Semah Bahroun
Maxime Blanchard
Dylan Fournier

Projet 2
Architecture des Applications d'Entreprise

PrixBanque - Conception d'architecture

Novembre 2024

Table des matières

1 Introduction

PrixBanque est une nouvelle fintech qui développe une application bancaire fiable, sécurisée et entièrement numérique, destinée aux clients souhaitant éviter les démarches bancaires traditionnelles et simplifier la gestion de leurs finances.

Afin de diversifier ses services et se démarquer, **PrixBanque** cherche à inclure un service de billetterie dans son application pour faciliter la vente de billets pour des événements proposés par divers partenaires (organisations, associations, et autres entités).

L'objectif de ce rapport est de concevoir l'architecture d'une telle application, permettant aux clients de gérer à la fois leurs finances et leurs loisirs depuis une seule interface.
Dans un premier temps, nous détaillerons les exigences de l'application. Ensuite, l'architecture sera définie à travers le modèle C4, proposant différentes vues du système.

2 Exigences du système

2.1 Liste des exigences fonctionnelles

Nous présentons ici une liste des exigences fonctionnelles que l'application doit satisfaire. Chaque exigence est décrite par un titre, une description et un niveau de priorité. Le niveau de priorité est défini comme suit : HIGH, MEDIUM, LOW.
Dans cette architecture, nous nous concentrons principalement sur les exigences de haut niveau, afin d'assurer que les besoins essentiels de l'application sont bien pris en compte.

Requirement 1 : Création de compte
Priorité : **HIGH**
Description : Le système permet à un utilisateur de faire une demande de création de compte en saisissant ses informations personnelles. Ces données sont ensuite vérifiées par le système, qui procède à leur validation.

Requirement 2 : Solde de compte
Priorité : **HIGH**
Description : Le système doit permettre à l'utilisateur de voir le solde de son compte.

Requirement 3 : Relevé de compte
Priorité : **HIGH**
Description : Le système doit permettre de consulter l'historique complet de son compte.

Requirement 4 : Transferts de valeurs
Priorité : **HIGH**
Description : Le système doit permettre le transfert d'argent entre utilisateurs ou vers l'extérieur. L'utilisateur payeur doit fournir les informations suivantes : type de virement (immédiat ou programmé), montant, question de sécurité, réponse, et l'email du destinataire.

Requirement 5 : Création de factures
Priorité : **HIGH**
Description : Le système doit permettre la génération de factures pour solliciter un paiement d'un autre client. L'utilisateur initiant la demande de facture doit fournir les informations suivantes : montant, date d'échéance et email du client payeur.

Requirement 6 : Consultation des factures
Priorité : **HIGH**
Description : Le système doit permettre la consultation des factures, qu'elles soient émises, reçues, payées, impayées ou annulées.

Requirement 7 : Paiement des factures
Priorité : **HIGH**
Description : Le système doit permettre de payer une facture reçue en validant la demande.

Requirement 8 : Exclusion des factures

Priorité : **HIGH**
Description : Le système doit permettre la suppression des factures. Seules les factures non payées et non traitées peuvent être exclues. Avant toute exclusion, une confirmation doit être obtenue de l'émetteur. Les parties concernées seront ensuite informées de l'exclusion.

Requirement 9 : Ajout d'argent
Priorité : **HIGH**
Description : Le système doit permettre à un utilisateur d'ajouter de l'argent sur son compte PrixBanque.

Requirement 10 : Achat de billets
Priorité : **HIGH**
Description : Le système doit permettre d'acheter des billets pour un spectacle.

Requirement 11 : Catalogue
Priorité : **HIGH**
Description : Dans le catalogue, le client doit pouvoir rechercher des événements et spectacles en définissant des critères de recherche.

Requirement 12 : Liste d'attente
Priorité : **HIGH**
Description : En cas de forte demande pour l'achat de billets d'un spectacle, l'utilisateur peut être placé dans une file d'attente.

Requirement 13 : Vérification des disponibilités
Priorité : **HIGH**
Description : Le système doit permettre la vérification en temps réel de la disponibilité des places pour un événement.

Requirement 14 : Proposition d'événements
Priorité : **HIGH**
Description : Une entité doit être capable de présenter son spectacle à l'administration afin de pouvoir mettre en vente des places sur l'application

Requirement 15 : Transfert de billets
Priorité : **HIGH**
Description : Le système doit permettre le transfert de billets vers un destinataire, associé à une facture, afin de faciliter les demandes de remboursement et les transferts de billets entre clients.

