ChatGPT

Architecture globale du projet

Le système suit une **architecture microservices** où chaque fonctionnalité métier (authentification, gestion de projets, facturation, etc.) est déployée en tant que service indépendant. Chaque service back-end (par exemple écrit en Spring Boot) est empaqueté dans son conteneur Docker dédié, de même que la base de données (par exemple PostgreSQL) [1]. Les conteneurs Docker isolent l'environnement d'exécution (OS, dépendances, configuration) et se déploient de façon identique sur tout hôte supportant Docker [1]. Le backend Spring Boot s'exécute ainsi dans un conteneur qui se connecte au conteneur de la base de données via le réseau Docker interne. Concrètement, on utilise souvent **Docker Compose** (pour le développement ou le CI) ou un orchestrateur type Kubernetes (en production) pour lancer simultanément le conteneur BD et les conteneurs de chaque microservice. Le backend accède à la base sur un réseau Docker privé (par exemple `--network`), sans exposer directement le SGBD au public. La connexion conteneur-à-conteneur est courante : *« Connecting to a containerized database from another container is a common scenario in microservices architecture »* [2]. Le point d'entrée externe du système est géré par une **API Gateway** (ou BFF) qui reçoit toutes les requêtes clients et les redirige vers le service approprié [3].

Services par domaine fonctionnel

Le projet est organisé en plusieurs microservices autonomes, chacun étant responsable d'un domaine fonctionnel précis. Par exemple :

- **Service Authentification / Utilisateurs** : gère l'inscription, la connexion et l'émission de tokens JWT, ainsi que l'attribution des rôles (SuperAdmin plateforme, Chef de projet, Développeur) [4]. Ce service centralise la sécurité ; les autres services vérifient le JWT pour identifier l'utilisateur.
- **Service Gestion des Entreprises (Tenants)** : gère la création, l'activation et la désactivation des comptes « entreprise ». Il ajoute un contexte multi-tenant à chaque requête (par exemple via un en-tête HTTP `X-TenantID`) pour isoler les données (schéma ou base séparé) de chaque entreprise [5]. Les autres services filtrent leurs données en fonction de ce contexte « tenant ».
- **Service Gestion de projet** : gère les projets agiles, le backlog, les sprints et les tâches. Il permet de créer et planifier des projets, définir des tâches (description, compétences requises, estimation, livrables, dépendances) et d'assigner dynamiquement des développeurs. Ce service consulte à la volée le service Auth (pour valider l'utilisateur et ses droits) et tient compte du contexte locataire [6].
- **Service Suivi du temps (Time Tracking)** : gère la saisie quotidienne de temps par tâche. Il reçoit les entrées de temps des développeurs, notifie le chef de projet pour validation et stocke le temps réel validé. Pour cela, il interroge le service Projet (pour référencer les tâches) et envoie les données validées au service Facturation (pour calcul des montants) [7].
- **Service Facturation (Gestion financière)** : calcule automatiquement les montants à facturer à partir du temps validé (en appliquant les taux horaires, marges, TVA locale). Il génère les factures au format PDF (numérotation unique, détails projets/tâches), stocke l'historique et vérifie la validation avant émission. Il dépend des données du service Temps (temps validé) et du service Projet (infos client/projet) pour calculer et lister les éléments facturés [8].
- **Service Reporting et BI** : agrège les données de tous les autres services pour produire des indicateurs (CA par projet, marge, temps/charges, graphiques de tableaux de bord). Il peut consommer des événements métiers (fin de sprint, validation du temps, émission de facture) ou interroger en lecture les bases de données agrégées pour constituer les rapports [9].

- **Service Notifications (optionnel)** : écoute les événements métiers des services (tâche affectée, temps saisi validé, facture émise, etc.) et envoie automatiquement des e-mails de rappel ou d'information (par exemple en utilisant un broker/message queue pour recevoir les messages) [10].
- **API Gateway / BFF** : sert de point d'entrée unique pour le front-end. Elle authentifie la requête (vérifie le JWT, gère le contexte locataire) puis redirige vers le microservice concerné [3].
- **Client Web Front-end (React/Angular)** : développé séparément (par rôle utilisateur) et consomme les APIs exposées par la Gateway. Il affiche les tableaux de bord, formulaires Kanban, formulaires de saisie de temps, etc.

