acensi

MCCI BUSINESS SCHOOL
A tradition of excellence

RAPPORT DE STAGE

Fait par
Dimitri Méyépa

BTS SIO

MCCI Business School

Année 2025

SOMMAIRE

INTRODUCTION

Dans le cadre de ma formation en Brevet de Technicien Supérieur Services Informatiques aux Organisations (BTS SIO), j'ai eu l'opportunité d'effectuer un stage en entreprise d'une durée de six semaines, du 2 juin 2025 au 11 juillet 2025, au sein de la société Acensi Mauritius.

Ce stage avait pour objectif principal de me permettre de mettre en pratique les compétences acquises durant ma formation et de développer de nouvelles aptitudes techniques et professionnelles. Mon rôle a été orienté vers le métier de développeur fullstack, ce qui m'a conduit à travailler sur différentes étapes du cycle de développement applicatif, tant du côté front-end que back-end.

Cette expérience m'a permis non seulement de découvrir le fonctionnement concret d'une entreprise spécialisée dans les services numériques, mais également d'approfondir mes connaissances techniques, de travailler en équipe et de mieux comprendre les exigences liées au développement informatique en milieu professionnel.

1.PRÉSENTATION DU PROJET

1.1 Qui est Acensi Mauritius ?

ACENSI Mauritius est la filiale mauricienne du groupe ACENSI, ouverte pour répondre aux besoins croissants en services numériques. Elle propose des solutions en développement logiciel, ingénierie informatique et support technique pour des clients locaux et internationaux. Ses activités couvrent aussi bien le développement front-end et back-end que l'intégration de systèmes et la gestion de projets IT.

1.2 Demande du client

Lors de l'entretien avec le client, qui n'était qu'autre que la directrice d'Acensi Mauritius. Il nous a été demandé de développer un CRM (Customer Relationship Management) afin de créer une application permettant à une entreprise de gérer ses relations avec ses clients. Le client qui possède déjà une maquette de CRM veut la rendre fonctionnelle et dynamique. En ayant la possibilité d'ajouter, modifier et supprimer les informations présentes dans le CRM.

1.3 CAHIER DE CHARGES

Le projet visait à concevoir un CRM (Customer Relationship Management) destiné à Acensi Maurice, afin de centraliser et d'optimiser la gestion des informations clients. Les principales fonctionnalités attendues étaient :

- La gestion complète des contacts et clients (ajout, modification, suppression)
- Le suivi des interactions et activités liées à chaque client
- La production de rapports et de statistiques sur les suivis clients et projets
- Une interface utilisateur intuitive et responsive, adaptée aux besoins internes
- Une base de données sécurisée et évolutive garantissant l'intégrité des informations

Pour celà nous utilisons, pour le front-end, Angular, pour le back-end Java avec Spring Boot et enfin pour la base de donnée Postgresql. Nous utilisons IntelliJ comme IDE pour le back-end et Vs Code pour le front-end.



1.4 Mise en place du projet et répartition des tâches

Le projet CRM a été réalisé en équipe de sept stagiaires, chaque membre étant responsable d'un module spécifique (contacts, deals, projets, user...).

Afin d'assurer une bonne répartition des tâches et un suivi efficace de l'avancement, nous avons utilisé JIRA comme outil de gestion de projet.

- Chaque tâche a été enregistrée et assignée à un membre de l'équipe.
- Un calendrier de réalisation a été établi afin de respecter les délais et les jalons du projet.
- Le suivi des tâches se faisait à l'aide de tableaux Kanban, permettant de visualiser l'avancement (À faire / En cours / Terminé).
- Des réunions quotidiennes étaient organisées pour synchroniser les progrès de l'équipe et résoudre les éventuelles difficultés.

Cette organisation a permis de répartir le travail de manière équilibrée et d'assurer une bonne coordination entre les différents modules du CRM.

Pour centraliser et partager nos développements, nous avons mis en place deux dépôts GitHub distincts :

- Un dépôt pour le Front-end
- Un dépôt pour le Back-end

Chaque membre de l’équipe réalisait ses développements sur des branches dédiées, puis procédait à des pull requests pour intégrer son travail dans la branche principale après validation par la cheffe de projet.

Dans mon cas, lors de la semaine de formation nous avons travaillé à trois personnes sur le module project afin de comprendre Spring Boot. Puis nous avons été séparés et j’ai travaillé sur le module Deal puis le module Task.

