Documentation technique

Site de gestion des factures pour une école privée

TA75 – Projet tutoré

FISA Informatique - UTBM

Année académique : 2024 – 2025

Fayçal BIJJI

Antoine ROLLIN

Maxence LELANDAIS

Table des matières

Glossaire

Spring Boot : Un framework Java basé sur Spring qui simplifie le développement d'applications en fournissant une configuration automatique et des conventions prêtes à l'emploi. Il permet de créer rapidement des applications web et microservices.

Framework : Un ensemble d'outils, de bibliothèques et de bonnes pratiques facilitant le développement logiciel en imposant une structure et en fournissant des fonctionnalités préconstruites. Exemples : Spring, Angular, Django.

Modèle de donnée : Une représentation abstraite de la structure des données utilisées dans une application, incluant les relations entre les entités et les règles de gestion associées. Il peut être logique, conceptuel ou physique.

Procédure de base de données : Un ensemble d'instructions SQL stockées dans une base de données et exécutées comme une unité. Elle permet d'automatiser des opérations complexes, d'améliorer les performances et de sécuriser l'accès aux données.

REST API (Representational State Transfer - Application Programming Interface) : Une interface permettant la communication entre systèmes via le protocole HTTP en respectant les principes REST. Elle utilise des méthodes HTTP standard (GET, POST, PUT, DELETE) pour manipuler des ressources représentées généralement en JSON ou XML. Elle est stateless, ce qui signifie que chaque requête est indépendante et ne conserve pas d'état côté serveur.

DTO (Data Transfer Object) : Un objet utilisé pour transférer des données entre différentes couches d'une application sans exposer directement les entités du modèle de données. Il permet d'optimiser la communication et d'améliorer l'encapsulation.

Présentation du sujet

Le sujet

Objectif principal

Créer un outil de gestion et de facturation destiné à une école privée pour :

- Générer des factures personnalisées pour chaque famille.
- Gérer des réductions spécifiques (notamment pour les fratries).
- Facturer divers services annexes : heures d'études, cantines, sorties scolaires, etc.
- Suivre les paiements et automatiser les rappels pour les factures impayées.

Fonctionnalités principales

Gestion des familles

- Création de profils familiaux avec la liste des enfants inscrits.
- Configuration des niveaux scolaires et des services associés (classe, cantine, études, activités extrascolaires).
- Application automatique des réductions pour les fratries (ex. : -10% pour le deuxième enfant, - 20% pour le troisième, etc).

Facturation personnalisée

- Génération automatique des factures mensuelles/annuelles basées sur les services sélectionnés :
    - Frais de scolarité.
    - Heures d'étude.
    - Repas à la cantine.
    - Sorties scolaires ou activités extra-scolaires.
    - Gestion de plans de paiement (ex : mensualisation).
    - Calcul des réductions et application de remises spécifiques.

Paiements

- Suivi des paiements : payé, en retard, partiellement payé.
- Génération de rappels automatiques pour les factures impayées par e-mail/SMS.
- Intégration avec des plateformes de paiement en ligne (Stripe, PayPal, ou autres).

Rapports financiers

- Tableau de bord pour l'administration de l'école montrant :
    - Le total facturé par mois.
    - Le suivi des paiements.
    - Les soldes restants dus par les familles.
    - Génération de rapports annuels pour audit ou reporting interne.

Interface utilisateur

- Portail pour les administrateurs de l'école :
    - Gestion des familles, des enfants et des services.
    - Création, modification et suppression des factures.
- Portail parents :
    - Accès aux factures en ligne.
    - Historique des paiements.
    - Possibilité de régler directement en ligne.

Aspects techniques

- Frontend : Angular, PHP
- Backend : Python, Java
- Intégration : API REST, Stripe (paiement en ligne)

Sécurité

- HTTPS
- Authentification multi-facteurs pour les administrateurs et parents.
- Chiffrement des données sensibles (ex. : coordonnées bancaires).

Scénarios

Génération des factures

- L'administrateur entre les informations sur les enfants, les services choisis et les réductions.
- Une facture est automatiquement générée et envoyée aux parents par e-mail.

Suivi des paiements

- Un parent règle une facture directement via l'application mobile.
- L'administrateur voit immédiatement que le paiement est effectué sur le tableau de bord.

Ajout d'un service

- En milieu d'année, un enfant s'inscrit à des cours de musique. L'administrateur ajoute ce service et génère une facture mise à jour.

Livrables

- Une application web fonctionnelle.
- Applications mobiles pour iOS et Android.
- Documentation technique (architecture, API, documentation technique).

