gestion du temps et à fournir une solution robuste et évolutive pour répondre aux divers besoins de ses utilisateurs.

### **1.3.2 Description générale du projet**

Ce projet vise à développer une application dédiée à la gestion du temps d'activité (GTA) pour un ensemble diversifié de clients. L'application GTA facilitera la gestion des horaires et des activités pour les utilisateurs. Les fonctionnalités avancées de l'application permettent aux utilisateurs de planifier et de gérer les activités en toute simplicité, contribuant ainsi à une gestion plus efficace du temps et des ressources.

## 1.4 Conduite du projet

Dans un projet informatique, il est fondamental de se baser sur un cycle de développement sur lequel s'articule l'ensemble des solutions afin de détecter les erreurs au plus tôt et ainsi de maîtriser la qualité du logiciel, les délais de sa réalisation et les coûts associés. Le cycle de vie d'un logiciel désigne toutes les étapes du développement d'un logiciel, de sa conception à sa livraison. L'objectif d'un tel découpage est de définir des jalons intermédiaires permettant la validation du développement logiciel, c'est-à-dire la conformité du logiciel avec les besoins exprimés, et la vérification du processus de développement.

### 1.4.1 Équipe du projet

L'équipe DataTeam est pilotée par Monsieur Amine BENAZZOUZ. Elle se compose d'un stagiaire et d'un ingénieur développeur.

La figure suivante représente les collaborateurs du projet :

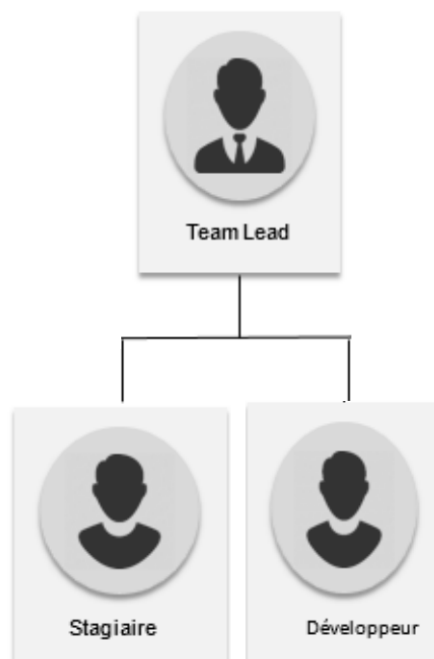


FIGURE 1.3 – Les membres de l'équipe de développement

### 1.4.2 Méthodologie adoptée : Développement en Y

Un processus de développement définit une séquence d'étapes, en partie ordonnée, qui concourt à l'obtention d'un système logiciel nouveau ou à l'évolution d'un système existant. Ce processus a pour objectifs de produire des solutions informatiques de qualité répondant aux besoins des utilisateurs dans des délais et des coûts prévisibles.

De ce fait, l'adéquation du projet au processus de développement peut largement affecter le sort d'un projet informatique. Pour éviter tout risque, le processus 2TUP a été choisi, car il s'adapte le mieux au projet. En effet, ce projet présente une difficulté technique et fonctionnelle élevée. La difficulté technique réside dans l'utilisation des technologies de pointes. Quant à la difficulté fonctionnelle, elle réside essentiellement dans la gestion des interactions complexes entre différents modules logiciels, nécessitant une coordination précise et une intégration harmonieuse pour assurer le bon fonctionnement de l'ensemble du système. Cette description doit porter sur les multiples fonctionnalités offertes par le système. Ainsi, pour faire face à cette complexité, le processus 2TUP propose un cycle de développement en Y, qui dissocie les aspects techniques des aspects fonctionnels, ce qui permet de gérer les risques que présente le projet. La figure 1.4 représente le cycle de développement en Y.

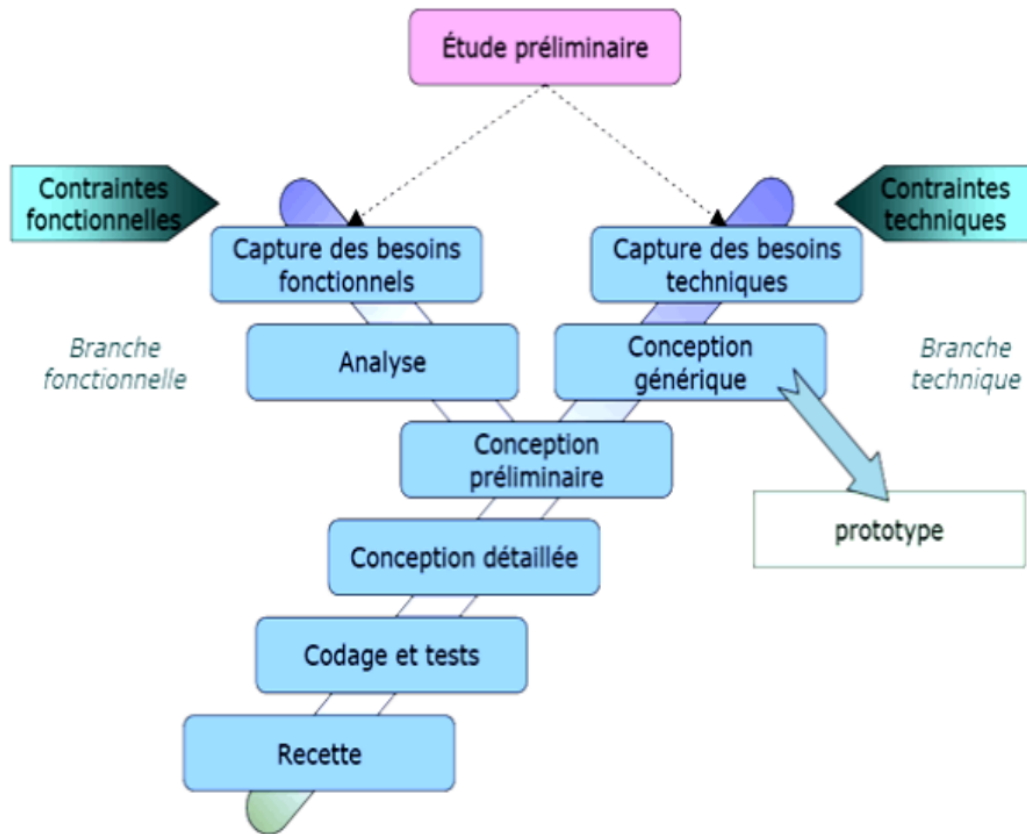


FIGURE 1.4 – Cycle de développement en Y

Le processus en Y ou Two Track Unified Process (2TUP) est un processus qui s’articule autour de l’architecture. Il est constitué de deux branches, l’une fonctionnelle et l’autre technique. Celles-ci se rencontrent dans la partie de la réalisation, d’où son appellation de cycle en Y. L’étude fonctionnelle a pour objectif de capturer les besoins fonctionnels en termes de fonctionnalités que doit remplir le système cible. L’étude technique, quant à elle, précise les contraintes techniques à prendre en considération, et propose une architecture logicielle et applicative qui répond aux contraintes dégagées. La phase suivante de conception consiste à reprendre le modèle d’analyse et le refaire selon les décisions prises dans la branche technique. Il s’agit donc d’adapter le modèle d’analyse à l’architecture logicielle adoptée et aux frameworks techniques choisis.[1]

### 1.4.3 Planification du projet

La phase de planification permet de découper le projet en tâches, de décrire leur enchaînement dans le temps, et d’affecter à chacune une durée. Dans le cadre d’une bonne gestion du projet et du respect des délais des différentes étapes de l’application, un planning a été établi,

subdivisé en quatre grandes étapes :

1. Initialisation
2. Analyse et conception
3. Réalisation et teste
4. Rédaction d'un rapport

Le diagramme de Gantt 1.5 présente le planning du projet :

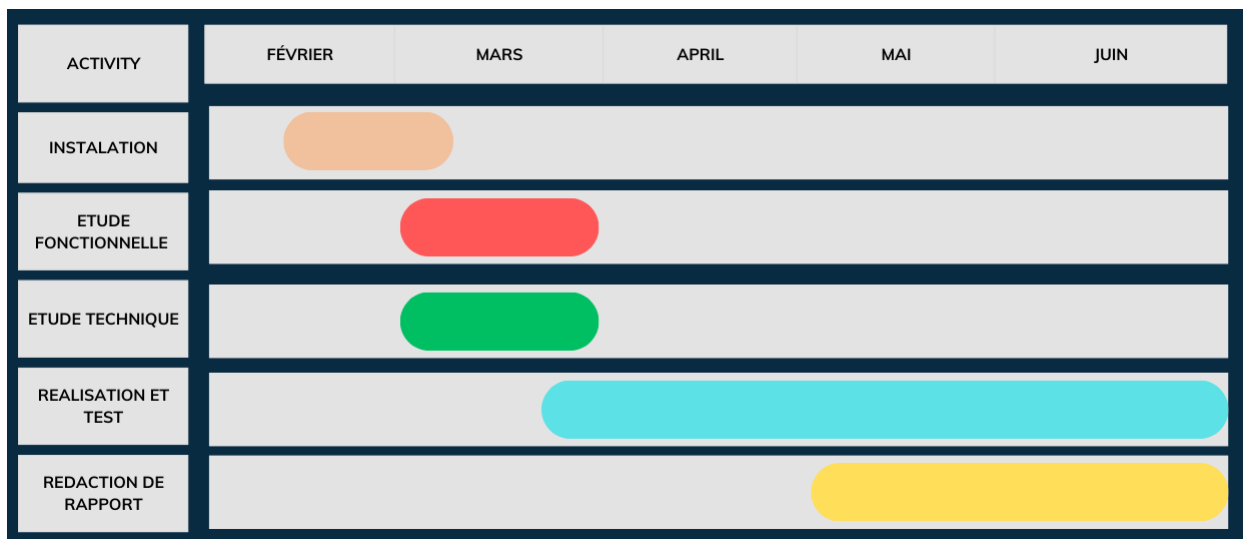


FIGURE 1.5 – Diagramme de Gantt

## 1.5 Conclusion

Dans ce chapitre, l'organisme d'accueil, DataTeam, est d'abord présenté en soulignant sa structure et ses activités. Ensuite, le projet est situé dans son contexte général, mettant en évidence les défis et les opportunités qu'il présente. Enfin, ce chapitre est conclu par la planification détaillée du stage, décrivant les étapes prévues et les objectifs à atteindre. Le prochain chapitre abordera en détail l'analyse du projet, en examinant ses composantes essentielles et en identifiant les besoins et les exigences spécifiques.

## Chapitre 2

# Analyse et Conception

### 2.1 Introduction

Dans ce chapitre d'Analyse et Conception, les aspects fondamentaux de l'architecture et de la structure du système sont abordés. Les mécanismes d'authentification et d'autorisation sont examinés en premier lieu, étant essentiels pour garantir la sécurité et le contrôle d'accès approprié. Ensuite, les cas d'utilisation sont présentés pour définir les interactions clés entre les utilisateurs et le système. Pour approfondir la compréhension de la structure statique de l'application, une analyse du diagramme de classes est menée, illustrant les relations entre les différentes entités du système. Enfin, le diagramme de séquence est exploré pour offrir une vue dynamique des interactions entre les objets au fil du temps, permettant ainsi de visualiser le flux des opérations critiques. Cette approche multidimensionnelle permet de construire une vision complète et cohérente de l'architecture du système, jetant ainsi les bases d'une implémentation robuste et efficace.

### 2.2 Authentication/Authorization

Le système de gestion du temps d'activité nécessite un mécanisme robuste d'authentification et d'autorisation pour garantir la sécurité et l'intégrité des données dans un environnement multi-tenant. Une approche a été conçue pour répondre aux besoins variés de différents types d'utilisateurs tout en maintenant une séparation claire des responsabilités.

### 2.2.1 Structure des Rôles

Quatre rôles principaux ont été définis dans l'application, chacun avec des niveaux d'accès et des responsabilités spécifiques :

1. **Enterprise Admin** : Ce rôle au sommet de la hiérarchie peut créer des tenants et des utilisateurs à tous les niveaux.
2. **Tenant Admin** : Responsable de la création d'organisations au sein d'un locataire et de la gestion des utilisateurs pour ces organisations.
3. **Organization Admin** Dispose d'un contrôle complet sur une organisation spécifique, y compris la création d'utilisateurs et l'accès à toutes les fonctionnalités de l'application.
4. **Organization Operator** : Limité aux fonctionnalités opérationnelles comme la création d'employés, sans pouvoir créer de nouveaux utilisateurs.

### 2.2.2 Processus d'Authentification

Le système utilise une authentification basée sur les cookies pour une expérience utilisateur fluide et sécurisée. Le processus d'authentification se déroule comme suit :

1. L'utilisateur saisit ses identifiants.
2. Le système vérifie ces identifiants dans la base de données de sécurité centralisée.
3. En cas de succès, le rôle de l'utilisateur est déterminé.
4. Un cookie de session sécurisé est créé et envoyé au navigateur de l'utilisateur.
5. Ce cookie est vérifié à chaque requête ultérieure pour maintenir la session et les autorisations.

### 2.2.3 Gestion des Autorisations

Les autorisations sont gérées de manière granulaire en fonction du rôle de l'utilisateur. Chaque action dans l'application est soumise à une vérification d'autorisation, garantissant

que les utilisateurs n'accèdent qu'aux fonctionnalités et aux données correspondant à leur rôle et à leur niveau dans la hiérarchie organisationnelle.

#### **2.2.4 Intégration avec l'Architecture Multi-Tenant**

Le système d'authentification et d'autorisation s'intègre étroitement avec l'architecture multi-tenant. Les informations de session incluent non seulement le rôle de l'utilisateur, mais aussi son appartenance à un tenant et une organisation spécifiques, assurant une séparation efficace des données et des accès entre les différents tenants.