2. ANALYSE ET RÉALISATION DU PROJET

2.1 Objectif global du projet

L'objectif principal du projet CRM était de fournir à l'entreprise un outil centralisé et efficace pour la gestion de ses relations clients. Ce système devait permettre de regrouper et d'organiser toutes les informations liées aux clients, aux projets et aux opportunités commerciales au sein d'une seule plateforme.
Les objectifs globaux étaient les suivants :

- Centraliser les données : disposer d'une base unique regroupant les informations clients, les projets, les deals et les tâches.
- Assurer la cohérence entre les modules : établir des liens fonctionnels entre les différents modules (par exemple, relier un deal à un projet, ou rattacher une tâche à un contact).
- Faciliter le suivi des activités : offrir une vision claire de l'état d'avancement des projets, des opportunités et des actions à mener.
- Améliorer la collaboration interne : permettre à plusieurs utilisateurs d'intervenir sur les mêmes données, tout en gardant une traçabilité des actions.
- Fournir une interface intuitive et responsive : rendre l'application accessible et ergonomique, aussi bien sur ordinateur que sur mobile.
- Possibilité d'exporter en format PDF et Excel le contenu de chaque module.

2.2 Analyse des besoins du module “Project”

Lors de la semaine de formation j’ai travaillé sur l’analyse du module Project en équipe de 3. Dans le cadre de l’analyse du module Project, notre objectif était de définir les informations essentielles nécessaires à sa mise en place, ainsi que ses liens avec les autres modules du CRM.

Champs à remplir	Champs à récupérer dans les autre modules
• Le nom du projet • L’ID du projet • le type de projet • la catégorie • le temps du projet • le prix • La date de début et la date de fin • la priorité • le statut • la description •	• Company pour “client” • User pour “team leader” et "" responsible person” • Pipeline stage

Nous remarquons aussi que pas tous les champs sont affichés il faut donc faire en sorte qu’on affiche que les champs qu’on veut afficher.

2.3 Analyse des besoins du module “Deal”

À l’issue de la première semaine de formation consacrée à la découverte des outils et des technologies utilisés dans le projet, la cheffe de projet a procédé à la répartition définitive des responsabilités au sein de l’équipe.
Chaque membre s’est vu attribuer un module spécifique du CRM afin d’approfondir son développement et d’assurer une progression efficace en parallèle.
Pour ma part, j’ai été désigné responsable du module Deal, qui occupe une place centrale dans le CRM puisqu’il permet de gérer les opportunités commerciales, leur suivi dans le pipeline et leurs liens avec les autres entités

Champs à remplir	**Champs à récupérer dans les autre modules**
• Le nom du Deal • Pipeline • Le statut • Le deal value • currency • Period • Expected Closing Date • Source • Tags • Priority • Description	• Pipeline stage • Contact • Project • User

2.4 Réalisation et développement du module “Deal”

Back-end

Le développement du module Deal a commencé par la mise en place de la partie Back-end, en respectant une architecture claire et organisée.

1. Création de l’entité Deal
    - Nous avons défini l’entité Deal avec l’ensemble des champs nécessaires (nom, valeur, devise, statut, dates, priorité, description, etc.). Avec une relation one to many avec les autres
    - Cette entité représente directement la table en base de données.
2. Mise en place des DTO (Data Transfer Object)
    - Le DealRequest permet de recevoir les données envoyées par l’utilisateur lors de la création ou de la mise à jour d’un deal.
    - Le DealResponse permet de renvoyer des données formatées au Front-end.
    - L’utilisation des DTO permet de sécuriser les échanges et de séparer les données internes de la base de données des données exposées à l’extérieur
3. Création du Mapper
    - Le DealMapper est chargé de transformer une entité Deal en DealResponse et inversement (ou en DealRequest).
    - Il assure une meilleure lisibilité et réutilisabilité du code, tout en limitant les duplications.

Mise en place du DealService

Le DealService a été créé pour centraliser la logique métier liée aux deals.

Il contient toutes les fonctionnalités du CRUD (Create, Read, Update, Delete) :

- createDeal() : pour ajouter un nouveau deal.
- getDealById() et getAllDeals() : pour récupérer un ou plusieurs deals.
- updateDeal() : pour modifier les informations d'un deal existant.
- deleteDeal() : pour supprimer un deal.

