William Leroux

Nicolas Morin

Samuel Oliveira Martel

GR : 01

**GymExpress – Documentation technique**

**Remis le 14 mars 2025**

Travail présenté à :

Sylvain Jr. Rivard

Département d’informatique

CEGEP Régional de Lanaudière à Joliette

Table des matières

[Documentation Technique 4](#_Toc192502524)

[1. Architecture du système 4](#_Toc192502525)

[1.1 Vue d'ensemble 4](#_Toc192502526)

[1.2 Frontend (Interface utilisateur) 4](#_Toc192502527)

[1.3 Logique métier 4](#_Toc192502528)

[1.4 Base de données 4](#_Toc192502529)

[1.7 Distribution et mise à jour 5](#_Toc192502530)

[2. Fonctionnalités principales 6](#_Toc192502531)

[Communes 6](#_Toc192502532)

[2.1 Authentification et gestion des utilisateurs 6](#_Toc192502533)

[Administrateur 6](#_Toc192502534)

[2.1 Gestion des abonnements 6](#_Toc192502535)

[2.2 Module financier 6](#_Toc192502536)

[2.3 Gestion des employés 6](#_Toc192502537)

[Entraîneur 6](#_Toc192502538)

[2.1 Gestion des abonnements 6](#_Toc192502539)

[2.2 Module financier 6](#_Toc192502540)

[2.3 Gestion des employés 7](#_Toc192502541)

[Clients 7](#_Toc192502542)

[2.1 Gestion des abonnements 7](#_Toc192502543)

[2.2 Module financier 7](#_Toc192502544)

[2.3 Gestion des employés 7](#_Toc192502545)

[3. API et interfaces 7](#_Toc192502546)

[3.1 Vue d’assemble API 7](#_Toc192502547)

[4. Sécurité 9](#_Toc192502548)

[4.1 Authentification et autorisation 9](#_Toc192502549)

[4.2 Protection des données 9](#_Toc192502550)

[4.3 Améliorations de sécurité recommandées 10](#_Toc192502551)

[5. Déploiement (Contexte académique) 11](#_Toc192502552)

[5.1 Configuration requise pour le déploiement 11](#_Toc192502553)

[5.2 Processus de déploiement théorique 11](#_Toc192502554)

[5.3 Procédures de maintenance 12](#_Toc192502555)

[5.4 Considérations pour un déploiement réel futur 13](#_Toc192502556)

[6. Guide de développement 14](#_Toc192502557)

[6.1 Configuration de l'environnement 14](#_Toc192502558)

[6.2 Standards de code 14](#_Toc192502559)

[6.3 Tests 14](#_Toc192502560)

[8. Dépannage et résolution de problèmes 15](#_Toc192502561)

[Problèmes connus 15](#_Toc192502562)

[API non fonctionnelle 15](#_Toc192502563)

[9. Feuille de route 16](#_Toc192502564)

[Prochaines évolutions majeures 16](#_Toc192502565)

[Améliorations techniques prévues 16](#_Toc192502566)

# Documentation Technique

## 1. Architecture du système

### 1.1 Vue d'ensemble

* **Type d'application**: Application de bureau native macOS
* **Architecture**: Application avec interface graphique native et base de données locale
* **Plateforme cible**: macOS

### 1.2 Frontend (Interface utilisateur)

* **Langage**: Swift 6.0.2
* **Framework UI**: SwiftUI
* **Structure de l'interface**:
  + Système de navigation par barre latérale
  + Écrans principaux: Connexion, Tableau de bord, Entraîneur, Employés, Client.
  + Composants personnalisés: Graphiques, tableaux de données, formulaires, progressions.
* **Gestion d'état**:
  + Pattern MVC (Model-View-Controller)
* **Ressources graphiques**:
  + Icons et assets intégrés au bundle de l'application
  + Support des thèmes clair et sombre de macOS

### 1.3 Logique métier

* **Langage**: Swift 6.0.2
* **Organisation du code**:
  + Modèles: Rendez-vous, Évènement de calendrier, Jour, Exercice, Fréquence, Abonnement, Mois, Objectif, Utilisateur, Semaine, Entraînement.
* **Gestion des erreurs**:
  + Système de journalisation des erreurs dans la documentation

### 1.4 Base de données

* **Type**: SQLite 3 (base de données relationnelle)
* **Emplacement du fichier DB**: ~/Library/Containers/gymExpress.GymExpress/Data/Documents/gymexpress.sqlite

### 1.7 Distribution et mise à jour

* **Méthode de distribution**:
  + Mac App Store
* **Mise à jour**:
  + À travers le Mac App Store

## 2. Fonctionnalités principales

## Communes

### 2.1 Authentification et gestion des utilisateurs

* Système de connexion et d'inscription
* Gestion des rôles (Client, Entraîneur ou administrateur)
* Se souvenir de moi