### **2.3 les cas d'utilisation**

Dans le chapitre dédié aux cas d'utilisation, les différentes interactions entre les utilisateurs et le système sont explorées. Chaque cas d'utilisation décrit un ensemble spécifique d'actions que les utilisateurs peuvent entreprendre pour atteindre leurs objectifs à travers le système. Ces scénarios sont essentiels pour comprendre comment les fonctionnalités du système répondent aux besoins des utilisateurs et comment ils interagissent avec les différentes entités du système. Les diagrammes de cas d'utilisation illustrent visuellement ces interactions, décrivant les acteurs impliqués, leurs actions et les résultats attendus. Cette approche permet non seulement de clarifier les fonctionnalités attendues du système, mais aussi de guider le développement et la validation des exigences fonctionnelles.

#### **2.3.1 Gestion du profil**

Le module de gestion des profils offre la possibilité de créer de nouveaux comptes utilisateur, de les consulter, ainsi que de modifier et de gérer les informations personnelles et les préférences associées à chaque compte. Ce module constitue un élément essentiel de l'expérience utilisateur, offrant une interface conviviale et fonctionnelle pour la gestion efficace des profils dans le système.



**Cas d'utilisation « Créer un compte »**

Description	Création des comptes des utilisateurs
Acteurs	Enterprise Admin, Tenant Admin, Organization Admin et Organization Operator
Entrées	Prénom, Nom, UserName, Email, Mot de passe et Confirmation du mot de passe
Sorties	Le compte est créé et un e-mail ou SMS de confirmation est envoyé à l'utilisateur, contenant les informations du compte.
Scenario nominal	<ul style="list-style-type: none"> <li>— L'utilisateur se connecte.</li> <li>— L'utilisateur saisit les informations du nouveau utilisateur.</li> <li>— Le système sauvegarde les informations et notifie le nouveau utilisateur de la création de son compte.</li> </ul>
Scenario d'exception	Si un profil avec les mêmes informations clés existe déjà dans le système, celui-ci affiche la notification : « Cet utilisateur semble déjà exister dans le système. »
Règles de gestion	L'adresse e-mail et le nom d'utilisateur doivent être uniques

TABLE 2.1 – Description du Cas d'utilisation « Créer un compte »

**Cas d'utilisation « S'authentifier »**

Description	L'authentification de l'utilisateur
Acteurs	Enterprise Admin, Tenant Admin, Organization Admin et Organization Operator
Entrées	UserName et Mot de passe
Sorties	Utilisateur authentifié
Scenario nominal	<ul style="list-style-type: none"> <li>— L'utilisateur s'authentifie en saisissant son UserName et son mot de passe.</li> <li>— Le système authentifie l'utilisateur.</li> <li>— Le système affiche le compte de l'utilisateur.</li> </ul>
Scenario d'exception	Les identifiants saisis ne correspondent à aucun utilisateur, le système affiche la notification : « Identifiants incorrects ». Il offre alors deux possibilités à l'utilisateur : Nouvelle tentative d'authentification ou Mot de passe oublié.
Règles de gestion	Si l'utilisateur ferme le navigateur, il sera automatiquement déconnecté.

TABLE 2.2 – Description du Cas d'utilisation « S'authentifier »

**Cas d'utilisation « Modifier un compte utilisateur »**

Description	Modification des données de l'utilisateur
Acteurs	Enterprise Admin, Tenant Admin, Organization Admin et Organization Operator
Entrées	Les informations à modifier
Sorties	Compte utilisateur modifié
Scenario nominal	<ul style="list-style-type: none"><li>— L'utilisateur s'authentifie en saisissant son UserName et son mot de passe.</li><li>— L'utilisateur entre les informations à modifier.</li><li>— Le système sauvegarde les informations modifiées.</li></ul>
Règles de gestion	L'utilisateur ne peut pas modifier UserName.

TABLE 2.3 – Description du Cas d'utilisation « Modifier un compte utilisateur »

**Cas d'utilisation « Afficher le profil de l'utilisateur »**

Description	L'utilisateur accède à son profil
Acteurs	Enterprise Admin, Tenant Admin, Organization Admin et Organization Operator
Entrées	Données d'authentification de l'utilisateur
Sorties	Informations personnelles et les opérations que l'utilisateur peut effectuer
Scenario nominal	<ul style="list-style-type: none"><li>— L'utilisateur s'authentifie en saisissant son UserName et son mot de passe.</li><li>— L'utilisateur consulte les informations personnelles et les opérations qu'il peut effectuer.</li></ul>

TABLE 2.4 – Description du Cas d'utilisation « Afficher le profil de l'utilisateur »

### 2.3.2 Cas d'utilisation « CRUD sur les entités »

Description	L'utilisateur peut faire les opérations de CRUD sur les entités
Acteurs	Enterprise Admin, Tenant Admin, Organization Admin et Organization Operator
Entrées	Une opération de CRUD
Sorties	L'opération de CRUD est réalisée
Scenario nominal	<ul style="list-style-type: none"> <li>— L'utilisateur s'authentifie en saisissant son UserName et son mot de passe.</li> <li>— L'utilisateur réalise les opérations de CRUD possibles.</li> </ul>
Règles de gestion	<p>L'utilisateur peut effectuer des opérations CRUD sur les entités en fonction de son rôle :</p> <ul style="list-style-type: none"> <li>— l'Enterprise Admin peut effectuer des opérations CRUD sur l'entité Tenant</li> <li>— le Tenant Admin peut effectuer des opérations CRUD sur l'entité Organization</li> <li>— l'Organization Operator peut effectuer des opérations CRUD sur les entités Employee, Shift, Planning et Pointage</li> <li>— l'Organization Admin peut effectuer des opérations CRUD sur les entités Employee, Shift, Planning, Pointage et ClockDevice</li> </ul>

TABLE 2.5 – Description du Cas d'utilisation « Les opérations CRUD »

### 2.3.3 Cas d'utilisation « Calculer la synthèse »

Description	L'utilisateur peut calculer la synthèse des pointages
Acteurs	Organization Admin et Organization Operator
Entrées	Cliquer sur « Calculer la synthèse »
Sorties	Opération de « Calculer la synthèse » est réalisée
Scenario nominal	<ul style="list-style-type: none"> <li>— L'utilisateur s'authentifie en saisissant son UserName et son mot de passe.</li> <li>— L'utilisateur réalise l'opération de « Calculer la synthèse »</li> </ul>
Règles de gestion	L'Organization Admin et L'Organization Operator sont les seuls rôles qui peuvent réaliser cette opération

TABLE 2.6 – Description du Cas d'utilisation « Calculer la synthèse »

### 2.3.4 Cas d'utilisation « exécuter des Jobs »

Description	L'utilisateur peut exécuter un Jobs
Acteurs	Organization Admin
Entrées	Cliquer sur le Job
Sorties	Le Job est réalisée
Scenario nominal	<ul style="list-style-type: none"><li>— L'utilisateur s'authentifie en saisissant son UserName et son mot de passe.</li><li>— L'utilisateur exécuter un Jobs</li></ul>

TABLE 2.7 – Description du Cas d'utilisation « exécuter des Jobs »

### 2.3.5 Diagramme de Cas d'utilisation

Un diagramme de cas d'utilisation est un outil essentiel en ingénierie logicielle, faisant partie du langage UML (Unified Modeling Language). Il représente visuellement les interactions entre un système et ses utilisateurs ou d'autres systèmes externes. Ce diagramme se compose d'acteurs (représentés par des figurines) qui symbolisent les utilisateurs ou entités interagissant avec le système, et de cas d'utilisation (représentés par des ovales) qui décrivent les fonctionnalités ou actions spécifiques du système. Le tout est généralement encadré dans un rectangle représentant les limites du système. Les relations entre ces éléments sont indiquées par des lignes, pouvant être de simples associations ou des relations plus complexes comme l'inclusion (« include »), l'extension (« extend ») ou la généralisation. Ce type de diagramme est crucial pour définir la portée du projet, identifier les besoins des utilisateurs, faciliter la communication entre les parties prenantes et servir de base pour la planification et l'estimation du projet. Il aide à visualiser clairement les fonctionnalités du système, à identifier les exigences manquantes ou redondantes, et à fournir une vue d'ensemble compréhensible même pour les non-techniciens. Cependant, il est important de noter que ces diagrammes ne montrent pas l'ordre ou la séquence des actions et peuvent devenir complexes pour les grands systèmes. L'application GTA offre

une suite complète de fonctionnalités pour la gestion du temps et des activités en entreprise, comprenant :

- Authentification et autorisation
- Gestion des utilisateurs, tenants, organisations, emplois, dispositifs de pointage, employés, Jobs, shifts et planings
- Pointage manuel et calcul de synthèse
- Exportation des synthèses vers Excel

Ces fonctionnalités visent à optimiser la gestion des ressources humaines, améliorer l'efficacité opérationnelle et augmenter la productivité des entreprises.

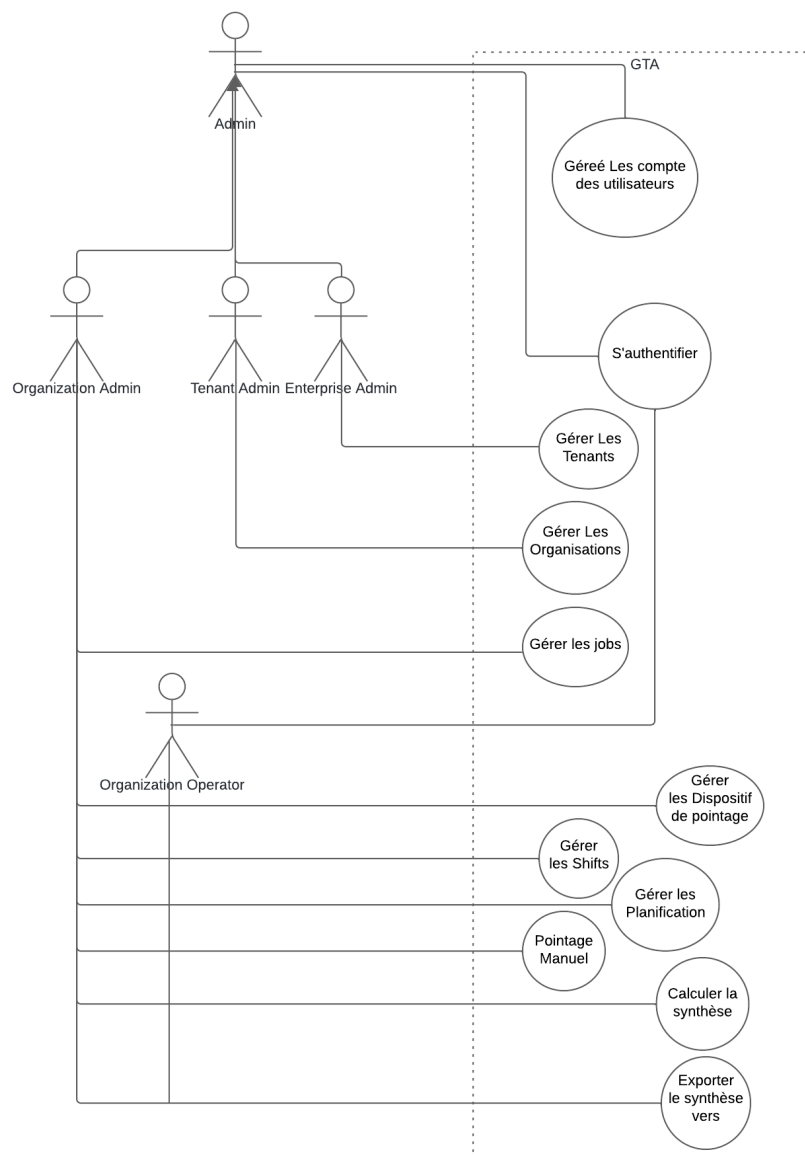


FIGURE 2.1 – Diagramme de Cas d'utilisation

## 2.4 Diagramme de classes

Un diagramme de classes est un élément fondamental de la modélisation orientée objet en UML, utilisé pour représenter la structure statique d'un système logiciel. Il décrit les classes du système, leurs attributs, leurs méthodes et les relations entre elles. Chaque classe est représentée par un rectangle divisé en trois parties : le nom de la classe en haut, les attributs au milieu et les méthodes en bas. Les relations entre les classes sont illustrées par différents types de lignes et symboles, incluant l'association (ligne simple), l'agrégation (losange vide), la composition (losange plein), l'héritage (flèche triangulaire pleine) et la réalisation (flèche triangulaire en pointillés). Ce diagramme est crucial pour la conception de logiciels, car il fournit une vue d'ensemble de l'architecture du système, facilite la communication entre développeurs et sert de base pour la génération de code et la documentation. Il permet de visualiser les dépendances entre les classes, d'identifier les opportunités de réutilisation du code et de comprendre la structure globale du système avant même de commencer le codage.