Le service permet de séparer la logique métier de la gestion des requêtes HTTP, rendant le code plus maintenable et évolutif.

4.Tests unitaires (JUnit)

- Des tests unitaires ont été réalisés sur les services associés au module Deal.
- L'objectif est de vérifier que chaque méthode fonctionne correctement (création, mise à jour, suppression, récupération).
- Cela garantit la fiabilité et la maintenabilité du code.

5.Création du Controller et mise en place de l'API

- Un DealController a été développé pour exposer les endpoints REST de l'API (CRUD : Create, Read, Update, Delete).
- Chaque méthode du controller correspond à une route testable depuis un client HTTP.

6.Tests avec Postman

- Une fois le back-end en place, nous avons testé les différentes routes (POST, GET, PUT, DELETE) grâce à l'outil Postman.
- Ces tests permettent de valider le bon fonctionnement de l'API et de s'assurer que les données envoyées/reçues correspondent bien aux besoins fonctionnels.

Après avoir réalisé le back-end qui a été validé au préalable par la cheffe de projet, l'objectif est de l'intégrer dans le front-end pour rendre le CRM opérationnel.

Front-end

Après la mise en place et les tests du back-end, l'étape suivante a été l'intégration du module Deal dans la partie Front-end Angular. L'objectif était de permettre aux utilisateurs d'interagir avec l'API et de gérer les deals via une interface ergonomique.

1. Création du Component Deal
    - Un component dédié (DealComponent) a été créé pour afficher la liste des deals.
    - D'autres components spécifiques ont été ajoutés :
        - DealFormComponent pour l'ajout et la modification d'un deal.
        - DealDetailsComponent pour la visualisation détaillée d'un deal.
2. Service Angular pour l'API
    - Un service DealService a été développé afin de centraliser les appels HTTP vers l'API (via HttpClient).
    - Les méthodes principales du service correspondent aux opérations CRUD :
        - getDeals() → récupérer la liste des deals
        - getDealById(id) → récupérer un deal précis
        - createDeal() → créer un nouveau deal
        - updateDeal() → mettre à jour un deal existant
        - deleteDeal() → supprimer un deal
3. Connexion avec l'API
    - Les components consomment les méthodes du DealService pour communiquer avec le back-end.
    - Les données sont affichées dynamiquement grâce au data binding et aux directives Angular (*ngFor, *ngIf).

4. Formulaires réactifs (Reactive Forms)
   - Pour l’ajout et la modification d’un deal, des formulaires réactifs ont été mis en place avec FormBuilder.
   - Ces formulaires permettent de gérer facilement la validation des champs (nom obligatoire, valeur numérique, date valide, etc.).
5. Tests et validation
   - Après l’intégration, l’ensemble des fonctionnalités du module Deal a été testé côté front pour s’assurer du bon fonctionnement avec l’API.
   - Ces tests ont validé la cohérence entre les données saisies dans l’interface et celles stockées en base via le back-end.

L’intégration dans le front-end était la partie dans laquelle j’ai rencontré plusieurs difficultés. C’était ma première expérience avec Angular et j’ai eu beaucoup de mal au début à intégrer le CRUD surtout la partie Create et Update. Une erreur dans l’intégration du Create bloquait l’update mais tout s’est mieux passé ensuite et j’ai appris de ces erreurs.

Après avoir fini l’intégration dans le front, j’ai fait un test complet du module Deal avec la cheffe de projet. Le test s’est très bien passé, j’ai réussi à respecter la demande et les tâches qu’on m’a donné.

2.5 Analyse des besoins du module “Task”

Après avoir terminé le développement complet du module Deal (back-end et intégration front-end), j’ai poursuivi mon travail en prenant en charge le module Task.
Ce module avait pour objectif de permettre la gestion des tâches liées aux autres entités du CRM. Une tâche peut être associée à un Deal, à un Projet ou encore assignée à un Utilisateur.

Champs à remplir	Champs à récupérer dans les autre modules
• Title • Category • Start Date • Due Date • Tags • Priority • Status • Description	• User pour “responsible person”

Remarque : les tâches sont rangées par date de création. Quand on crée une tâche aujourd’hui elle se met dans “recent” et ensuite le lendemain elle se met automatiquement dans “yesterday” puis se range par sa date par exemple une tâche crée le 20 juin 2025 se range dans “20 june 2025”

2.4 Réalisation et développement du module “Deal”

Back-end

Le développement du module Task a suivi la même logique que pour le module Deal, avec une architecture claire et bien organisée.