Les solutions retenues

Fonctionnalités principales

Gestion des familles

- Création de profils familiaux avec la liste des enfants inscrits.
- Configuration des niveaux scolaires et des services associés (classe, cantine, études, activités extrascolaires).

Les réductions sont sélectionnées par l'administrateur quand il créait une facture. L'application automatique des réductions se fera dans une prochaine version.

Facturation personnalisée

- Génération par l'administrateur des factures mensuelles/annuelles basées sur les services sélectionnés :
  - Frais de scolarité.
  - Heures d'étude.
  - Repas à la cantine.
  - Sorties scolaires ou activités extra-scolaires.
  - Gestion de plans de paiement (ex : mensualisation).
  - Calcul des réductions et application de remises spécifiques.

Paiements

Non inclus dans cette version.

Rapports financiers

Non inclus dans cette version.

Interface utilisateur

- Portail pour les administrateurs de l'école :
  - Gestion des familles, des enfants et des services.
  - Création et archivage des factures.
- Portail parents :
  - Historique des paiements.

Le règlement et l'accès des factures en ligne n'ai pas géré dans cette version.

Aspects techniques

- Frontend : Angular
- Backend : Java
- Intégration : API REST

Sécurité

Cette version du projet est hébergée en local et fonctionne qu'en local. Le chiffrement de la connexion, des données et la double authentification ne seront pas implémentés dans cette version.

Scénarios

Génération des factures

- L'administrateur entre les informations sur les enfants, les services choisis et les réductions.

Pour réduire la charge de développement, les factures seront créées. L'envoi d'e-mail ne se fera pas. Seuls les administrateurs gèrent les factures. Les utilisateurs pourront visualiser leurs factures sur le site web.

Suivi des paiements

La gestion des paiements ne sera pas incluse dans cette version.

Ajout d'un service

- En milieu d'année, un enfant s'inscrit à des cours de musique. L'administrateur ajoute ce service et génère une facture mise à jour.

Livrables

- Une application web fonctionnelle.
- Documentation technique (architecture, API, documentation technique).

L'applications mobiles pour iOS et Android ne sera pas développé par choix organisationnel.

Réalisation

Les choix techniques

Nous utilisons une base de données pour stocker les informations de l'application web (comptes utilisateurs, les factures, ...).

La base de données choisi est MySQL. Son inferface simple, sa simplicité de gestion et notre expérience d'utilisation de ce logiciel ont été les principaux critères de sélection.

Pour utiliser cette base de données, nous avons développé un backend Java qui contient un service permettant au backend d'interagir avec la base de données.

L'interface de programmation RestAPI, qui nous permet de configurer les end-points d'une API java, est utilisé pour communiquer entre le backend et le frontend de note application web.

Le framework SpringBoot, nous permet de simplifier l'architecture de notre backend en automatisant des tâches.

L'orchestrateur Maven est utilisé pour gérer les librairies utilisées par le backend et de générer facilement des .jar exécutables.

Le frontend est développé en type script avec les bases du html/css. Le framework Angular nous permet de rendre plus dynamique notre site web et améliore l'architecture du frontend.

La communication avec l'API se fait simplement avec des requêtes http.

Ces choix techniques ont été fait en fonction des connaissances de et des compétences de notre équipe.

Architecture du projet

Base de données

(Annexe 1)

Notre base de données est découpée en 3 parties :

- Gestion des utilisateurs
- Gestion des familles
- Gestion des factures

Nous utilisons des procédures pour faciliter les échanges avec la base de données et le backend. Pour la modification des données, le backend passe par les procédures. Pour la récupération des données et l'archivage les requêtes sont directement effectuées par le backend.

Gestion des comptes utilisateur

- Table utilisateur : Stocke les informations des utilisateurs (identifiants, mots de passe, rôle d'administrateur).
- Chaque utilisateur a un identifiant unique (id_user), un login et un mot de passe.
- Un utilisateur peut être administrateur (admin = TRUE).

Gestion des familles

- Table famille : Chaque famille est associée à un utilisateur (id_user).
- Table enfant : Contient les informations des enfants (nom, prénom, âge).
- Table liste_enfant : Relie les enfants aux familles (id_enfant, id_famille).
- Table statut_parent : Définit les statuts des parents (ex. père, mère, tuteur).
- Table parent : Stocke les informations des parents, avec un statut défini (id_statut_parent).
- Table liste_parent : Relie les parents aux familles (id_parent, id_famille).

Gestion des réductions

- Table reduction : Définit des réductions sous forme de montant ou de pourcentage.
- Table liste_reduction_enfant : Associe des réductions aux enfants.
- Table liste_reduction_famille : Associe des réductions aux familles.