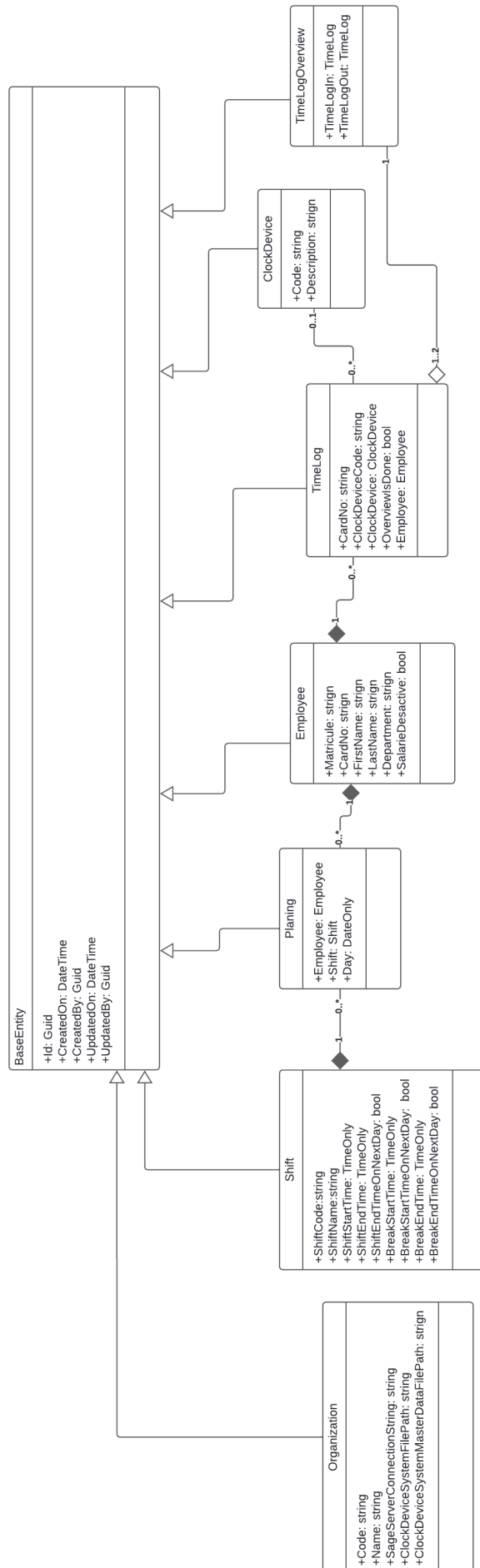


FIGURE 2.2 – Diagramme de classes

## 2.5 Diagramme de séquence

Un diagramme de séquence est un type de diagramme d'interaction en UML qui illustre la séquence chronologique des messages échangés entre les objets dans un scénario spécifique. Il se compose d'une dimension verticale représentant le temps qui s'écoule de haut en bas, et d'une dimension horizontale montrant les différents objets ou acteurs impliqués dans l'interaction. Chaque objet est représenté par une ligne de vie verticale, avec un rectangle en haut indiquant son nom. Les messages entre objets sont symbolisés par des flèches horizontales, le type de flèche indiquant la nature du message (synchrone, asynchrone, retour). Les activations, représentées par des rectangles fins sur les lignes de vie, montrent quand un objet est actif ou en train de traiter une opération. Ce diagramme peut également inclure des structures de contrôle comme des boucles et des conditions. Il est particulièrement utile pour modéliser les flux de travail, comprendre les interactions complexes entre composants, identifier les goulots d'étranglement potentiels dans les performances et valider la logique des processus métier. Les diagrammes de séquence sont essentiels pour les développeurs et les architectes logiciels car ils fournissent une vue dynamique du système, complémentaire à la vue statique offerte par les diagrammes de classes.



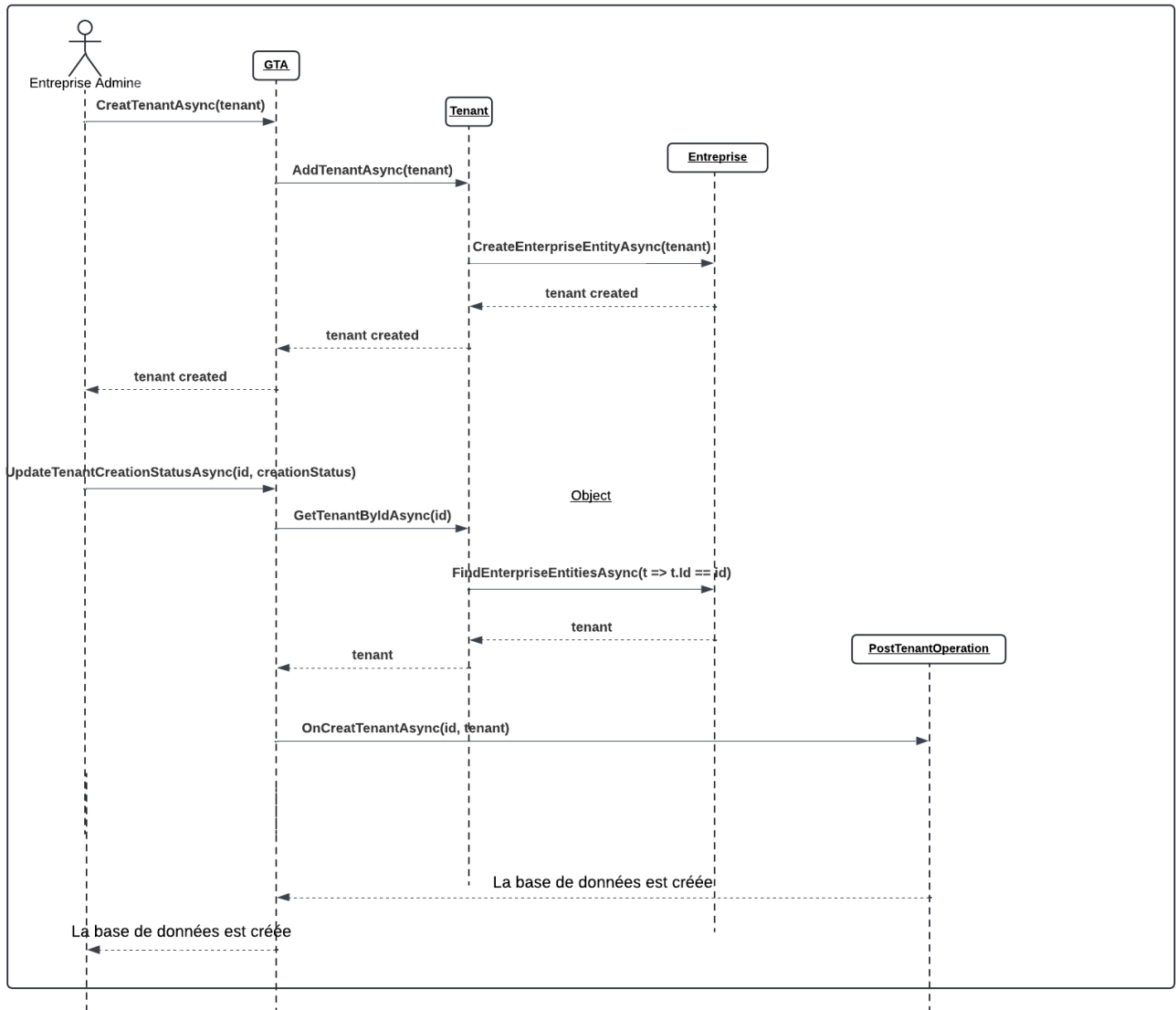


FIGURE 2.3 – Diagramme de séquence de la création et de l’initialisation d’un Tenant

## 2.6 Conclusion

En conclusion, ce chapitre d’Analyse et Conception a établi les fondements essentiels du système. Les aspects cruciaux de la sécurité ont été abordés à travers les mécanismes d’authentification et d’autorisation, garantissant ainsi l’intégrité et la confidentialité des données. Les cas d’utilisation ont clarifié les interactions utilisateur-système, tandis que le diagramme de classes a offert une vue structurelle approfondie de l’application. Le diagramme de séquence, quant à lui, a apporté une perspective dynamique en illustrant le flux des opérations et les interactions entre les composants. Cette analyse multifacette a permis de construire une architecture solide et cohérente, répondant aux exigences fonctionnelles et non fonctionnelles du projet.

Alors qu’une compréhension claire de la structure et du comportement du système a été établie, le prochain chapitre, « Étude technique », s’appuiera sur ces bases pour explorer en détail les choix technologiques, les outils de développement et les considérations d’implémentation spécifiques. Cette transition permettra de passer de la conception théorique à la réalisation concrète du projet, en garantissant que les décisions techniques s’alignent parfaitement avec l’architecture et les objectifs définis dans ce chapitre.

## Chapitre 3

# Etude technique

### 3.1 Introduction

Dans ce chapitre consacré à l'Étude technique, la transition cruciale de la phase conceptuelle vers les aspects pratiques et technologiques du projet est marquée. Dans cette section, les détails techniques sous-tendant la réalisation concrète du système seront examinés en détail, en s'appuyant sur les fondements établis lors de l'analyse et de la conception.

L'identification et l'analyse des contraintes techniques encadrant le projet constitueront le point de départ. Ensuite, l'architecture physique sera explorée en profondeur, en mettant l'accent sur le concept de Multi-Tenancy, une approche essentielle pour optimiser l'efficacité et la flexibilité de l'infrastructure.

L'architecture logicielle sera au cœur de cette étude, avec un focus particulier sur la Clean Architecture, le Domain-Driven Design, et l'utilisation du Generic Repository. Ces paradigmes et patterns permettront de construire une structure logicielle robuste, maintenable et évolutive. À travers ce chapitre, l'objectif est de fournir une compréhension approfondie des choix technologiques et architecturaux qui guideront le développement du système. Cette étude technique jettera les bases solides nécessaires pour une implémentation efficace et cohérente, en ligne avec les objectifs et les exigences du projet définis précédemment.

### 3.2 Contraintes techniques

Cette section présente les contraintes techniques de la solution. En effet, l'identification de ces contraintes est considérée comme étant la phase la plus importante pour le bon fonctionnement des systèmes finaux. Je peux citer les contraintes suivantes :

- Les données fournies par l'application doivent être fiables.

- Le système doit prendre en compte la confidentialité de l'information, pour cela des mécanismes de sécurités ont été mis en place.
- Le système doit fournir une interface conviviale et simple pour le client car elle représente le premier contact entre l'utilisateur et le système.
- Le système peut être amélioré par l'ajout de d'autres modules pour garantir la souplesse, l'évolutivité et l'ouverture de la solution.
- La solution doit avoir un temps de réponse optimal en ce qui concerne le chargement de l'application, les délais de rafraîchissement et importation / exportation des données.
- Le système doit être capable de capturer et gérer les erreurs et les mauvaises données.

### 3.3 Architecture physique : Multi-Tenant

#### 3.3.1 Introduction au Multi-Tenancy

Le Multi-Tenancy, est un principe architectural clé dans le développement de logiciels modernes, cette approche permet à une seule instance d'une application de servir simultanément plusieurs clients. Chaque locataire bénéficie d'un environnement virtuellement séparé, tout en partageant les ressources sous-jacentes de l'application. Cette méthode optimise l'utilisation des ressources, réduit les coûts opérationnels et simplifie la maintenance, tout en garantissant l'isolation des données et la personnalisation pour chaque client.[2]

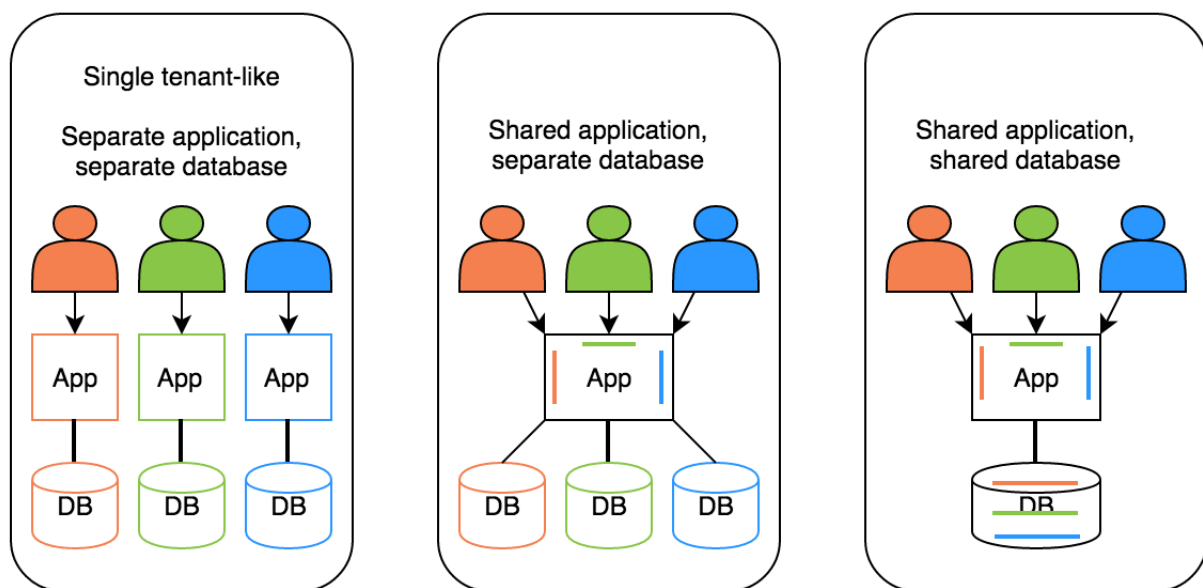


FIGURE 3.1 – Architecture multitenant

### 3.3.2 Avantages du Multi-Tenancy

Le Multi-Tenancy offre de nombreux avantages tant pour les fournisseurs de services que pour les clients :

**1. Optimisation des ressources :**

- Partage efficace des ressources matérielles et logicielles entre les locataires
- Réduction des coûts d'infrastructure et de maintenance

**2. Mise à jour et maintenance simplifiées :**

- Déploiement centralisé des mises à jour pour tous les locataires
- Réduction du temps et des efforts nécessaires pour la maintenance

**3. Sécurité et isolation des données :**

- Séparation logique des données entre les locataires
- Implémentation des contrôles d'accès et des politiques de sécurité spécifiques

**4. Analyse et insights :**

- Collecte centralisée des données d'utilisation de tous les locataires
- Possibilité d'obtenir des insights précieux pour l'amélioration du produit

**5. Time-to-market réduit :**

- Déploiement plus rapide de nouvelles fonctionnalités pour tous les clients
- Capacité à réagir rapidement aux demandes du marché

### 3.3.3 Multi-Tenant Architecture Design

La conception de l'architecture Multi-Tenant est un aspect crucial du projet, où il est nécessaire d'accommoder plusieurs locataires (tenants), chacun pouvant potentiellement avoir de nombreuses organisations. Deux approches principales ont été envisagées pour cette architecture :

#### Approche de Base de Données Unique

La première approche de l'architecture Multi-Tenant repose sur l'utilisation d'une base de données unique pour tous les locataires et leurs organisations. Dans ce modèle, deux identifiants clés sont utilisés : 'tenantid' et 'organizationid', pour distinguer les données appartenant

à chaque client et à leurs organisations respectives. Ces identifiants sont intégrés dans chaque table de la base de données, permettant une séparation logique des données sans nécessiter de séparation physique. Cette méthode vise à optimiser l'utilisation des ressources et à simplifier la gestion de l'infrastructure, tout en maintenant une isolation logique des données entre les différents locataires.