1. Création de l’entité Task

- L’entité Task contient tous les champs nécessaires
- Cette entité correspond directement à la table Task en base de données et permet de gérer la persistance des tâches.

2. Mise en place des DTO (Data Transfer Object)

Deux DTO ont été définis :

- TaskRequest : reçoit les données lors de la création ou modification d’une tâche.
- TaskResponse : renvoie des données formatées au Front-end.

Ces DTO permettent de sécuriser les échanges et d’éviter d’exposer directement l’entité.

3. Création du Mapper

- Le TaskMapper transforme une entité Task en TaskResponse et inversement en TaskRequest.
- Il centralise la logique de conversion, améliore la lisibilité et évite les duplications dans le code.

4. Mise en place du TaskService

Le TaskService regroupe toute la logique métier autour des tâches et contient les fonctionnalités CRUD :

- createTask() : ajoute une nouvelle tâche.
- getTaskById() : récupère une tâche par son identifiant.
- getAllTasks() : retourne la liste complète des tâches.
- updateTask() : met à jour une tâche existante.
- deleteTask() : supprime une tâche.

6. Mise en place du Controller
Le TaskController expose les fonctionnalités via des endpoints REST :

- POST /api/tasks → création d'une tâche.
- GET /api/tasks/{id} → récupération d'une tâche par son identifiant.
- GET /api/tasks → récupération de toutes les tâches.
- PUT /api/tasks/{id} → mise à jour d'une tâche.
- DELETE /api/tasks/{id} → suppression d'une tâche

7. Tests avec Postman
Une fois le controller et l'API mis en place, l'ensemble des endpoints a été testé sur Postman pour vérifier :

- L'exactitude des données envoyées et reçues.
- La cohérence entre les informations saisies côté Front-end et celles stockées en base.
- La gestion correcte des erreurs (champs manquants, identifiants inexistants, etc.).

Après la validation du back-end par la cheffe de projet, je me suis maintenant occupé de l'intégration dans le front-end de Task.

Front-end

1. Création du component AllTasksComponent

Le component AllTasksComponent est le cœur de l'affichage et de la gestion des tâches. Il gère :

- la récupération des tâches depuis le back-end via le service DataService ;
- l'affichage dans un tableau paginé (MatTableDataSource) ;
- l'ouverture de sidebars pour créer ou modifier une tâche ;
- le filtrage et la sélection des tâches selon plusieurs critères (titre, catégorie, responsable, tags, date de création).

2. Récupération et affichage des données

- Les tâches sont récupérées via this.data.getTaskList() et stockées dans allTasks et actualData.
- La pagination est gérée avec PaginationService, qui découpe les données pour le tableau (getTableData()).
- Les tâches sont affichées dans un tableau Angular Material, avec les numéros de ligne et les données paginées.

3. Logique de tri par date de création

Pour organiser les tâches par date, le component utilise la méthode classTask(tasks: TaskResponse[]) :

- Les tâches créées aujourd'hui sont stockées dans recentTasks.
- Les tâches créées hier sont stockées dans yesterdayTasks.
- Les tâches plus anciennes sont regroupées par date (olderTasksMap) et triées par ordre décroissant (les plus récentes d'abord).

La méthode isSameDay(d1, d2) compare deux dates pour déterminer si elles correspondent au même jour, et formatDate(date) formate les dates pour un affichage lisible.

Cette logique permet de visualiser rapidement les tâches récentes et de trier automatiquement les anciennes.

4. Filtrage avancé

Le component gère également un système de filtrage multi-critères :

- par titre, catégorie, tags, responsable, et date de création.
- Chaque filtre possède une méthode de chargement (loadDistinctTitles, loadDistinctCategories, etc.) et une logique pour sélectionner ou désélectionner les valeurs.
- L'application des filtres se fait via applyFilter(), qui met à jour actualData et réorganise les tâches avec classTask().

5. Formulaires réactifs

Le component utilise FormGroup pour créer et modifier des tâches :

- createTaskForm contient tous les champs nécessaires (titre, catégorie, responsable, dates, tags, priorité, statut, description).
- Les méthodes createTask() et updateTask() envoient les données au back-end via le service DataService.
- loadTaskIntoForm() charge une tâche existante dans le formulaire pour modification.