Requirement 16 : Billet de spectacle
Priorité : **HIGH**
Description : Le client doit pouvoir visualiser les billets de spectacles qu'il a achetés.

Requirement 17 : Notifications
Priorité : **MEDIUM**
Description : Le système doit notifier l'utilisateur en cas de modifications apportées à un spectacle

pour lequel il a réservé des places. L'utilisateur doit également pouvoir choisir le format de notification (e-mail ou SMS) selon ses préférences.

Requirement 18 : Authentification à deux facteurs
Priorité : **MEDIUM**
Description : L'utilisateur peut activer l'authentification à deux facteurs pour renforcer la sécurité de son compte.

Requirement 19 : Offres Promotionnelles
Priorité : **MEDIUM**
Description : L'application doit permettre aux 10 000 premiers clients de bénéficier d'une somme de 1000$ sur leur compte une fois l'inscription validée.

Requirement 20 : Informations bancaires
Priorité : **MEDIUM**
Description : Le système doit permettre de consulter les informations du compte bancaire.

Requirement 21 : Support client
Priorité : **MEDIUM**
Description : Le système doit permettre à l'utilisateur de contacter le service client en cas de problème ou de besoin d'assistance.

Requirement 22 : Modification du profil
Priorité : **MEDIUM**
Description : Le système doit permettre à l'utilisateur de modifier ses informations personnelles et de mettre à jour son profil.

Requirement 23 : Reçus
Priorité : **LOW**
Description : Le système doit permettre à l'utilisateur de consulter les reçus des opérations effectuées, incluant les virements et les paiements de factures. Un reçu doit fournir tous les détails de l'opération correspondant.

Requirement 24 : Résumé des transactions
Priorité : **LOW**
Description : Le système doit permettre à l'utilisateur de consulter un résumé des transactions effectuées, soit pour l'ensemble des opérations, soit pour un client donné.

Requirement 25 : Contacter l'organisateur
Priorité : **LOW**
Description : Le système doit permettre au client de contacter l'organisateur d'un événement en cas de problème ou de demande spécifique (mobilité réduite, etc.).

Requirement 26 : Sélection de place
Priorité : **LOW**
Description : L'utilisateur doit pouvoir réserver une place spécifique dans le plan de salle pour un

spectacle.

Requirement 27 : Gestion des amis
Priorité : **LOW**
Description : Le système doit permettre à l'utilisateur d'ajouter des amis en utilisant l'adresse mail.

Requirement 28 : Devise
Priorité : **LOW**
Description : Le client peut faire la demande de changer de devise

2.2 Scénarios

Dans cette section, nous présentons le diagramme de cas d'utilisation de l'application (fig.1), qui permet de modéliser les exigences fonctionnelles ainsi que leurs relations avec les acteurs.

Notre diagramme met en avant trois acteurs :

- **Le client** : C'est l'acteur principal de notre application, il initie l'utilisation des fonctionnalités de l'application.
- **L'agence événementielle** : C'est un des acteurs secondaires de l'application. Il est impliqué dans la gestion des événements accessibles au client.
- **La banque** : Le deuxième acteur secondaire qui fournit les services bancaires nécessaires aux opérations financières de l'application.

Dans le diagramme on peut remarquer différentes catégories de fonctionnalités :

- **Fonctionnalités de base** : création de compte, connexion et notifications.
- **Fonctionnalités bancaires** : consultation du solde et du relevé de compte, transfert de fonds.
- **Gestion des factures** : création, paiement et suppression de factures.
- **Gestion des événements** : recherche d'événements et vente de billets.

Pour conclure, ce diagramme illustre les principales interactions entre les acteurs et les fonctionnalités, en insistant sur les exigences fonctionnelles prioritaires.