L'implémentation de cette approche exige une attention particulière à la conception des requêtes et des index. Chaque requête doit systématiquement inclure les clauses de filtrage appropriées basées sur ces identifiants pour garantir que les utilisateurs n'accèdent qu'aux données qui leur sont propres. De plus, des index composites incluant 'tenantid' et 'organizationid' sont généralement nécessaires pour optimiser les performances des requêtes.

Cette approche présente plusieurs avantages notables. Elle offre une grande simplicité de gestion et de maintenance, car il n'y a qu'une seule base de données à administrer, mettre à jour et sauvegarder. Cela se traduit par une réduction significative de la complexité opérationnelle et des coûts associés.

L'utilisation des ressources est optimisée. Comme toutes les données sont stockées dans une seule base, cela permet une meilleure utilisation de l'espace disque et de la mémoire.

Cependant, cette approche comporte aussi des inconvénients significatifs qu'il faut prendre en compte. Le plus important est le risque accru de fuite de données entre les locataires. Une simple erreur de programmation dans une requête, comme l'omission d'un filtre sur 'tenantid', pourrait potentiellement exposer les données d'un locataire à un autre, ce qui constitue une grave violation de la confidentialité.

De plus, à mesure que le nombre de locataires et le volume de données augmentent, les performances peuvent se dégrader.

La personnalisation devient également plus complexe avec cette approche. Il est difficile d'offrir des fonctionnalités ou des schémas de données spécifiques à certains locataires sans affecter les autres, ce qui peut limiter la flexibilité du service.

Enfin, cette approche peut poser des défis en termes de conformité réglementaire, particulièrement dans les cas où certains locataires exigent que leurs données soient physiquement isolées ou stockées dans des régions géographiques spécifiques.

## Architecture à bases de données multiples

La seconde approche, choisie pour ce projet, est une architecture à bases de données multiples. Cette méthode utilise trois types distincts de bases de données pour gérer efficacement la structure multi-tenant tout en maintenant une séparation claire des données.

Le premier type est une base de données dédiée à la sécurité. Elle gère l'authentification et les autorisations pour l'ensemble du système. Cette base centralisée stocke les informations relatives aux utilisateurs, leurs rôles, et leurs permissions, assurant ainsi une gestion cohérente de la sécurité à travers toute l'application.

Le deuxième type est une base de données des tenants. Elle contient les informations essentielles sur tous les locataires, telles que leurs identifiants uniques, leurs noms, leurs paramètres de configuration généraux. Cette base sert de point central pour la gestion des tenants au niveau système.

Le troisième type, et le plus crucial, consiste en des bases de données individuelles pour chaque tenant. Chaque locataire dispose de sa propre base de données physiquement séparée, contenant toutes ses données spécifiques. Au sein de ces bases de données individuelles, on utilise un 'organizationId' pour distinguer les différentes organisations appartenant à ce tenant.

Cette approche présente plusieurs avantages significatifs. Premièrement, elle fournit une isolation forte des données entre les tenants. Comme chaque tenant a sa propre base de données physique, le risque de fuite de données entre locataires est considérablement réduit. Cela répond aux préoccupations de sécurité et de confidentialité, particulièrement importantes pour les clients manipulant des données sensibles.

Deuxièmement, cette architecture offre une grande flexibilité pour personnaliser le schéma de la base de données en fonction des besoins spécifiques de chaque tenant. Si un client nécessite des tables ou des champs supplémentaires, ces modifications peuvent être apportées à sa base de données sans affecter les autres.

Troisièmement, les performances sont optimisées pour chaque tenant. L'activité d'un locataire n'a pas d'impact sur les performances des autres, car ils ne partagent pas les mêmes ressources de base de données. Cela permet une meilleure gestion des charges de travail.

Enfin, cette approche facilite la conformité aux réglementations sur la localisation des données. Si certains clients exigent que leurs données soient stockées dans des régions géographiques

spécifiques, cela peut être facilement réalisé en hébergeant leur base de données dans le centre de données approprié.

Cependant, cette approche présente aussi des défis. La gestion de l'infrastructure devient plus complexe avec de multiples bases de données à maintenir. Les coûts en ressources peuvent être plus élevés, en particulier pour les petits tenants qui pourraient ne pas utiliser pleinement une base de données dédiée.

Malgré ces défis, cette approche a été choisie pour le projet en raison de ses avantages en termes de sécurité, de flexibilité et de performance. Elle permet de répondre aux exigences variées des clients tout en maintenant un haut niveau de séparation et de personnalisation des données.

## **3.4 Architecture logicielle**

### **3.4.1 Clean Architecture**

Dans le cadre du projet de gestion du temps d'activité, les principes de la Clean Architecture ont été choisis pour structurer l'application. Cette approche, popularisée par Robert C. Martin (Uncle Bob), vise à créer des systèmes logiciels indépendants des frameworks, testables, et maintenables à long terme.[3]

#### **Principes fondamentaux**

La Clean Architecture repose sur plusieurs principes clés. Elle garantit l'indépendance totale des frameworks, en les utilisant comme des outils plutôt que de contraindre le système à leurs limitations spécifiques. Cette approche favorise également la testabilité, en permettant la validation de la logique métier sans dépendre d'éléments externes tels que l'interface utilisateur, la base de données ou des services web. De plus, elle assure une flexibilité optimale, en permettant des modifications aisées de l'UI sans impact sur le reste du système, et en facilitant le changement de système de stockage sans affecter la logique métier. Enfin, la Clean Architecture maintient une indépendance complète vis-à-vis de tout agent externe, assurant ainsi une conception robuste et facilement évolutive.



## Structure en couches

L'application sera structurée en couches concentriques, chacune ayant une responsabilité spécifique.

1. **Entités (Entities)** : Représentent les objets métier de l'application.
2. **Cas d'utilisation (Use Cases)** : Contiennent les règles métier spécifiques à l'application.
3. **Interfaces d'adaptateurs (Interface Adapters)** : Convertissent les données entre les formats les plus pratiques pour les cas d'utilisation et les entités.
4. **Frameworks et Drivers** : Composée de frameworks et d'outils comme la base de données ou le framework web.

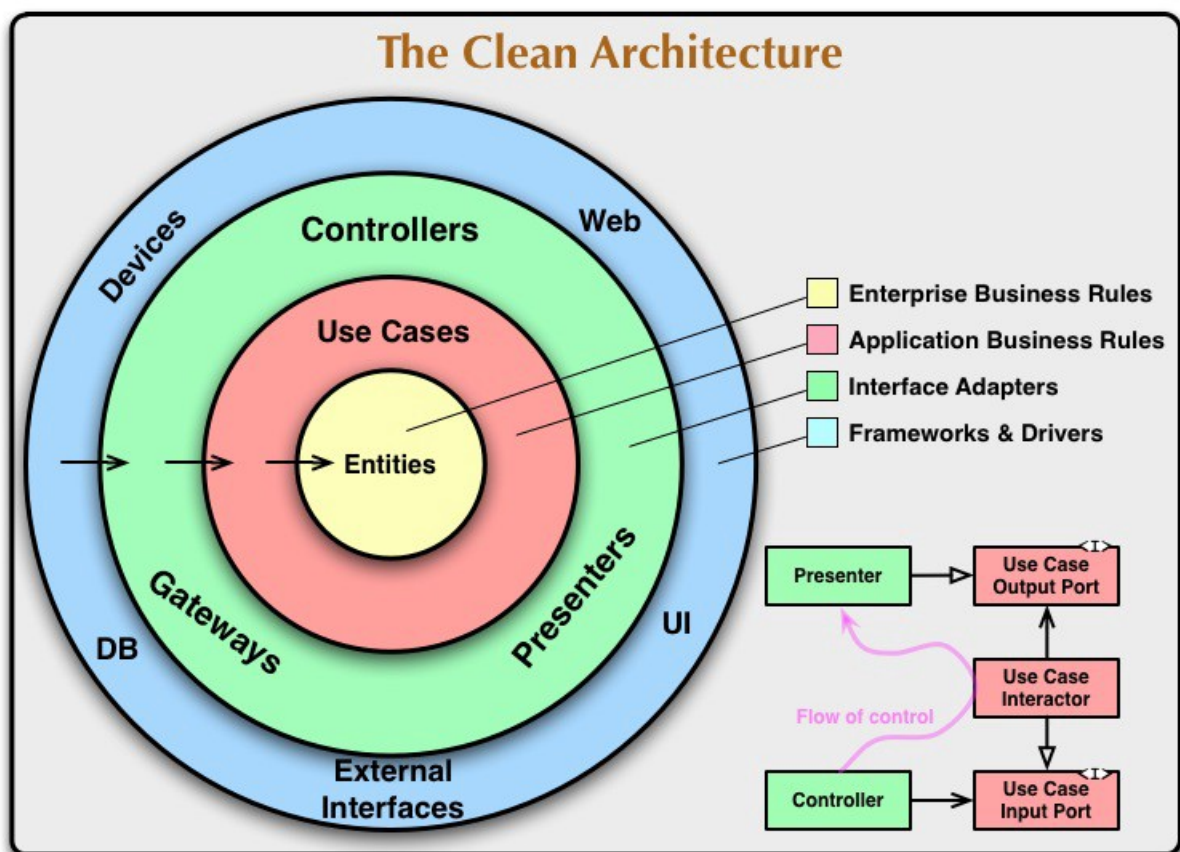


FIGURE 3.2 – Clean Architecture

## Règle de dépendance

La règle fondamentale qui régit l'architecture est la règle de dépendance : les dépendances de code source ne peuvent pointer que vers l'intérieur. Rien dans un cercle interne ne peut connaître quoi que ce soit d'un cercle externe.

## Avantages pour le projet

L'adoption de la Clean Architecture pour le système de gestion du temps d'activité offre plusieurs avantages :

1. **Flexibilité** : Permet d'adapter facilement le système à différents clients avec des besoins spécifiques.
2. **Maintenabilité** : La séparation claire des préoccupations facilite la maintenance et l'évolution du système.
3. **Testabilité** : La logique métier peut être testée indépendamment de l'infrastructure.

### 3.4.2 Domain-Driven Design (DDD)

Dans le cadre du projet de gestion du temps d'activité, les principes du Domain-Driven Design (DDD) ont été intégrés pour structurer l'application. Cette approche, introduite par Eric Evans, vise à aligner étroitement le développement logiciel avec la complexité du domaine métier qu'il supporte. Le DDD met l'accent sur la collaboration entre experts métier et développeurs pour capturer et formaliser les concepts clés du domaine, en les traduisant directement dans la conception logicielle.[4]

## Principes fondamentaux

Le Domain-Driven Design repose sur plusieurs principes essentiels :

1. **Ubiquitous Language** : Un langage commun partagé entre les experts du domaine et les développeurs.
2. **Bounded Contexts** : Des frontières explicites au sein desquelles un modèle particulier s'applique.

3. **Context Mapping** : La définition des relations entre différents Bounded Contexts.
4. **Entités et Objets de Valeur** : Distinction entre les objets définis par leur identité et ceux définis par leurs attributs.
5. **Agrégats** : Groupes d'entités et d'objets de valeur traités comme une unité cohérente.
6. **Repositories** : Abstraction de la persistance des données pour les agrégats.
7. **Domain Events** : Capture des occurrences significatives dans le domaine.

#### Avantages pour le projet

L'intégration du Domain-Driven Design dans le système de gestion du temps d'activité offre plusieurs avantages stratégiques :

1. **Alignement Métier-Technologie** : Assure une correspondance directe entre les exigences métier et la mise en œuvre technique, améliorant ainsi la qualité et la pertinence de solution logicielle.
2. **Flexibilité et Adaptabilité** : Permet d'évoluer et de s'adapter rapidement aux changements métier tout en maintenant une architecture cohérente et robuste.
3. **Meilleure Compréhension et Communication** : Favorise une communication efficace entre les équipes techniques et métier grâce à un langage commun et une modélisation précise.
4. **Robustesse** : La modélisation précise du domaine conduit à un système plus fiable et plus facile à maintenir.

#### 3.4.3 Generic Repository

Pour structurer l'application de gestion du temps d'activité, l'adoption du pattern Generic Repository a été choisie. Ce pattern, largement utilisé en développement logiciel, vise à abstraire et centraliser l'accès aux données, indépendamment du type de données ou de la source de données sous-jacente. En utilisant le Generic Repository, les opérations CRUD (Create, Read, Update, Delete) sont encapsulées dans des méthodes génériques, réduisant ainsi la duplication de code et facilitant la gestion des entités de données à travers l'application.[5]

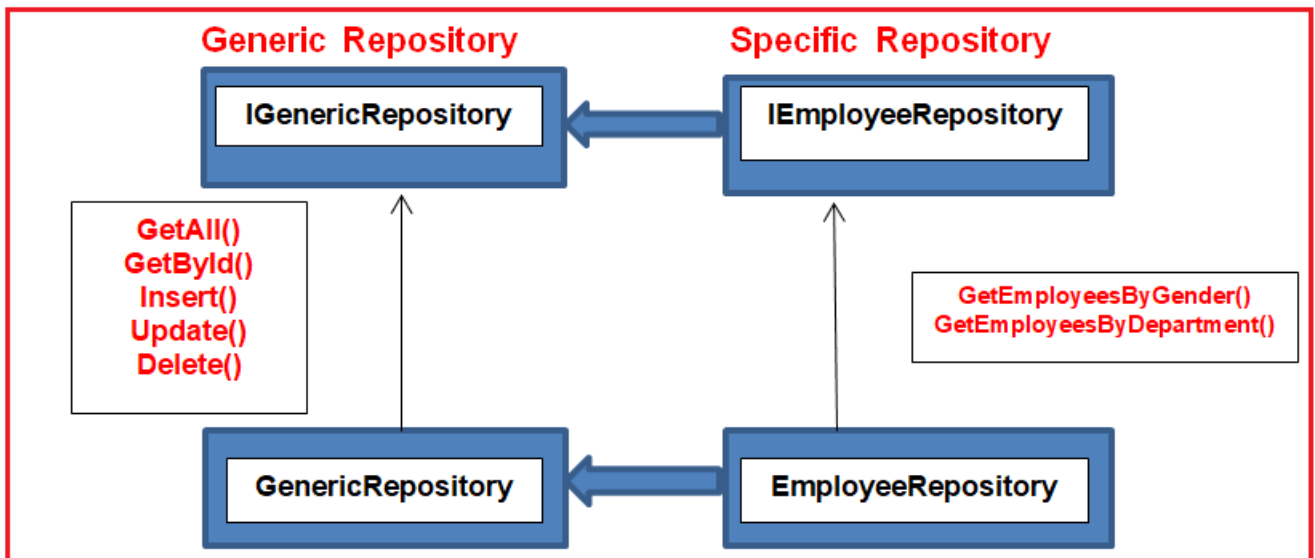


FIGURE 3.3 – Generic Repository

### Principes fondamentaux

Le Generic Repository repose sur le principe d'abstraction des opérations de persistance des données. Il permet d'interagir avec différentes entités de manière uniforme, sans dépendre directement de la logique spécifique de chaque entité ou de la technologie de stockage sous-jacente. Cela favorise la réutilisation du code, simplifie la maintenance et améliore la cohérence dans la gestion des données au sein de l'application.