6. Actions CRUD et interactions

- Création / Modification : via sidebars et le formulaire réactif.
- Suppression : via modal de confirmation (confirmDeleteTask() / deleteConfirmedTask()).
- Les méthodes communiquent avec le back-end pour effectuer les actions, et les données sont rechargées après modification (reloadTaskListSimple()).

7. Tests et vérification

- Les endpoints ont été testés via Postman pour vérifier la communication avec le back-end.
- Le tri et le filtrage ont été testés en front pour s'assurer que les tâches étaient correctement organisées et affichées.

8. Résultat

L'intégration front-end permet ainsi :

- un affichage clair des tâches avec tri par date ;
- la possibilité de créer, modifier, supprimer et filtrer les tâches ;
- un affichage dynamique et réactif grâce aux formulaires Angular et à Angular Material

3. RESSENTI PERSONNEL DU STAGE

3.1 Compétences acquises

Durant mon stage chez Acensi Mauritius, j'ai pu développer de nombreuses compétences techniques et professionnelles. Sur le plan technique, j'ai approfondi mes connaissances en développement full-stack, en travaillant à la fois sur le back-end avec Java et Spring Boot, et sur le front-end avec Angular. J'ai appris à structurer correctement les modules, créer des entités, des DTO, des services et des mappers, et à assurer la communication entre le front et le back via des API REST sécurisées.

L'utilisation des outils de collaboration et de gestion de projet, tels que JIRA pour le suivi des tâches et GitHub pour la gestion du code source, m'a permis de comprendre l'importance de l'organisation et du travail en équipe dans un projet réel. J'ai également découvert l'efficacité des tests unitaires (JUnit) et des tests d'API via Postman, qui garantissent la fiabilité et la qualité du code.

Sur le plan organisationnel et méthodologique, j'ai appris à travailler selon une approche agile, avec un découpage précis des tâches, des deadlines définies et une coordination avec mes collègues. Cette expérience m'a permis de mieux comprendre la logique de développement d'un projet CRM complet, ainsi que la nécessité de maintenir une architecture claire et cohérente entre les différents modules.
Enfin, j'ai développé ma capacité à résoudre des problèmes et à trouver des solutions face à des défis techniques, notamment pour l'intégration des modules Deal et Task, la mise en place de filtres complexes et le tri par date de création des tâches. Ces apprentissages me seront très utiles pour mes futurs projets professionnels et pour ma carrière en développement logiciel.

3.2 Appréciation personnelle

Durant mon stage, j'ai particulièrement apprécié l'ambiance de travail et l'accueil au sein de l'entreprise. Le personnel était toujours très respectueux, à l'écoute et disponible pour répondre à mes questions ou m'aider en cas de difficulté. Cette attitude m'a permis de me sentir rapidement intégré à l'équipe et de travailler dans un environnement motivant et stimulant.

J'ai également pu développer mes compétences relationnelles et personnelles, notamment en matière de communication, de travail en équipe et de gestion du temps. Être en contact quotidien avec mes collègues m'a appris l'importance de la collaboration, de l'échange d'idées et du partage des connaissances.

CONCLUSION

Ce stage chez Acensi Mauritius a été une expérience très enrichissante, tant sur le plan professionnel que personnel. Il m'a permis de mettre en pratique mes connaissances théoriques acquises au cours de ma formation en BTS SIO, en travaillant sur un projet concret et complet : le développement d'un CRM pour la gestion des relations clients.

J'ai pu développer et consolider mes compétences techniques, notamment en Java, Spring Boot, Angular et dans l'utilisation d'outils de gestion et de collaboration comme JIRA et GitHub. Le travail sur différents modules tels que Project, Deal et Task m'a permis de comprendre l'importance d'une architecture bien pensée et d'une communication efficace entre le back-end et le front-end.

Au-delà des aspects techniques, ce stage m'a également apporté une grande valeur humaine. L'écoute, le respect et le soutien constant de l'équipe m'ont permis de progresser dans un environnement motivant et positif. Cette expérience m'a donné confiance en mes capacités et m'a encouragé à poursuivre dans la voie du développement full-stack et de la gestion de projets informatiques.

En somme, ce stage a constitué un véritable tremplin vers mon avenir professionnel, en me permettant de concilier théorie et pratique, tout en développant des qualités personnelles essentielles au travail en entreprise.