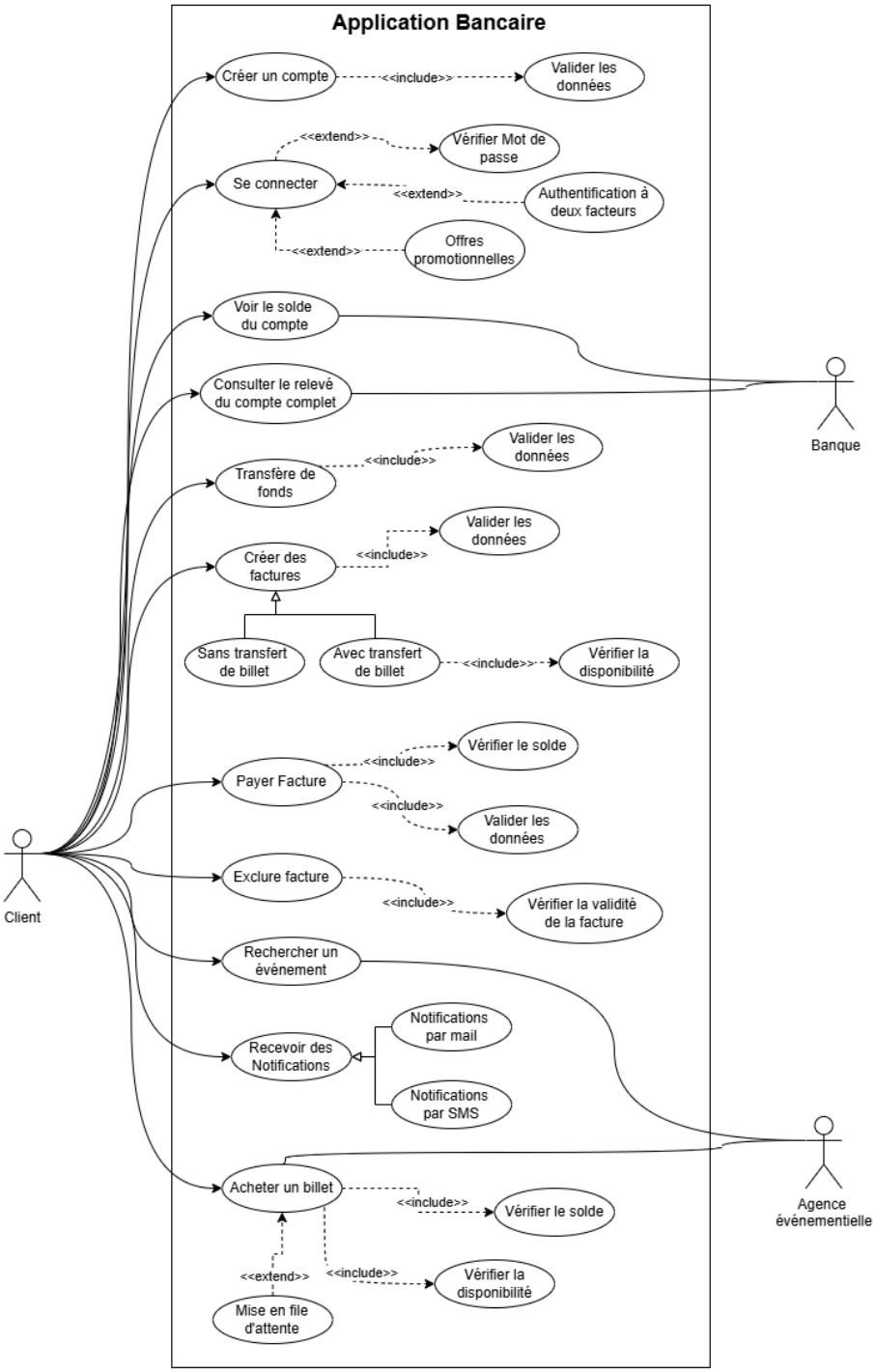

FIG. 1 – *Diagramme de cas d'utilisation*

2.3 Liste des exigences non fonctionnelles

Nous présentons dans cette section les exigences non fonctionnelles du système, chacune définie par un titre, une description et un niveau de priorité.

Requirement 1 : **Simplicité d'utilisation**
Priorité : **HIGH**
Description : Afin de garantir une meilleure expérience utilisateur, toutes les interfaces et fonctionnalités principales de l'application doivent être accessibles en un minimum d'étapes, avec un maximum de cinq clics ou interactions.
De plus, PrixBanque doit offrir une gestion 100% numérique, éliminant les démarches en agence et la paperasserie. Ainsi tous les processus doivent être digitalisés.

Requirement 2 : **La scabilité horizontale**
Priorité : **HIGH**
Description : Le système doit être capable de traiter au moins 2000 transactions par seconde et de supporter simultanément jusqu'à 1,5 million de clients.

Requirement 3 : **La disponibilité/résilience**
Priorité : **HIGH**
Description : Le système doit intégrer des mécanismes de reprise après panne pour garantir une disponibilité de 99,99%, soit un maximum de 50 minutes d'indisponibilité par an.
De plus, les mises à jour doivent pouvoir être effectuées sans interruption du service.