### Avantages pour le projet

L'adoption du Generic Repository présente plusieurs avantages significatifs :

1. **Réduction de la Redondance** : Centralise les opérations CRUD dans des méthodes génériques, éliminant ainsi la duplication de code.
2. **Flexibilité et Extensibilité** : Facilite l'ajout de nouvelles entités sans modifier l'infrastructure de persistance existante, grâce à une approche générique et adaptable.

## 3.5 Conclusion

Ce chapitre d'Étude technique a permis d'explorer en profondeur les aspects technologiques et architecturaux essentiels à la réalisation du projet. L'identification des contraintes techniques

a établi un cadre clair pour les choix technologiques. L'exploration de l'architecture physique, notamment le concept de Multi-Tenancy, a mis en lumière une approche favorisant une infrastructure efficace et évolutive.

L'accent porté sur l'architecture logicielle, incluant la Clean Architecture, le Domain-Driven Design (DDD), et l'utilisation du Generic Repository, a fourni une base solide pour la conception d'un système robuste et maintenable. Ces choix architecturaux permettent de développer une solution flexible, capable de s'adapter aux besoins changeants et de faciliter les évolutions futures.

En abordant ces aspects techniques, il a été établi un lien crucial entre la phase de conception et la phase de mise en œuvre. Les décisions prises dans ce chapitre guideront directement l'approche de développement, assurant une cohérence entre la vision du projet et sa réalisation concrète.

En prévision du prochain chapitre, "Mise en Œuvre", une préparation adéquate est mise en place pour traduire ces concepts techniques en une implémentation pratique, centrée sur les aspects concrets du développement.

## Chapitre 4

# Mise en Œuvre

### 4.1 Introduction

Ce chapitre consacré à la Mise en Œuvre représente une étape cruciale dans le projet, passant de la conception théorique à la réalisation concrète de l'application. Il détaille le processus de développement, mettant en lumière les outils, technologies et méthodologies utilisés pour transformer les concepts en un produit fonctionnel.

Le chapitre commence par explorer l'environnement et les outils de développement sélectionnés pour le projet. Les différents frameworks utilisés sont également examinés, chacun apportant des fonctionnalités spécifiques pour enrichir le développement.

Une attention particulière est accordée au système de gestion de base de données et à la stratégie de contrôle de version. Ces outils sont essentiels pour garantir la robustesse, la scalabilité et la collaboration efficace au sein de l'équipe de développement.

Enfin, les interfaces homme-machine (IHM) de l'application sont présentées, illustrant comment les concepts et les fonctionnalités définis dans les chapitres précédents prennent vie dans une interface utilisateur concrète et intuitive.

À travers ce chapitre, l'objectif est de fournir une compréhension claire et détaillée du processus de développement, démontrant comment les choix technologiques et architecturaux se traduisent en une application fonctionnelle et performante. Cette mise en œuvre concrétise l'aboutissement du travail de conception et d'analyse, transformant la vision en un produit tangible.

## 4.2 Environnement et outils de développement

### 4.2.1 Langage C#

C# est un langage de programmation moderne et robuste, développé par Microsoft pour la plateforme .NET. Depuis son introduction en 2002, il s'est rapidement imposé comme l'un des langages de programmation les plus populaires dans l'industrie du développement logiciel. Grâce à sa syntaxe claire et à sa structure orientée objet, le C# offre aux développeurs un environnement de développement puissant et efficace pour créer une grande variété d'applications, incluant les applications de bureau, web, mobiles et même les jeux vidéo.

Un des principaux avantages du C# réside dans son typage statique fort, assurant la vérification des types de variables lors de la compilation. Cela permet de détecter les erreurs dès les premières étapes du développement, renforçant ainsi la robustesse et la fiabilité des applications écrites en C#..

Par ailleurs, le C# bénéficie d'une intégration étroite avec l'écosystème de développement de Microsoft, notamment Visual Studio, l'environnement de développement intégré (IDE) de premier plan utilisé par de nombreux développeurs. Cette intégration facilite le processus de développement en offrant des outils avancés pour la conception, le débogage et le déploiement d'applications.

En outre, le C# offre un support natif pour les fonctionnalités avancées telles que la programmation asynchrone, la réflexion, les expressions lambda et les types anonymes, permettant aux développeurs de créer des applications plus performantes et évolutives.[6]

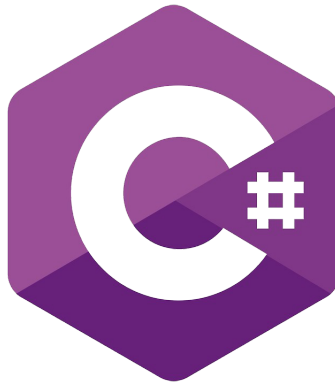


FIGURE 4.1 – Logo du Langage de programmation C#

#### 4.2.2 Visual Studio 2022

Visual Studio 2022 est un outil incontournable pour les développeurs travaillant avec le langage C#. Cette dernière version de l'IDE de Microsoft offre une gamme complète de fonctionnalités spécialement conçues pour améliorer l'expérience de développement en C#. Grâce à son intégration étroite avec le framework .NET, Visual Studio 2022 permet aux développeurs de bénéficier des dernières avancées en matière de langage et d'outils de développement.

L'un des points forts de Visual Studio 2022 est son éditeur de code puissant et intuitif, qui offre des fonctionnalités avancées telles que la coloration syntaxique, l'achèvement automatique, la refactoring et la navigation facilitée dans le code. Ces fonctionnalités permettent aux développeurs de travailler de manière plus efficace et de réduire les erreurs tout au long du processus de développement.

De plus, Visual Studio 2022 propose une suite complète d'outils de débogage et de profilage facilitant l'identification et la résolution des problèmes dans les applications C#. Les développeurs peuvent utiliser les points d'arrêt pour diagnostiquer et corriger les erreurs plus rapidement.

En outre, Visual Studio 2022 prend en charge une large gamme de projets C#, y compris les applications de bureau, web, mobiles et cloud. Les développeurs peuvent créer, déployer et gérer leurs applications C# dans un environnement unifié, ce qui facilite la collaboration et la gestion de projet.[7]





FIGURE 4.2 – Logo de Visual Studio 2022

### 4.2.3 La plateforme .NET

La plateforme .NET, est bien plus qu'un simple ensemble d'outils et de modules. C'est un écosystème complet conçu pour faciliter le développement d'une grande variété d'applications, qu'elles soient destinées aux environnements de bureau ou aux plateformes Web. Avec son vaste ensemble de bibliothèques et de classes, .NET offre aux développeurs un cadre solide pour créer des applications robustes et évolutives.

Au cœur de la plateforme .NET se trouve C#, un langage de programmation moderne et polyvalent. C# est devenu l'un des langages les plus populaires parmi les développeurs en raison de sa syntaxe claire, de sa puissance et de sa flexibilité. En choisissant C# comme langage de développement, les développeurs bénéficient d'une large communauté de soutien, de nombreuses ressources disponibles et d'une évolution continue du langage.

En ce qui concerne le développement web, ASP.NET est un élément clé de la plateforme .NET. Il s'agit d'un framework web puissant qui permet aux développeurs de créer des applications web dynamiques et interactives. Grâce à son intégration étroite avec le serveur web Internet Information Services (IIS), ASP.NET offre des performances exceptionnelles et une sécurité renforcée pour les applications Web.[8]



FIGURE 4.3 – Logo de la plateforme .NET

#### 4.2.4 Frameworks utilisés

Un Framework est un ensemble d'outils sur lequel repose la conception et le développement d'une application logicielle. En effet, il fournit un ensemble d'outils, de bibliothèques et de modèles de conception qui permettent aux développeurs de créer des applications de manière efficace et cohérente. En d'autres termes, le Framework établit les bases et la structure fondamentale d'une application, en fournissant un cadre dans lequel les développeurs peuvent construire et organiser leur code de manière logique et structurée. Les principaux Frameworks utilisés dans le développement logiciel offrent des fonctionnalités spécifiques adaptées à différents domaines et types d'applications, permettant ainsi aux développeurs de choisir le Framework le mieux adapté à leurs besoins et exigences spécifiques.

##### **Blazor**

Blazor est une technologie révolutionnaire dans le domaine du développement web, offrant une approche novatrice pour la création d'applications web interactives et dynamiques. Basé sur le framework .NET, Blazor permet aux développeurs d'utiliser le langage C# pour construire des applications web à la fois côté client et côté serveur, sans avoir à se soucier du JavaScript. L'un des principaux avantages de Blazor est sa capacité à utiliser le code C# pour manipuler le DOM (Document Object Model) et répondre aux événements utilisateur, ce qui simplifie considérablement le processus de développement et réduit la complexité du code. De plus, Blazor

offre une expérience de développement familière pour les développeurs .NET, en leur permettant de tirer parti de leur expertise existante dans le langage C# et les bibliothèques .NET. Grâce à son modèle de programmation basé sur les composants, Blazor facilite la création d'interfaces utilisateur modulaires et réutilisables, ce qui permet aux développeurs de construire des applications web évolutives et maintenables. De plus, Blazor offre une intégration transparente avec d'autres technologies .NET, telles que ASP.NET Core, Entity Framework Core et SignalR, permettant aux développeurs de construire des applications web complètes et riches en fonctionnalités.[9]



FIGURE 4.4 – Logo de Blazor

#### **Blazor côté client**

Avec Blazor côté client, le code C# est compilé en WebAssembly, permettant ainsi aux applications de s'exécuter de manière efficace et sécurisée dans le sandbox du navigateur.

#### **Blazor avec rendu côté serveur**

Blazor avec rendu côté serveur offre une alternative flexible et performante pour le développement d'applications web. Avec cette approche, une grande partie du code est exécutée côté serveur, tandis que l'interface utilisateur est rendue et mise à jour dynamiquement dans le

navigateur à l'aide de SignalR. Blazor avec rendu côté serveur réduit la charge de travail sur le client en déplaçant une partie du traitement vers le serveur, ce qui peut être particulièrement avantageux pour les applications nécessitant des calculs intensifs ou un accès aux ressources serveur. Le projet utilise Blazor avec rendu côté serveur.

## **.NET Core Web API**

.NET Core Web API est un framework puissant et flexible développé par Microsoft pour la création d'interfaces de programmation d'applications (API) web. En utilisant .NET Core Web API, les développeurs peuvent créer des services web hautement performants et évolutifs, capables de répondre aux besoins les plus exigeants en matière de développement d'applications modernes. .NET Core Web API prend en charge une variété de formats de données, y compris JSON et XML, ce qui permet aux développeurs de créer des API compatibles avec un large éventail de clients. Avec ses fonctionnalités avancées telles que la gestion des routes, la validation des données, la sécurité intégrée et le support des websockets, .NET Core Web API offre une solution complète pour la création d'API web robustes et sécurisées. En résumé, .NET Core Web API est un outil essentiel pour les développeurs souhaitant créer des services web modernes et performants dans l'écosystème .NET.

## **Entity Framework Core**

Entity Framework Core est un framework de mapping objet-relationnel (ORM) développé par Microsoft, conçu pour simplifier l'accès et la manipulation des données dans les applications .NET. En utilisant Entity Framework Core, les développeurs peuvent travailler avec des données relationnelles en utilisant des objets .NET, ce qui simplifie le développement en éliminant la nécessité de gérer manuellement les requêtes SQL et la correspondance entre les objets et les tables de la base de données.

Entity Framework Core prend en charge une variété de bases de données relationnelles populaires, telles que SQL Server, MySQL, PostgreSQL et SQLite, offrant ainsi aux développeurs

une flexibilité maximale dans le choix de la plateforme de données. De plus, Entity Framework Core offre des fonctionnalités avancées telles que le suivi des modifications, la gestion des transactions, la création de migrations et la gestion des relations entre les entités, permettant aux développeurs de créer des applications robustes et évolutives.

L'un des avantages majeurs d'Entity Framework Core est sa prise en charge du développement basé sur le code (Code-First). Ceci permet aux développeurs de définir le modèle de données à l'aide de classes .NET et d'annotations, puis de générer automatiquement la base de données à partir de ce modèle. Cette approche favorise un développement agile en permettant aux développeurs de se concentrer sur la logique métier de leur application sans se soucier des détails de la base de données sous-jacente.