## Administrateur

### 2.1 Gestion des abonnements

* Types d'abonnements disponibles
* Affichage des prix
* Gestion des abonnements

### 2.2 Module financier

* Rapports financiers et analytiques

### 2.3 Gestion des employés

* Structure des données employés
* Suivi des salaires
* Gestion des employés

## Entraîneur

### 2.1 Gestion des abonnements

* Types d'abonnements disponibles
* Affichage des prix
* Gestion des abonnements

### 2.2 Module financier

* Traitement des paiements
* Génération de factures
* Rapports financiers et analytiques
* Intégration avec des systèmes de paiement tiers

### 2.3 Gestion des employés

* Structure des données employés
* Système de planification des horaires
* Suivi des performances
* Gestion des accès

## Clients

### 2.1 Gestion des abonnements

* Types d'abonnements disponibles
* Affichage des prix
* Gestion des abonnements

### 2.2 Module financier

* Traitement des paiements
* Génération de factures
* Rapports financiers et analytiques
* Intégration avec des systèmes de paiement tiers

### 2.3 Gestion des employés

* Structure des données employés
* Système de planification des horaires
* Suivi des performances
* Gestion des accès

## 3. API et interfaces

### 3.1 Vue d’assemble API

* **API utilisée**: ExerciseDB API (<https://exercisedb-api.vercel.app/docs>)
* **Description**: API complète fournissant des informations détaillées sur des exercices d’entraînement divers.

## 4. Sécurité

Cette section détaille les mesures de sécurité actuellement implémentées dans l'application GymExpress, ainsi que les recommandations pour les améliorations futures. Bien que certains aspects de sécurité soient limités dans cette version académique, les principes fondamentaux sont documentés.

### 4.1 Authentification et autorisation

#### 4.1.1 Système d'authentification

* **Méthode implémentée**: Authentification basique par identifiant/mot de passe
* **Stockage des mots de passe**: En clair dans la base de données (À ne pas faire\*\*\*)
* **Écran de connexion**: Validation des champs obligatoires
* **Sessions**: Sessions locales gérées via SQLite sans expiration automatique

#### 4.1.2 Validation des données de formulaire

* **Validation des champs texte**:
  + Vérification des champs requis avec longueur minimale

#### 4.1.3 Gestion des rôles (simplifiée)

* **Rôles de base**:
  + Administrateur: Accès complet à toutes les fonctionnalités
  + Employé: Accès restreint à certaines fonctionnalités
  + Client : Accès restreint à certaines fonctionnalités
* **Vérification des autorisations**: Contrôle simple au niveau des contrôleurs

### 4.2 Protection des données

#### 4.2.1 Données sensibles

* **État actuel**: Stockage simple dans SQLite sans chiffrement additionnel
* **Recommandation future**: Implémentation du chiffrement SQLite
* **Données critiques identifiées**:
  + Informations personnelles des clients
  + Détails financiers
  + Identifiants des employés

#### 4.2.2 Rétention des données

* **État actuel**: Pas de politique automatisée de suppression des données
* **Recommandation**: Implémentation d'une purge automatique des données inactives après 2 ans

### 4.3 Améliorations de sécurité recommandées

#### 4.3.1 Améliorations à court terme

* Améliorer la robustesse de la validation des données avec des règles plus strictes
* Implémenter le hachage des mots de passe avec bcrypt ou Argon2
* Ajouter l'authentification à deux facteurs pour les comptes administrateurs

#### 4.3.2 Plan de sécurité à moyen terme

* Mise en œuvre du chiffrement de la base de données SQLite
* Audit de sécurité complet du code source
* Développement de tests de sécurité automatisés

#### 4.3.3 Bonnes pratiques déjà appliquées

* Paramétrage des requêtes SQL pour éviter les injections
* Validation côté client et serveur des données de formulaire
* Journalisation des événements importants pour la traçabilité

## 5. Déploiement (Contexte académique)

Cette section décrit l'approche théorique de déploiement pour l'application GymExpress. Bien que ce projet soit développé dans un cadre académique et ne soit pas actuellement déployé en production, cette documentation présente les meilleures pratiques qui seraient appliquées dans un environnement réel.