Requirement 4 : **L'évolutivité**
Priorité : **MEDIUM**
Description : Le système doit pouvoir s'adapter à l'ajout de nouvelles fonctionnalités sans modifier son architecture globale.

Requirement 5 : **La fiabilité**
Priorité : **MEDIUM**
Description : Le système doit garantir une haute tolérance aux fautes, avec un temps moyen entre pannes (MTBF) élevé.
Il doit détecter les erreurs rapidement, en proposant des solutions de résolution ou en redirigeant l'utilisateur vers le support technique.

Requirement 6 : **Sécurité**
Priorité : **MEDIUM**
Description : Le système doit utiliser le chiffrement des données ou bien encore une authentification multi-facteurs pour garantir une protection des données de l'utilisateur et des transactions financières.

3 Projet d'architecture

3.1 Vue logique 1 : Diagramme de contexte

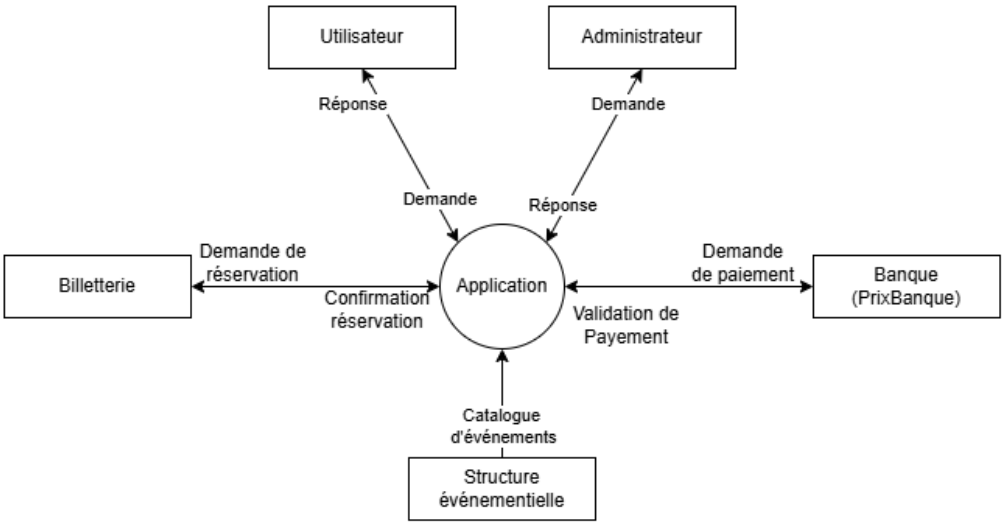

FIG. 2 – *Diagramme de contexte*

Le diagramme de contexte (fig.2) présente les acteurs externes interagissant avec l'application PrixBanque. Dans notre projet, ces acteurs incluent :

- Utilisateur : L'utilisateur utilise l'application pour envoyer des requêtes, auxquelles l'application répond.
- Administrateur : L'administrateur répond aux demandes de l'application, en gérant le support client, les erreurs et les problèmes techniques.
- Billetterie : La billetterie gère le processus de réservation des billets pour les spectacles.
- Structure événementielle : Elle est responsable de la gestion du catalogue des événements, proposant des événements à la vente et mettant à jour leurs informations.
- Banque (PrixBanque) : Cette entité gère toutes les transactions bancaires et les facturations associées à PrixBanque.