FIGURE 4.5 – Logo de Entity Framework Core

## Serilog

Serilog est une bibliothèque de logging pour les applications .NET. Elle se distingue par sa flexibilité et sa facilité d'utilisation, offrant une approche structurée de la logging qui facilite l'analyse et le traitement des logs. Serilog utilise un concept de "sinks" qui permet de diriger les logs vers différentes destinations comme des fichiers, des bases de données, ou des services cloud, de manière simultanée et configurable. Sa syntaxe fluide et ses templates de messages riches permettent aux développeurs de créer des logs expressifs et informatifs avec un minimum d'effort. Serilog supporte également la logging contextuelle, permettant d'enrichir au-

tomatiquement les logs avec des informations supplémentaires sur l'environnement d'exécution.



FIGURE 4.6 – Logo de Serilog

## Postman

Postman est un outil essentiel pour les développeurs et les testeurs d'API, offrant une plateforme robuste pour la conception, le test, la documentation et le partage d'interfaces de programmation d'applications (API). Cet outil polyvalent simplifie grandement le processus de développement d'API en permettant aux utilisateurs de créer et d'envoyer des requêtes HTTP complexes, de gérer des environnements multiples, d'automatiser des tests et de générer une documentation complète. Avec son interface utilisateur intuitive, Postman facilite la collaboration entre les équipes, permettant aux développeurs de partager des collections de requêtes et des environnements de test. Il prend en charge une variété de protocoles d'authentification et offre des fonctionnalités avancées telles que les scripts de pré-requête et de tests, les variables d'environnement, et la possibilité d'exécuter des flux de travail complexes.



FIGURE 4.7 – Logo de Postman

#### 4.2.5 Microsoft SQL Server

Microsoft SQL Server est un système de gestion de base de données (SGBD) développé et commercialisé par Microsoft, offrant une gamme complète de fonctionnalités pour le stockage, la manipulation et la gestion des données. Conçu comme un SGBD relationnel, SQL Server est largement utilisé dans les environnements d'entreprise pour prendre en charge une variété d'applications et de charges de travail.

SQL Server fonctionne sur une variété de systèmes d'exploitation, notamment Windows, Linux et Mac OS, offrant ainsi une flexibilité maximale aux utilisateurs.

La suite SQL Server comprend plusieurs services principaux, chacun offrant des fonctionnalités spécifiques pour répondre aux besoins des entreprises :

- Le moteur relationnel (OLTP) appelé SQL Server, qui gère les opérations de transaction en ligne, telles que l'insertion, la mise à jour et la suppression de données.
- Le moteur décisionnel (OLAP) appelé SQL Server Analysis Services (SSAS), qui prend en charge l'analyse multidimensionnelle, le data mining et les outils de business intelligence (BI).
- L'outil ETL (Extract Transform and Load) appelé SQL Server Integration Services (SSIS), qui facilite l'extraction, la transformation et le chargement des données entre différentes sources et destinations, notamment les entrepôts de données.
- L'outil de génération de rapports appelé SQL Server Reporting Services (SSRS), qui permet aux utilisateurs de créer, de gérer et de partager des rapports personnalisés dans différents formats, en exploitant les ressources du moteur décisionnel.
- Le système de planification de travaux et de gestion d'alertes appelé Agent SQL permet aux administrateurs de planifier et d'automatiser les tâches de maintenance et de surveillance de la base de données.



FIGURE 4.8 – Logo de Microsoft SQL Server

#### 4.2.6 Git/Github

Git est un système de contrôle de version distribué largement utilisé dans le développement logiciel collaboratif. Il permet aux développeurs de suivre les modifications apportées au code source de leur projet, de coordonner le travail en équipe et de gérer les différentes versions de leur application de manière efficace.

GitHub, quant à lui, est une plateforme web populaire basée sur Git, offrant des fonctionnalités avancées pour l'hébergement, la gestion et la collaboration sur des projets Git. En utilisant GitHub, les développeurs peuvent publier leur code source, suivre les problèmes, proposer des modifications (pull requests), collaborer avec d'autres développeurs et gérer le processus de développement de bout en bout.





FIGURE 4.9 – Logo de Git et Github

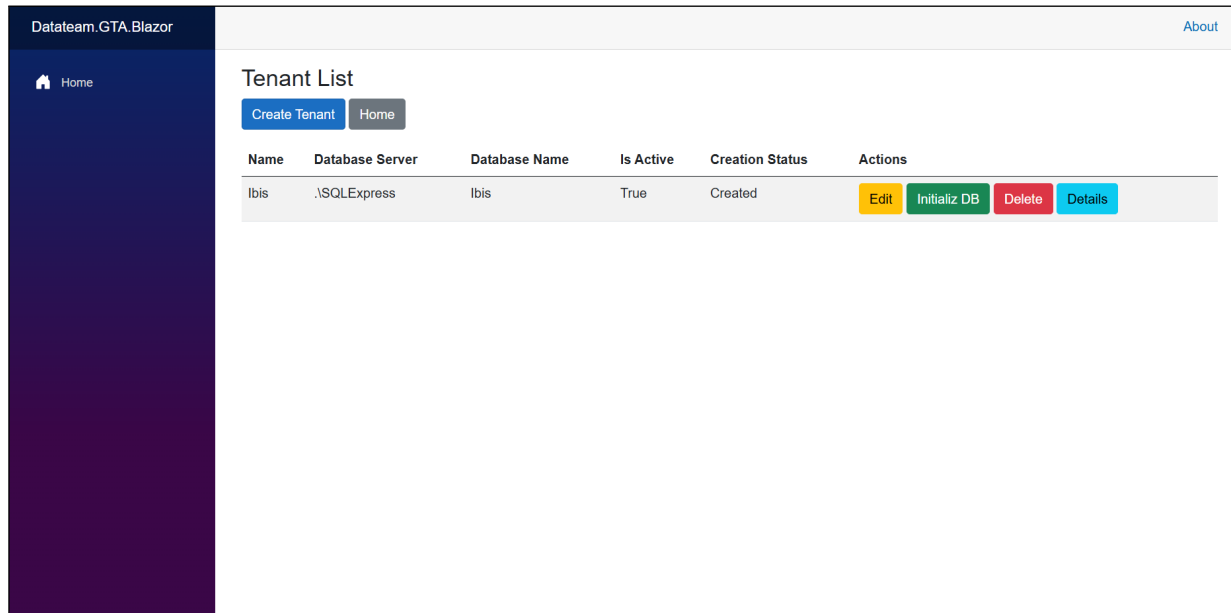
### 4.3 Les IHM de L'application

Cette page présente l'interface d'authentification de l'application. Elle comporte deux champs de saisie essentiels : un pour le nom d'utilisateur et un autre pour le mot de passe. Cette conception sobre et fonctionnelle permet aux utilisateurs de se connecter de manière sécurisée à leur compte, assurant ainsi la protection des données et l'accès personnalisé aux fonctionnalités de l'application.

The image shows a login form within a light blue rectangular frame. The form itself is white and contains the following elements: a label "UserName:" followed by a text input field containing the text "ibish6813admin"; a label "Password:" followed by a password input field with masked characters "\*\*\*\*\*"; and a blue "Login" button positioned below the password field. To the right of the form is a vertical rectangular area with a blue-to-cyan gradient.

FIGURE 4.10 – Page de connexion

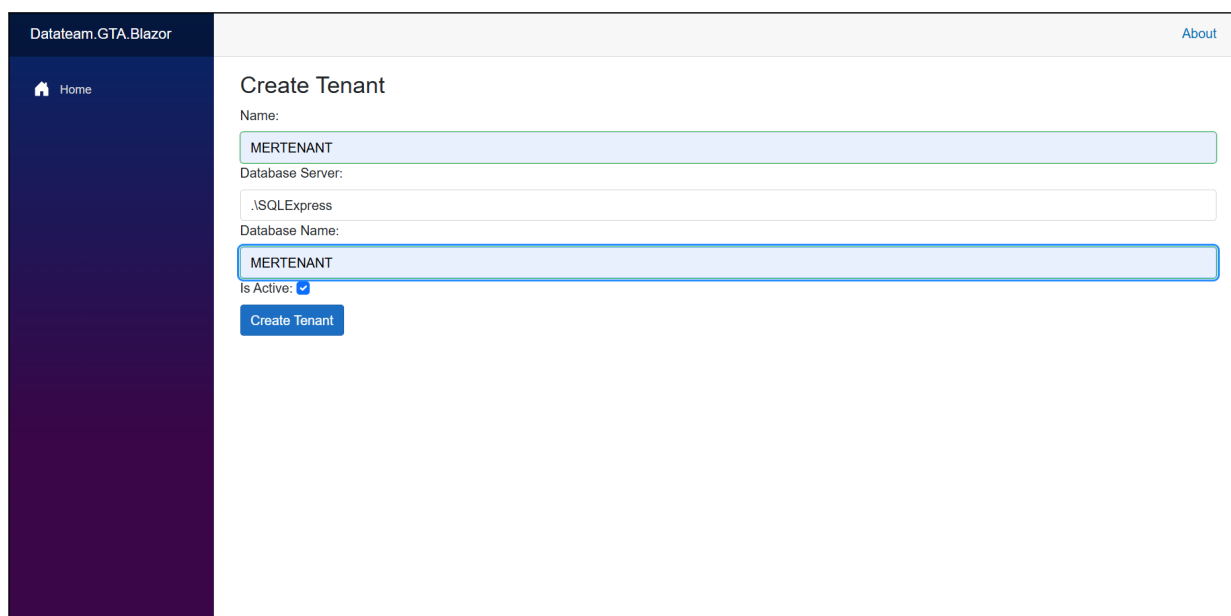
Cet écran affiche un tableau récapitulatif des différents tenants du système. Chaque colonne du tableau présente les informations clés d'un tenant : son nom, le serveur de base de données associé, le nom de la base de données, son statut d'activation, l'état de sa création, ainsi que les actions possibles.



Name	Database Server	Database Name	Is Active	Creation Status	Actions
Ibis	.SQLExpress	Ibis	True	Created	<a href="#">Edit</a> <a href="#">Initializ DB</a> <a href="#">Delete</a> <a href="#">Details</a>

FIGURE 4.11 – Liste des tenants

Cette page permet l'ajout d'un nouveau tenant dans le système. Elle propose un formulaire avec des champs pour saisir les informations nécessaires à la création d'un tenant, telles que le nom, serveur de la base de données, et d'autres options de configuration.



**Create Tenant**

Name:

Database Server:

Database Name:

Is Active: ☒

[Create Tenant](#)

FIGURE 4.12 – Création d'un tenant

Cet écran présente un tableau listant les organisations enregistrées dans le système. Chaque colonne affiche le code et le nom de l'organisation, ainsi que les actions possibles.

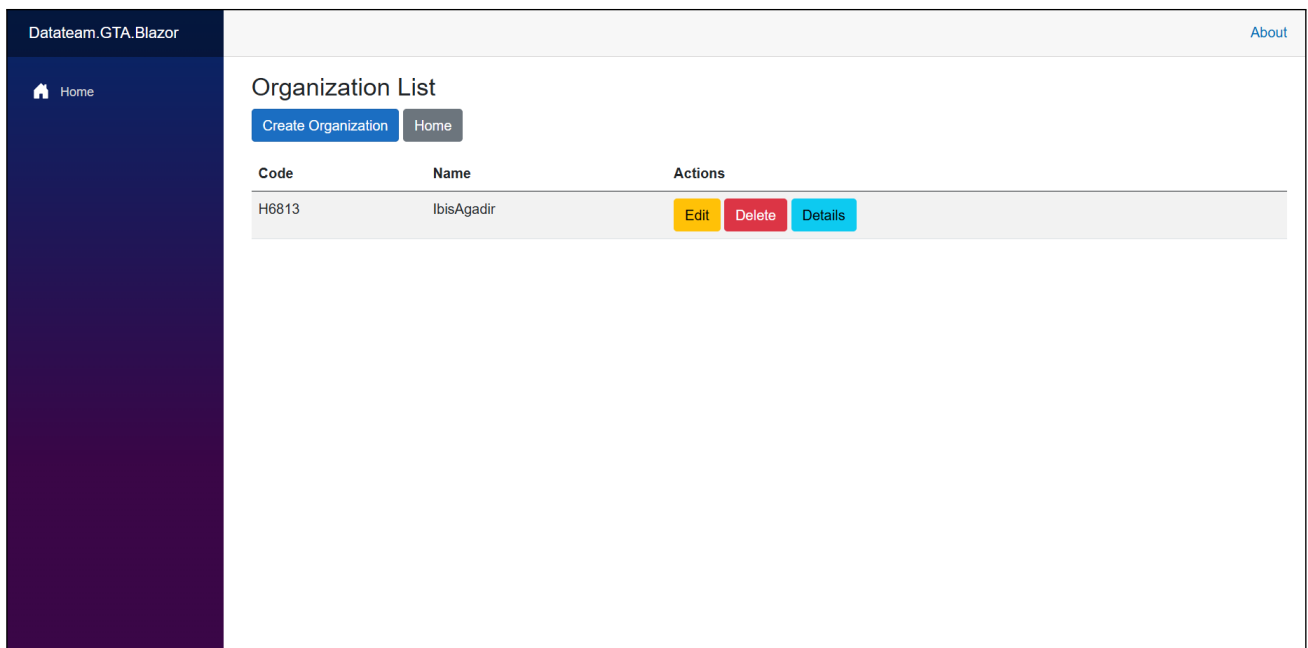


FIGURE 4.13 – Liste des organisation

Cette page comporte un formulaire pour ajouter une nouvelle organisation. Les champs incluent le code, le nom, la chaîne de connexion Sage Paie, booking folder et master data folder.