### 5.1 Configuration requise pour le déploiement

#### 5.1.1 Spécifications matérielles théoriques

* **Pour usage démonstratif/académique**:
  + Tout Mac moderne capable d'exécuter macOS 11 Big Sur ou ultérieur
  + Minimum 8 GB RAM
  + 256 GB d'espace disque SSD
  + Processeur Intel Core i5 ou Apple Silicon M1 (ou plus récent)
* **Pour un déploiement en production théorique**:
  + Mac mini ou MacBook dédié comme serveur de démonstration
  + Solution de backup pour la base de données SQLite
  + Connexion internet stable pour l'accès à l'API ExerciseDB

#### 5.1.2 Prérequis logiciels

* macOS 11
* Xcode 13+ (pour la compilation)
* Gestionnaire de packages Swift
* SQLite 3 (inclus dans macOS)
* Dépendances via Swift Package Manager déclarées dans le fichier Package.swift

#### 5.1.3 Distribution de l'application

* L'application est distribuée sous forme de fichier .app compilé
* Signature avec un certificat de développeur Apple (simulé pour le projet d'école)
* Documentation d'installation fournie aux utilisateurs

### 5.2 Processus de déploiement théorique

#### 5.2.1 Environnements de développement

* **Développement**: Machines locales des développeurs
* **Test**: Version de l'application utilisée pour les démonstrations en classe
* **"Production"**: Version finale soumise pour évaluation académique

#### 5.2.3 Pipeline de livraison (simulé)

* **Phase 1**: Build et tests unitaires
* **Phase 2**: Tests manuels et validation par l'équipe
* **Phase 3**: Soumission pour revue de code
* **Phase 4**: Création de la build finale avec un numéro de version incrémenté

#### 5.2.4 Documentation de release

* **Notes de version** décrivant:
  + Nouvelles fonctionnalités
  + Corrections de bugs
  + Changements connus
  + Instructions de mise à jour

#### 5.2.5 Déploiement simulé

Pour simuler un déploiement en environnement professionnel, l'application est:

* Exportée au format .app signé (simulation avec un certificat de développement)
* Archivée avec les notes de version
* Mise à disposition via un dépôt partagé pour l'équipe/évaluateurs

### 5.3 Procédures de maintenance

#### 5.3.1 Gestion des mises à jour

* Mises à jour incrémentielles via des pull requests sur GitHub/GitLab
* Tests de régression avant chaque nouvelle version
* Documentation des changements dans le CHANGELOG.md

#### 5.3.2 Sauvegarde des données

* Procédure théorique de sauvegarde SQLite:
  + Exports réguliers de la base de données
  + Script de backup simulé (non implémenté)
  + Stratégie documentée pour préserver l'intégrité des données

#### 5.3.3 Plan de reprise en cas d'incident (théorique)

* Procédure documentée pour restaurer l'application à un état fonctionnel
* Instructions pour récupérer des sauvegardes de la base de données
* Rollback à une version précédente stable si nécessaire

### 5.4 Considérations pour un déploiement réel futur

Si ce projet devait être déployé en production après la phase académique:

* Migration potentielle vers une architecture client-serveur
* Implémentation d'un backend robuste avec API RESTful
* Mise en place d'une base de données serveur (PostgreSQL, MySQL)
* Configuration d'un système de déploiement continu
* Mise en œuvre d'une surveillance et d'alertes

## 6. Guide de développement

### 6.1 Configuration de l'environnement

* Prérequis logiciels
* Variables d'environnement
* Instructions d'installation

### 6.2 Standards de code

* Conventions de nommage
* Pratiques de formatage
* Processus de revue de code

### 6.3 Tests

* Stratégie de test (unitaires, intégration, E2E)
* Outils de test utilisés
* Couverture de code ciblée

## 8. Dépannage et résolution de problèmes

## Problèmes connus

### API non fonctionnelle

L’API ne répond pas aux requêtes ce qui empêche l’application de récupérer et d’afficher les données attendues. Nous ne pouvons pas avoir les images et les différents d’exercices.

**Solution**

L’idéal serait d’utiliser une API plus fiable. Cependant, en raison des contraintes de délais et des coûts associés à l’utilisation d’une API de qualité, cette option n’est pas envisageable dans le cadre du projet scolaire.

**Finaliser la création d’un entrainement**

L’entraîneur ne peut pas finaliser un entraînement, car la création des exercices est impossible en raison du dysfonctionnement de l’API.

**Solution :**

* Vérifier si une autre source de données temporaire peut être utilisée pour contourner le problème. (Solution la plus réaliste)
* Implémenter une fonctionnalité permettant d’enregistrer des exercices localement jusqu’à ce que l’API soit à nouveau fonctionnelle.
* Contacter le fournisseur de l’API pour obtenir une assistance ou vérifier si une correction est prévue.

## 9. Feuille de route

## Prochaines évolutions majeures

### Entraîneur

Nous avons remarqué que l’entraîneur peut voir tous les clients et non seulement les siens. Une mise à jour serait nécessaire, car si le nombre de clients devient trop important, il sera difficile pour l’entraîneur de retrouver et modifier uniquement ses propres clients.

* **Gestion des permissions insuffisante** : L’entraîneur a accès à tous les clients au lieu d’avoir une vue restreinte à ses propres clients.
* **Ergonomie de l’interface** : L’affichage actuel ne facilite pas la gestion des clients pour l’entraîneur, surtout en cas de liste longue.