Gestion des factures

- Table etat_paiement : Contient les statuts des paiements (ex. payé, en attente, en retard).
- Table periode : Définit les périodes de facturation et les réductions associées.
- Table facture : Stocke les informations des factures (description, créancier, débiteur, échéance, état de paiement, etc.).
- Table historique_facture : Relie une facture à une famille.

Gestion des frais

- Table type_frais : Catégories de frais (ex. frais de scolarité, transport).
- Table frais : Stocke les frais avec un montant, une description et une réduction éventuelle.
- Table liste_frais_facture : Associe les frais aux factures.

Gestion des paiements

- Table type_paiement : Définit les types de paiement (ex. carte, virement).
- Table paiement : Stocke les paiements réalisés.
- Table liste_paiement_facture : Associe un paiement à une facture.

Backend et frontend

(Annexe 2)

FRONTEND (Interface utilisateur)

Le frontend est la partie visible de l'application, développée avec HTML, CSS et TypeScript (TS). Il est organisé en plusieurs couches :

Routing

- Gère la navigation entre les pages de l'application.

Pages

- Contient plusieurs vues :
  - Connexion : Interface d'authentification des utilisateurs.
  - Accueil : Page principale après connexion.
  - Listes des utilisateurs, factures, familles, services et réductions :
    - Permet de visualiser, créer, modifier et archiver des données.

Models (Modèles)

- Représente les entités utilisées dans le frontend :
    - `Utilisateur`, `Facture`, `Paiement`, `Parent`, `Enfant`, `Frais`, `Réduction`, etc.

Services

- Gère la communication avec l'API backend.
- Exemple : `FactureService` permet d'envoyer/récupérer les factures via des DTO (Data Transfer Objects) via la structure des modèles.

Les DTO normalement défini en plus des modèles contiennent les mêmes informations que les modèles. Ainsi, par simplification, les données des modèles sont directement transmises.

BACKEND (Traitement et logique métier)

Le backend est divisé en plusieurs couches :

Models (Modèles)

- Définit les entités utilisées dans le backend.
- Contient des classes représentant les mêmes modèles que le frontend (`Utilisateur`, `Facture`, `Paiement`, etc.).

Rest API

- Contient des contrôleurs qui exposent des endpoints pour :
    - Gérer les utilisateurs, factures, paiements, familles, réductions, etc.
- Chaque entité a son propre contrôleur.

Services

- BDService : Interface entre le backend et la base de données.

API

- Communique avec le frontend en envoyant des DTOs utilisant la structure des modèles.

BASE DE DONNÉES

La base de données stocke toutes les informations et est organisée comme suit :

Tables et relations

- Contient toutes les entités définies dans le script SQL fourni précédemment (`Utilisateur`, `Facture`, `Paiement`, etc.).

Procédures stockées

- Permet d'effectuer des opérations sur les données (création, modification, archivage).
- Exécute des requêtes `SELECT` et `ARCHIVE` pour récupérer ou désactiver des enregistrements.

Les qualités et les défauts

Les qualités

L'application est structurée en plusieurs couches : frontend, backend, base de données et API REST. Cette architecture modulaire facilite l'évolution du projet et sa maintenance.

Les technologies utilisées (Angular, Spring Boot, REST API) sont courantes dans l'industrie et offrent fiabilité, performance et compatibilité avec d'autres services.

L'utilisation de procédures stockées en base de données optimise les traitements en réduisant la charge du backend et en rapprochant les calculs des données.

L'application est bien organisée grâce à des services dédiés et une séparation des rôles claire, ce qui améliore la maintenabilité et les performances.

Les défauts

- Pas de gestion des paiements : L'application ne permet pas d'automatiser les paiements, ce qui complique la gestion des factures.
- Certaines fonctionnalités manquent :
    - Gestion des emails : Impossible d'envoyer automatiquement les factures et les rappels de paiement.
    - Sécurité limitée : L'authentification et la gestion des accès ne respectent pas encore les bonnes pratiques (chiffrement des mots de passe, sécurisation des connexions).
    - Interface basique : Fonctionnelle mais peu ergonomique, ce qui peut être amélioré avec un design plus intuitif.
    - Gestion des factures manuelle : Tout est fait à la main, alors qu'une automatisation permettrait de gagner du temps.