3.2 Vue développement

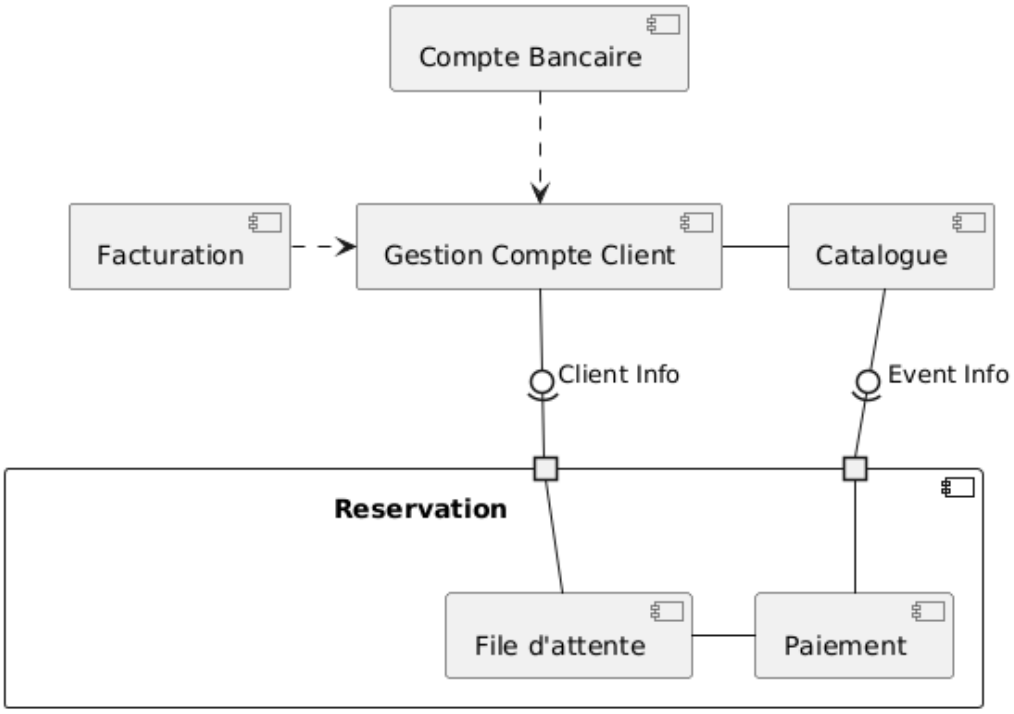

FIG. 3 – *Diagramme de composants*

Le diagramme de composants (fig.3) illustre les composants logiciels de l'application ainsi que les relations qui existent entre eux. Chaque composant fournit des services spécifiques et est conçu pour être autonome.

Les principaux composants de ce projet assurent la gestion des services de catalogue, de facturation, de transactions, de gestion de compte client, et de réservation.

Le composant *Catalogue* gère le catalogue des événements, leur mise à jour, ainsi qu'une recherche avancée pour trouver des spectacles en fonction des critères de l'utilisateur.

Le composant *Réservation* regroupe plusieurs sous-composants pour gérer le processus de réservation de spectacles : le service de paiement, ainsi qu'un composant *File d'attente* permettant de gérer le flux des utilisateurs en cas de forte demande.

Le composant *Gestion Compte client* permet aux utilisateurs de créer et gérer leur profil et de consulter les informations personnelles liées à leurs comptes. À partir de ce composant, les utilisateurs peuvent accéder aux systèmes de réservation, de facturation, de transactions et au catalogue des événements.

Le composant *Compte Bancaire* gère tous les services liés aux opérations bancaires, notamment les transactions, la consultation du solde et l'accès aux relevés de compte.
Le composant *Facturation* prend en charge la création, la suppression, le paiement de la facture.