Code:

Name:

Sage Paie Conn String:

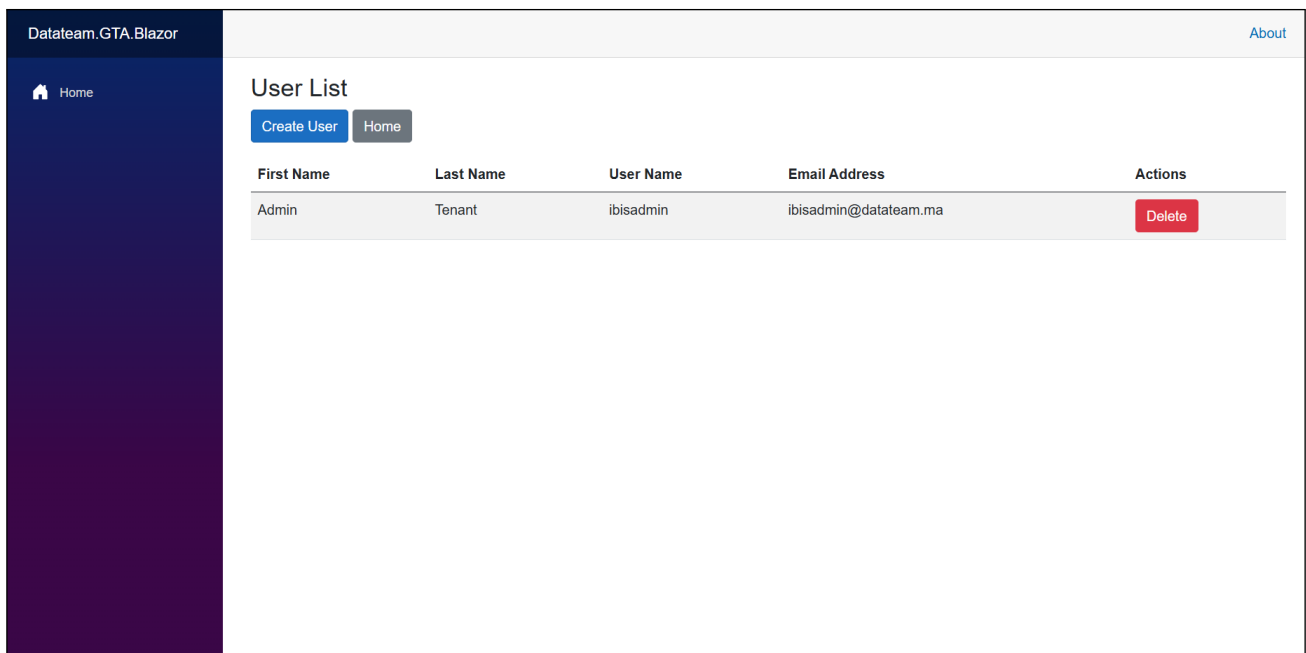
Booking Folder:

Master Data Folder:

[Create Organization](#)

FIGURE 4.14 – Création d'un organisation

Un tableau affichant les informations essentielles des utilisateurs : prénom, nom, nom d'utilisateur, email, et une option de suppression.

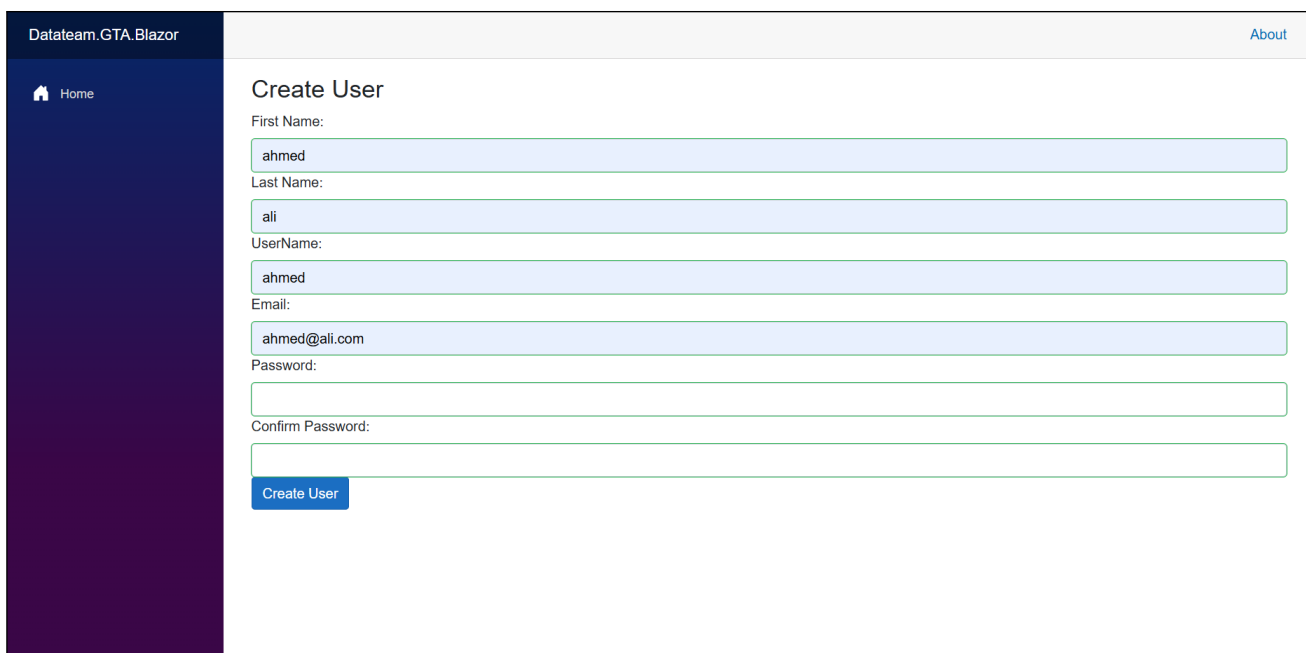


The screenshot shows a web application interface with a dark blue sidebar on the left containing a 'Home' link. The main content area is titled 'User List' and includes a 'Create User' button and a 'Home' button. Below these is a table with the following data:

First Name	Last Name	User Name	Email Address	Actions
Admin	Tenant	ibisadmin	ibisadmin@datateam.ma	<a href="#">Delete</a>

FIGURE 4.15 – List des utilisateurs

Ce formulaire permet l'ajout de nouveaux utilisateurs avec les champs : prénom, nom, nom d'utilisateur, email, mot de passe et confirmation du mot de passe.



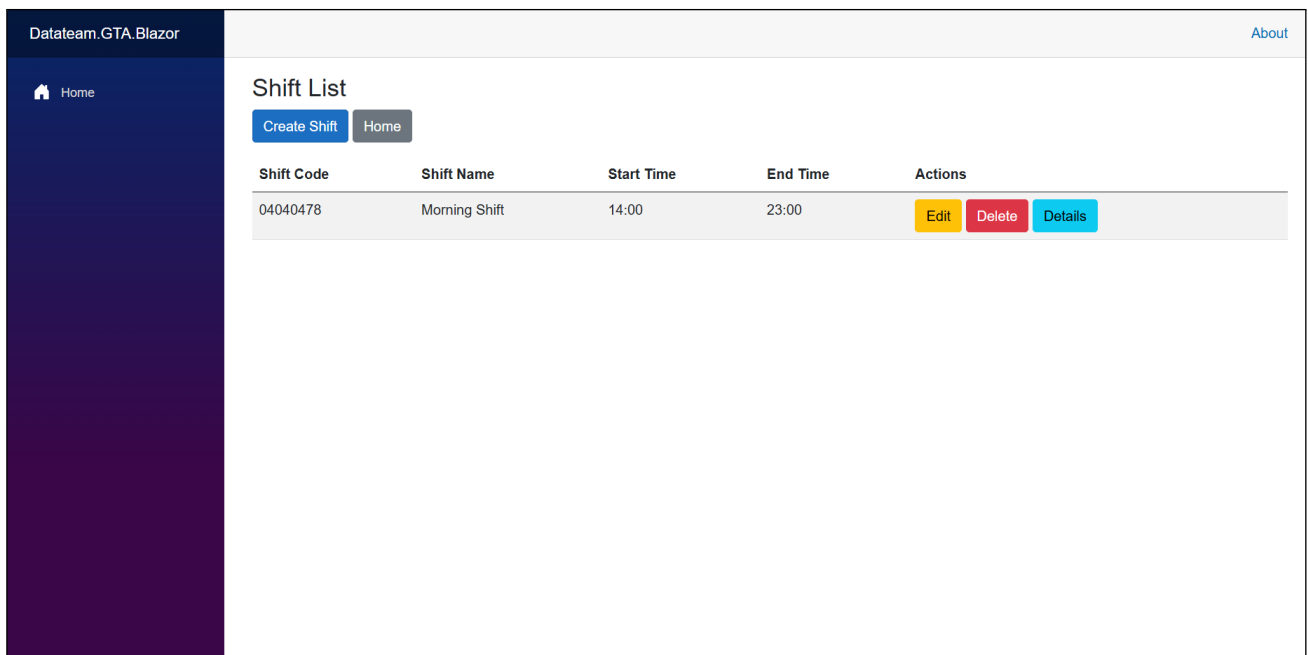
The screenshot shows a web application interface with a dark blue sidebar on the left containing a 'Home' link. The main content area is titled 'Create User' and includes a form with the following fields:

- First Name:
- Last Name:
- User Name:
- Email:
- Password:
- Confirm Password:

Below the form is a 'Create User' button.

FIGURE 4.16 – Création des utilisateurs

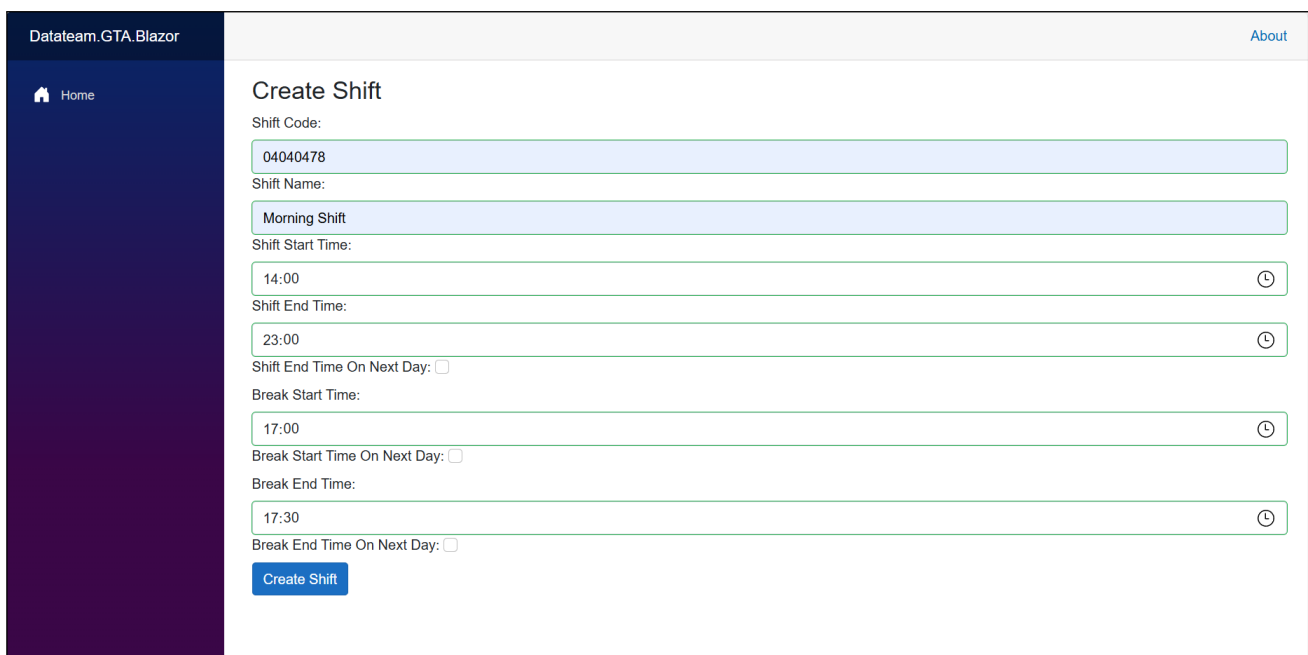
Cet écran présente un tableau des shift de travail, incluant le code, le nom, l'heure de début, l'heure de fin et les actions possibles.



Shift Code	Shift Name	Start Time	End Time	Actions
04040478	Morning Shift	14:00	23:00	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Details</a>

FIGURE 4.17 – List des Shifts

Le formulaire de création de shift inclut les mêmes champs que la liste, plus l'heure de début et de fin de pause.



**Create Shift**

Shift Code:

Shift Name:

Shift Start Time:

Shift End Time:

Shift End Time On Next Day: ☐

Break Start Time:

Break Start Time On Next Day: ☐

Break End Time:

Break End Time On Next Day: ☐

[Create Shift](#)

FIGURE 4.18 – Création d'un Shift

Cette page affiche les informations complètes d'un shift de travail spécifique, avec des options pour modifier ou supprimer le quart. Elle fournit une vue approfondie de la configuration de chaque période de travail.

The screenshot shows a web application interface for 'Datateam.GTA.Blazor'. On the left is a dark blue sidebar with a 'Home' link. The main content area is titled 'Shift Details' and contains a table with the following data:

Shift Code	04040478
Shift Name	Morning Shift
Shift Start Time	14:00
Shift End Time	23:00
Shift End Time On Next Day	False
Break Start Time	17:00
Break Start Time On Next Day	False
Break End Time	17:30
Break End Time On Next Day	False

Below the table are three buttons: 'Edit' (yellow), 'Delete' (red), and 'Cancel' (grey).

FIGURE 4.19 – Détails de Shift

Cette page présente un formulaire pour ajouter un nouvel employé. Les champs incluent le matricule, le numéro de carte, le prénom et le nom.