Axes d'amélioration

Plusieurs améliorations peuvent être apportées :

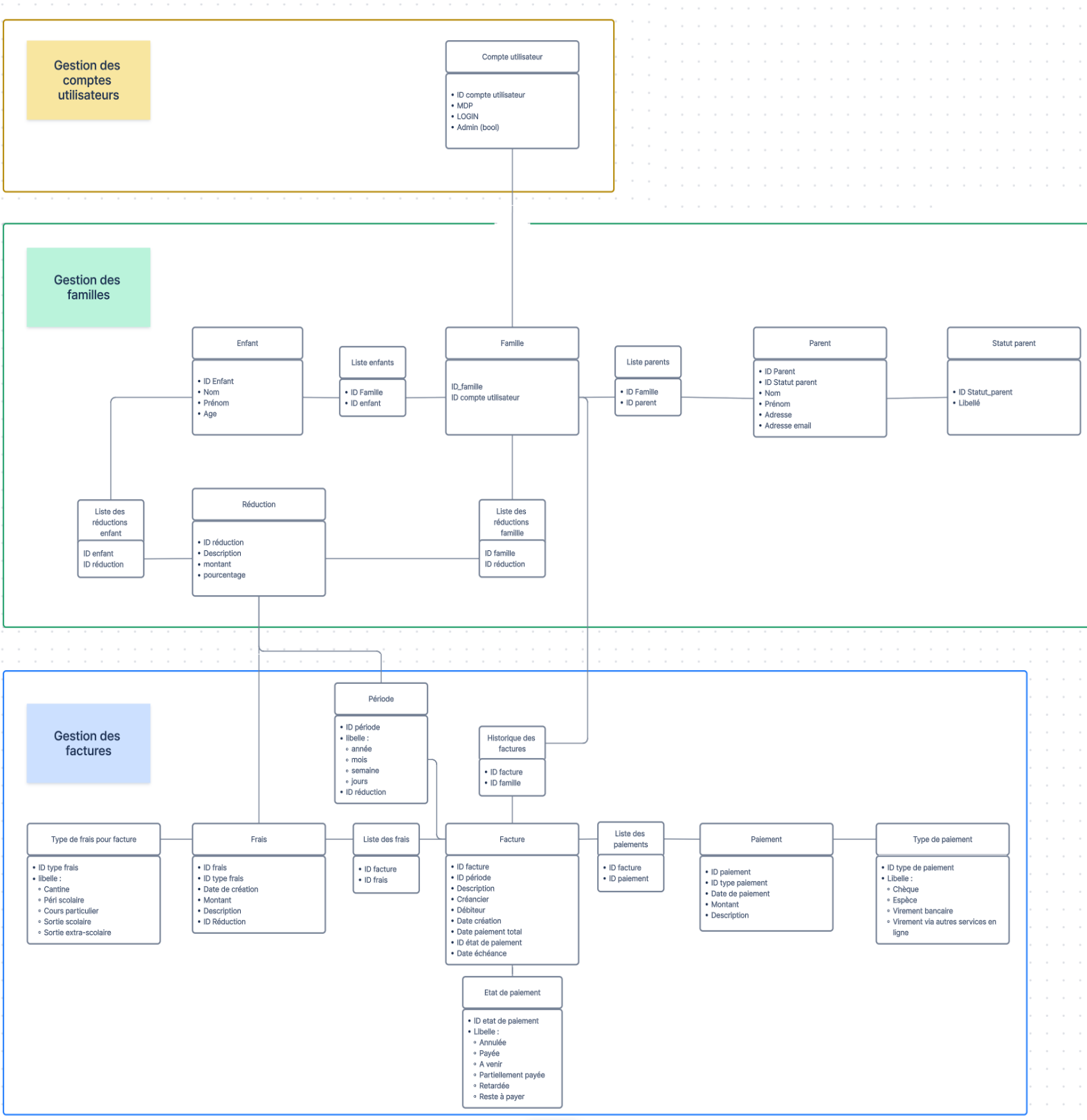
- Ajout d’une solution de paiement (Stripe, PayPal) pour automatiser les transactions et faciliter le suivi des paiements.
- Amélioration du design avec Bootstrap ou TailwindCSS pour une interface plus agréable et intuitive.
- Sécurisation des données avec chiffrement des mots de passe (bcrypt, Argon2), authentification multi-facteurs (2FA) et connexion HTTPS.
- Déploiement en ligne sur un serveur cloud (AWS, Azure, VPS) pour un accès distant et une meilleure fiabilité.
- Automatisation des factures :
    - Génération et envoi automatiques via des tâches planifiées.
    - Envoi de rappels de paiement par email/SMS.
    - Archivage des factures pour alléger la base de données.

Conclusion

L’application est fonctionnelle et repose sur une architecture moderne et solide. Elle peut évoluer facilement, mais plusieurs points doivent être améliorés avant un déploiement en production, notamment la gestion des paiements, la sécurité, et l’automatisation.

En intégrant ces évolutions, le projet deviendrait plus efficace, sécurisé et facile à utiliser pour les administrateurs et les utilisateurs.

Annexe 1 : Architecture de la base de données



Annexe 2 : Architecture de l'application