3.3 Vue déploiement (physique)

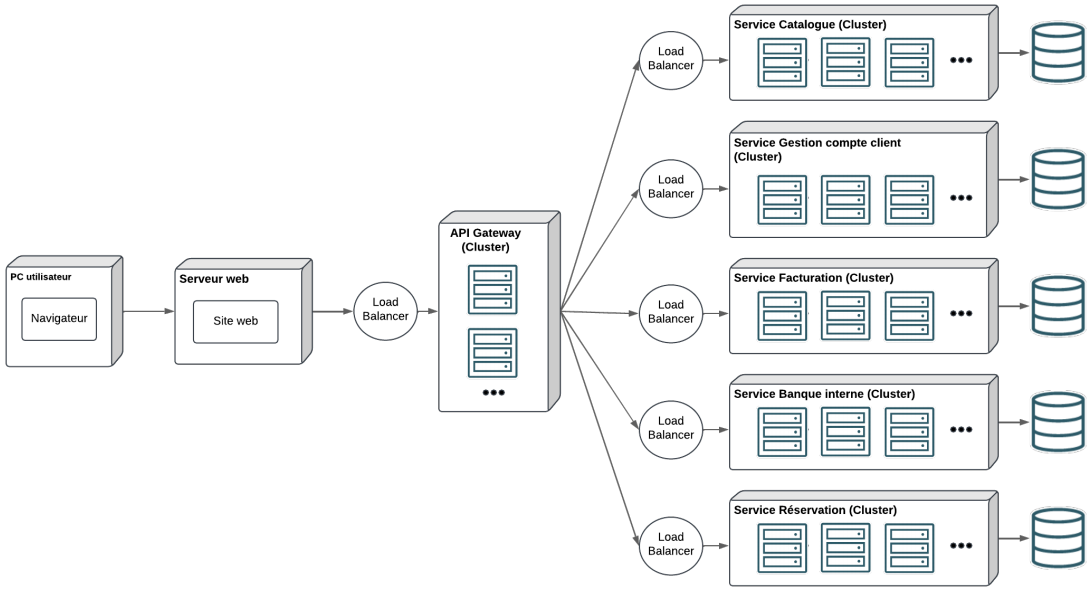

FIG. 4 – *Diagramme de déploiement*

Le diagramme de déploiement (fig.4) illustre la décomposition du système en micro-services. Chacun de ces micro-services correspondant à un composant de l'application, et possède sa propre base de données. L'interface web de l'application fait appel aux différents micro-services au travers d'une API Gateway, qui permet de réduire le couplage des micro-services et facilite la configuration, puisque le front-end envoie toutes ses requêtes au même serveur.

Ce diagramme illustre également le recours à des load balancers pour mettre en place une redondance des micro-services et l'API Gateway. En effet, le load balancer agit comme un intermédiaire qui réceptionne les reqûetes et les redistribue au différentes instances d'un service. Cela permet d'augmenter ou réduire le nombre d'instances d'un service à la volée en fonction de sa charge actuelle, et ainsi d'assurer sa disponbilité même lors de pics d'utilisation ou d'incident technique sur une des instances.

3.4 Vue logique 2 : Diagramme de classes

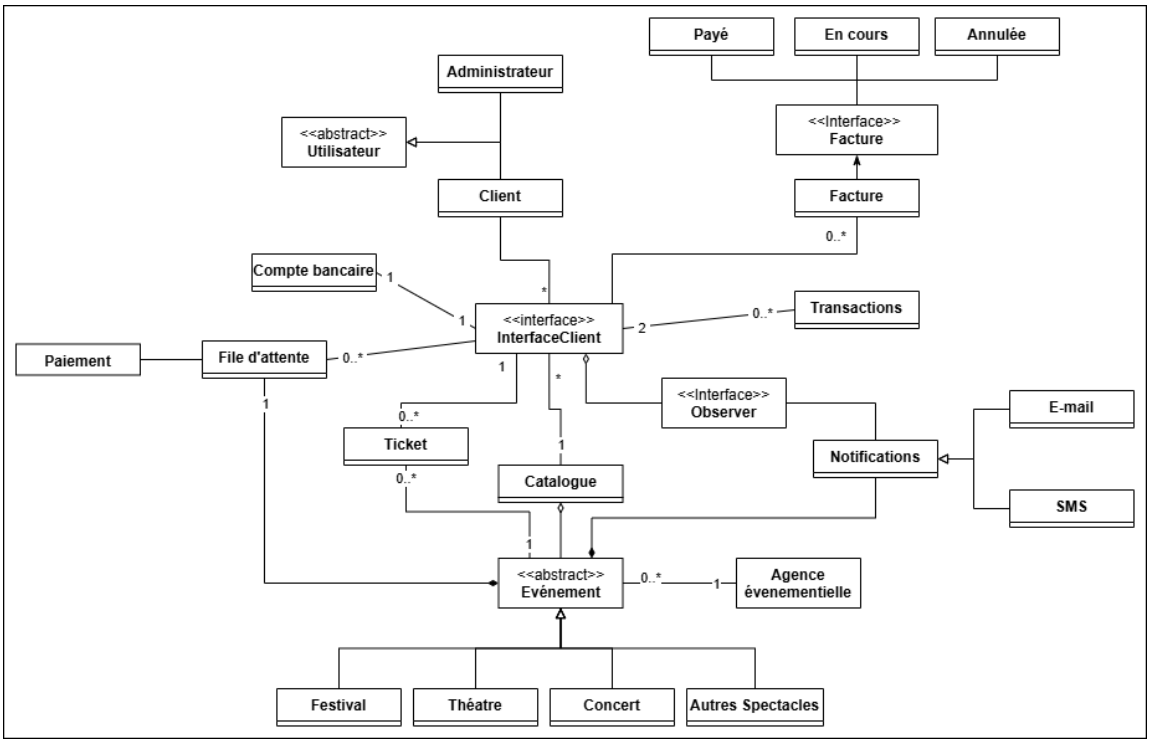

FIG. 5 – *Diagramme de classes*

Un diagramme de classes est une représentation de la structure d'un système, mettant en évidence les classes et leurs relations pour mieux comprendre l'organisation et les interactions entre les différents composants. Dans la figure 5, le diagramme de classes est présenté sans attributs ni méthodes afin de se concentrer uniquement sur la structure globale, sans entrer dans le détail des fonctionnalités. Nous allons ici détailler les rôles de chaque classe ainsi que leurs liens au sein du système.