Chacun de ces services s'exécute dans son propre conteneur Docker et possède son code et ses propres dépendances. Les communications entre services s'effectuent par API REST ou par messages (events). Les dépendances entre services sont explicites : *Auth* et *Tenant* doivent démarrer en premier, tous les autres services (Projet, Temps, Facturation, Reporting) requièrent leur contexte pour fonctionner. Par exemple, le service Projet dépend de Auth et Tenant ; le service Temps dépend du service Projet ; le service Facturation dépend des services Temps et Projet [11]. Le service Reporting peut quant à lui fonctionner en lecture seule sur les données regroupées. Ce découpage par domaine permet une meilleure évolutivité (chaque service peut être monté en charge indépendamment) et facilite l'isolation des erreurs. Par exemple, une panne dans le module Facturation n'interrompt pas le service Gestion de projet [11].

Isolation des domaines et multi-tenancy

L'architecture applique le principe « responsabilité unique » : chaque microservice gère uniquement son domaine métier [11]. Un service ne manipule pas les données d'un autre domaine (p. ex. le service Projet ne gère ni l'authentification ni la facturation) [11]. Cette séparation rend plus robuste le système : un bug ou une surcharge dans un service n'affecte pas les autres. De plus, chaque service persiste ses propres données et n'utilise pas le même schéma de base que les autres. En effet, *« le partage des bases de données avec d'autres microservices doit être évité, car il rend le dépannage difficile et entraîne de l'incohérence »* [12]. Ainsi chaque service dispose de sa base (ou son schéma) privé, ce qui renforce l'indépendance des équipes et des déploiements.

Le système étant **multi-tenant**, chaque entreprise cliente est isolée au niveau des données. Concrètement, on attribue à chaque tenant sa propre base de données ou son propre schéma au sein de la base [5] [13]. L'isolation est telle qu'aucune donnée locataire n'est accessible par un autre. Cela garantit la sécurité et l'intégrité des données de chaque client. Il peut néanmoins exister un *référentiel global* pour les données communes : certaines tables de référence ou de configuration ayant la même structure pour tous les clients (par exemple codes TVA, listes de statuts, etc.) peuvent être placées dans une base « master » partagée. Chaque service de chaque tenant accéderait alors à ce schéma global pour ces données communes, tout en conservant sa base privée pour les données métier spécifiques. Ce référentiel global unifie les données partagées (mêmes attributs) sans briser l'isolement des locataires.

Sources : L'architecture microservices se fonde sur le découpage par domaines fonctionnels avec communication par APIs [14] [11]. L'utilisation de conteneurs Docker pour le backend et la base de données suit les bonnes pratiques cloud-native [1] [2]. La conception des données privilégie l'isolation par service et par locataire [12] [13], tout en autorisant un schéma global pour les données universelles aux tenants. Ces éléments assurent une architecture claire, scalable et robuste pour une plateforme SaaS multi-tenant.

---

1 Créer des microservices avec conteneurs Docker | .NET
https://dotnet.microsoft.com/fr-fr/apps/aspnet/microservices

2 Use containerized databases | Docker Docs
https://docs.docker.com/guides/databases/

3 4 5 6 7 8 9 10 11 14 Analyse du projet et recommandations MVP SaaS FinTech.pdf
file://file_00000000f88071f481076d5efe3ef4d2

12 Qu'est-ce que les microservices ? | StarTechUP
https://www.startechup.com/fr/blog/what-are-microservices/

13 Qu'est-ce que l'implémentation du multi-tenant SaaS ? Conception de base de données
https://payproglobal.com/fr/reponses/quest-ce-que-le-multi-tenant-saas/