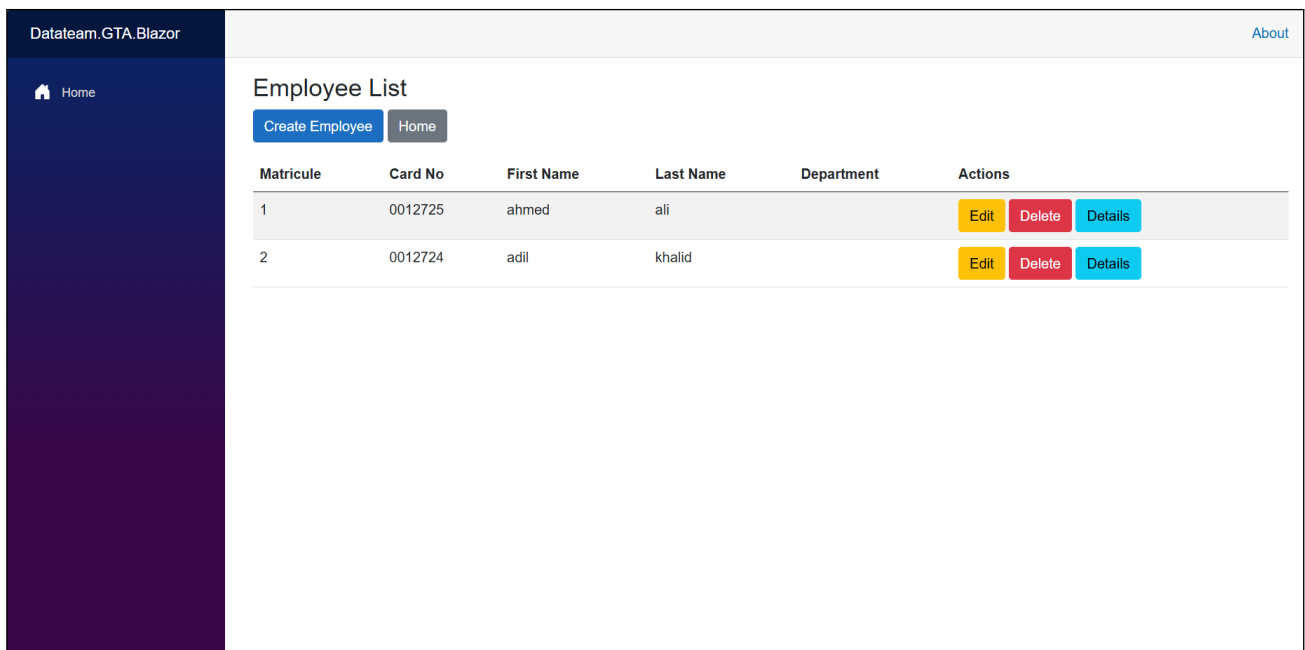
The screenshot shows a web application interface for 'Datateam.GTA.Blazor'. On the left is a dark blue sidebar with a 'Home' link. The main content area is titled 'Create Employee' and contains a form with the following fields:

- Matricule:
- Card No:
- First Name:
- Last Name:

Below the fields is a blue button labeled 'Create Employee'.

FIGURE 4.20 – Création d'un Employé

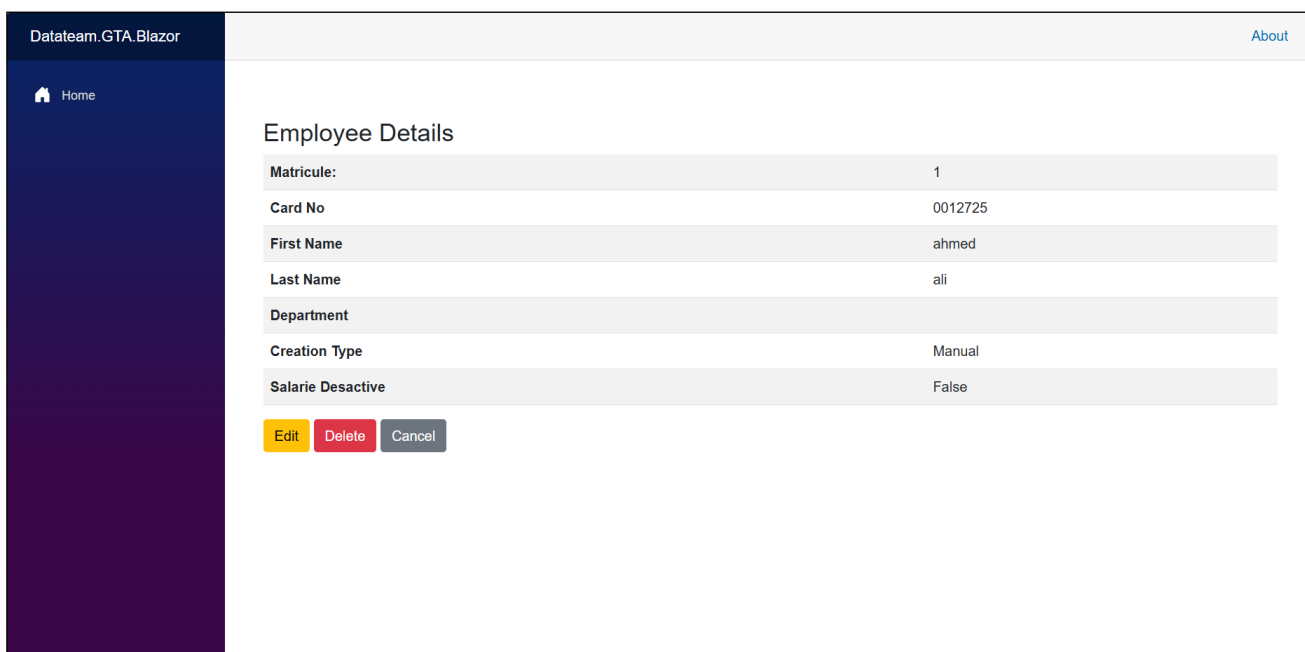
Un tableau affichant les employés avec leurs informations essentielles, y compris le prénom et le nom, le département et les actions possibles.



Matricule	Card No	First Name	Last Name	Department	Actions
1	0012725	ahmed	ali		<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Details</a>
2	0012724	adil	khalid		<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Details</a>

FIGURE 4.21 – Liste des Employé

Cette page montre les informations complètes d'un employé spécifique, incluant un champ de salaire désactivé, le type de création, ainsi que des options pour modifier et supprimer.



Matricule:	1
Card No	0012725
First Name	ahmed
Last Name	ali
Department	
Creation Type	Manual
Salarie Desactive	False

[Edit](#) [Delete](#) [Cancel](#)

FIGURE 4.22 – Détails d'un Employé

Un formulaire simple pour ajouter un nouveau dispositif de pointage, comprenant un code et une description.

The screenshot shows a web application interface with a dark blue sidebar on the left containing a 'Home' link. The main content area is titled 'Create Clock Device'. It features two input fields: 'Code:' with the value 'FA' and 'Description:' with the value 'test'. Below these fields is a blue button labeled 'Create Clock Device'. In the top right corner of the main area, there is a link labeled 'About'.

FIGURE 4.23 – Création de clock device

Un tableau répertoriant tous les dispositifs de pointage avec leurs informations de base et les actions possibles.

The screenshot shows the 'Clock Device List' page. It includes a sidebar with a 'Home' link and a main content area with a title 'Clock Device List'. Above the table are two buttons: 'Create Clock Device' (blue) and 'Home' (grey). The table has three columns: 'Code', 'Description', and 'Actions'. It contains one row with 'FA' as the code and 'test' as the description. The 'Actions' column for this row contains two buttons: 'Edit' (yellow) and 'Delete' (red). An 'About' link is visible in the top right corner of the main area.

Code	Description	Actions
FA	test	<a href="#">Edit</a> <a href="#">Delete</a>

FIGURE 4.24 – Liste des clocks device



Cet écran complexe comprend plusieurs éléments :

- Un bouton pour créer un nouveau pointage.
- Un formulaire de recherche permettant de filtrer les pointages par intervalle de dates et matricule.
- Une option pour calculer la synthèse des pointages sur l'intervalle sélectionné.
- Une liste détaillée des pointages affichant : No Badge, prénom, nom, matricule, terminal, entrée/sortie, date, heure, manuel/informatique, et actions possibles.

Cette interface offre une gestion complète et flexible des données de pointage, permettant à la fois la saisie, la recherche, l'analyse et la manipulation des enregistrements de temps.

The screenshot shows a web application interface for managing time logs. It features a dark blue sidebar on the left with a 'Home' button and an 'About' link in the top right corner. The main content area is titled 'Time Logs List' and includes a 'Create TimeLog' button and a 'Home' button. Below these are search filters: 'Matricule' (a text input), 'Start Date Filter' (a date picker showing '2024-07-04'), and 'End Date Filter' (a date picker showing '2024-08-04'). There are also 'Search' and 'Recalculer la Synthèse' buttons. At the bottom, a table is displayed with the following columns: 'No Badge', 'Prénom', 'Nom', 'Matricule', 'Terminal', 'E/S', 'Date', 'Heure', 'M/I', and 'Actions'. The table is currently in a 'Loading...' state, indicated by a grey bar at the bottom of the table header.

FIGURE 4.25 – Liste des Time Logs

## 4.4 Conclusion

Ce chapitre de Mise en Œuvre a offert un aperçu détaillé du processus de développement de l'application, mettant en lumière les technologies, outils et pratiques qui ont transformé la conception en une solution fonctionnelle.

L'environnement de développement a été exploré, en commençant par le langage C# et l'IDE Visual Studio 2022, qui forment le cœur de la stack technologique. La plateforme .NET et les frameworks associés ont fourni une base solide pour construire une application robuste et évolutive. L'utilisation de Microsoft SQL Server comme système de gestion de base de données assure

la fiabilité et la performance de stockage de données, tandis que Git et GitHub ont facilité la collaboration et le contrôle de version tout au long du développement.

L'intégration de Serilog pour la journalisation a ajouté une couche essentielle de surveillance et de débogage, renforçant la maintenabilité et la fiabilité de l'application.

La présentation des interfaces homme-machine (IHM) a démontré comment les concepts abstraits se sont matérialisés en une expérience utilisateur concrète et intuitive, reflétant les objectifs fonctionnels et ergonomiques définis dans les phases précédentes du projet.

En conclusion, ce chapitre a illustré comment les choix techniques et architecturaux se sont traduits en une implémentation pratique et efficace. Il souligne la synergie entre la conception théorique et la réalisation concrète, marquant une étape cruciale dans le développement de l'application. Cette mise en œuvre jette les bases d'un produit robuste, évolutif et prêt à répondre aux besoins des utilisateurs, tout en ouvrant la voie aux futures améliorations et itérations du projet..

# Conclusion Générale

Au terme de ce rapport, un parcours complet allant de la présentation de l'entreprise d'accueil jusqu'à la mise en œuvre concrète du projet est décrit. Ce stage de fin d'études a été une expérience enrichissante, mêlant théorie et pratique dans un contexte professionnel réel.

L'immersion au sein de l'entreprise a offert une perspective précieuse sur les défis et les opportunités du monde professionnel. Cette expérience a permis d'appliquer les connaissances académiques acquises tout en acquérant de nouvelles compétences techniques et des soft skills essentielles dans le domaine du développement logiciel.

L'immersion au sein de l'entreprise a offert une perspective précieuse sur les défis et les opportunités du monde professionnel. Cette expérience a permis d'appliquer les connaissances académiques acquises tout en acquérant de nouvelles compétences techniques et des soft skills essentielles dans le domaine du développement logiciel.

Le projet réalisé durant ce stage a suivi un processus complet, de l'analyse des besoins à la mise en œuvre technique. L'étude approfondie des cas d'utilisation, suivie de la conception des diagrammes de classes et de séquence, a posé les bases solides d'une architecture robuste. L'étude technique a permis d'explorer des concepts avancés comme le Multi-Tenancy, la Clean Architecture et le Domain-Driven Design, enrichissant considérablement la qualité et la scalabilité de la solution développée.

La phase de mise en œuvre a concrétisé ces concepts en une application fonctionnelle, utilisant des technologies modernes telles que C#, .NET, et divers frameworks. L'utilisation d'outils comme Visual Studio, SQL Server, et Git a renforcé les compétences en développement et en gestion de projet.

Ce stage a non seulement abouti à la réalisation d'un projet concret, mais a également contribué à une croissance personnelle et professionnelle significative. Les défis rencontrés et surmontés ont renforcé la capacité à travailler en équipe, à résoudre des problèmes complexes et à s'adapter à un environnement professionnel en constante évolution.

En conclusion, cette expérience de stage de fin d'études a été une étape cruciale dans la transition entre le monde académique et professionnel. Elle a fourni une base solide pour une future carrière dans le développement logiciel, tout en démontrant l'importance de l'apprentissage continu et de l'adaptation dans un domaine technologique en constante évolution.

# Bibliographie

- [1] *Le processus de développement : 2TUP*. [https://www.memoireonline.com/05/13/7195/m\\_Mise-en-place-dune-application-webmapping-de-geolocalisation-des-points-dintert-de-la-vill6.html/](https://www.memoireonline.com/05/13/7195/m_Mise-en-place-dune-application-webmapping-de-geolocalisation-des-points-dintert-de-la-vill6.html/). 2012.
- [2] Sandra SUSZTEROVA. *Multi-Tenant Architecture : What You Need To Know*. <https://www.gooddata.com/blog/multi-tenant-architecture/>. 2023.
- [3] Daniel DEUTSCH. *A quick introduction to clean architecture*. <https://www.freecodecamp.org/news/a-quick-introduction-to-clean-architecture-990c014448d2/>. 2018.
- [4] <https://redis.io/glossary/domain-driven-design-ddd/>.
- [5] Pranaya ROUT. *Generic Repository Pattern in C with Examples*. <https://dotnettutorials.net/lesson/generic-repository-pattern-csharp-mvc/>. 2019.
- [6] Coursera STAFF. *C Programming : What It Is, How It's Used + How to Learn It*. <https://www.coursera.org/articles/c-sharp>. 2024.
- [7] *What is Visual Studio ?* <https://learn.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2022>. 2024.
- [8] *Qu'est-ce que .Net ?* <https://aws.amazon.com/fr/what-is/net/>.
- [9] Claudio BERNASCONI. *Blazor Basics : What is Blazor—Introduction to Blazor Development*. <https://www.telerik.com/blogs/blazor-basics-introduction-blazor-development>. 2023.