La classe *InterfaceClient* est conçue pour servir de point central entre les utilisateurs et les services de l'application. Elle offre une interface à partir de laquelle les clients peuvent accéder aux fonctionnalités principales de l'application, simplifiant la communication entre l'utilisateur et le système backend.

Nous avons appliqué le patron de conception État (State) pour l'implémentation de la classe *Facture*, afin de gérer les différents comportements selon si la facture est payée, en cours ou annulée.

L'architecture repose également sur des classes abstraites et des interfaces pour assurer la flexibilité et l'extensibilité du système. Par exemple, *Utilisateur* est une classe abstraite qui sert de modèle de base pour les rôles de *Client* et *Administrateur*. De même, *Événement* permet de représenter différents types d'événements, tels que des concerts, festivals ou pièces de théâtre, offrant ainsi la

possibilité d'ajouter facilement d'autres types d'événements avec des caractéristiques spécifiques à l'avenir.

Pour gérer les paiements de tickets, l'application utilise une classe *Paiement* qui modélise les transactions financières liées à l'achat de billets de spectacles.

La classe *Catalogue* gère l'accès au catalogue des spectacles. Chaque événement est associé à une *File d'attente* pour gérer le flux des utilisateurs en cas de forte demande, garantissant une gestion fluide des accès aux événements populaires.

Enfin, le système de notifications utilise le patron de conception *Observer* pour informer les utilisateurs de toute mise à jour concernant les billets achetés. Les notifications sont envoyées par SMS ou par e-mail en fonction des préférences de l'utilisateur.

3.5 Définitions et justification des technologies choisies

Dans cette section, nous allons définir nos choix en ce qui concerne le style architectural, les cadriciels, le mécanisme de persistance et les patrons de conception.

Style Architectural
Pour l'application PrixBanque, nous avons choisi une architecture en **microservices**. Cela nous permettra de décomposer les fonctionnalités en services autonomes comme la gestion des utilisateurs, la réservation, le paiement et la facturation. Pour communiquer entre eux, les services utiliseront des API REST.

Cadriciels
Pour le développement du backend, nous avons choisi d'utiliser **Java Spring**. L'emploi de ce framework est particulièrement pertinent ici pour son utilité dans les architectures en microservices. De plus, il offre une grande flexibilité et permet de construire des applications web rapides, réactives, et sécurisées.
En ce qui concerne le développement du frontend, nous utiliserons **React**, qui est très réputé pour offrir une expérience utilisateur fluide et intuitive.

Mécanisme de persistance
Nous souhaitons intégrer de nombreuses fonctionnalités dans le projet et nous serons donc amenés à traiter un très grand nombre de données. Nous opterons donc pour une base de données relationnelle **MySQL** afin de structurer efficacement les informations.

Patrons de conception
Pour répondre aux défis de conception et rendre l'application plus modulaire et réutilisable, nous utilisons plusieurs patrons de conception :

- Observer : Pour gérer le système de notifications.
- Repository : Facilite l'accès et l'utilisation des données dans la base de données.
- État : Gère le cycle de vie des factures (payée, impayée, annulée).

- Factory : Créer des objets variés mais du même type. Utile pour les différents évènements et les différents modes de paiement

4 Exécution

4.1 Organigramme des tâches du projet

L'organigramme des tâches du projet (fig.6) offre une représentation structurée des différentes étapes et activités nécessaires au développement de l'application bancaire PrixBanque.

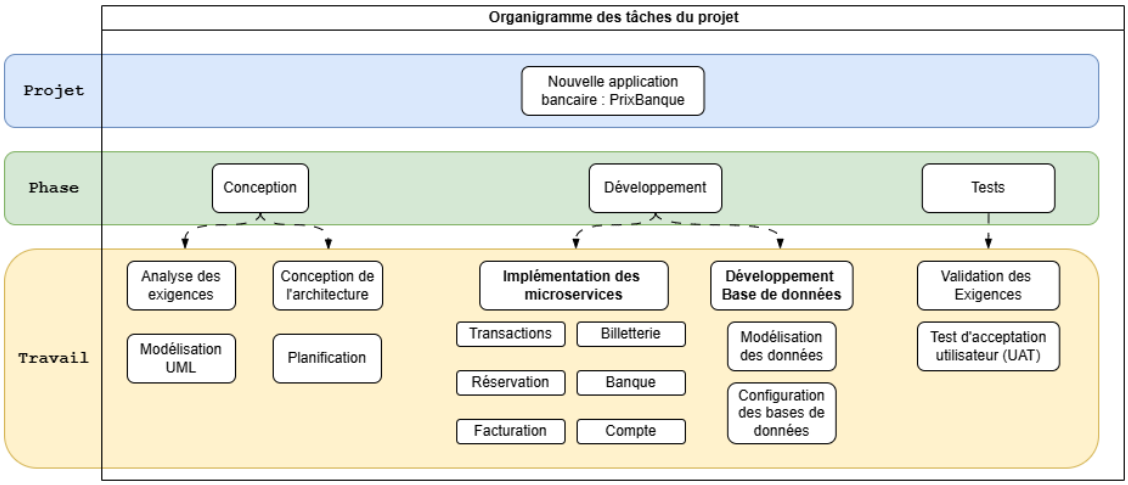

FIG. 6 – *Organigramme des tâches du projet*

4.2 Feuille de route

Un diagramme de Gantt est un outil de planification de projet qui permet de visualiser l'ensemble des tâches à accomplir sur une échelle temporelle. Nous avons établi un tel diagramme dans la figure 7 afin de présenter la planification du projet de développement de l'application bancaire PrixBanque.

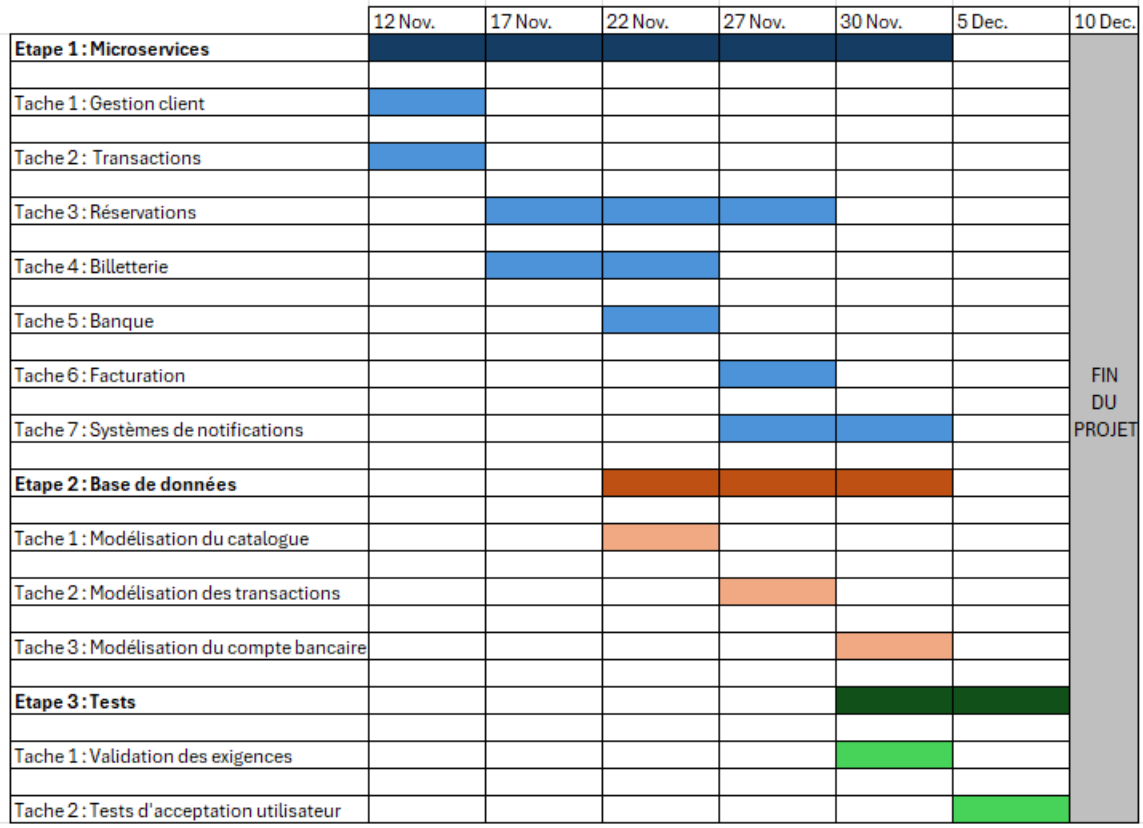

FIG. 7 – *Diagramme de Gantt*