PROJET EN INFORMATIQUE

INF34515-7T

UNIVERSITÉ DU QUÉBEC À RIMOUSKI

DÉPARTEMENT DE MATHÉMATIQUES, D'INFORMATIQUE ET DE GÉNIE

RAPPORT

Rapport Technique

ÉLÈVE :
Bah Oumar Diogo

PROFESSEUR :
Yacine Yaddadden

Date : 16 août 2025

Table des matières

1 Vue d'Ensemble **6**
1.1 Fonctionnalités Principales 6
1.2 Cibles et Technologies 6

2 Identité Visuelle et Thème **6**
2.1 Principes de Design 7

3 Architecture Logicielle **7**
3.1 Structure des Couches 7
3.1.1 Core Layer (Couche Noyau) 7
3.1.2 Domain Layer (Couche Métier) 8
3.1.3 Data Layer (Couche Données) 8
3.1.4 Presentation Layer (Couche Présentation) 9
3.2 Principes d'Architecture 9
3.2.1 Séparation des Responsabilités 9
3.2.2 Dépendances Unidirectionnelles 9
3.2.3 Inversion de Dépendances 10
3.2.4 Testabilité 10
3.2.5 Maintenabilité 10
3.3 Avantages de cette Architecture 10
3.4 Injection de Dépendances 10
3.4.1 Configuration du Service Locator 10
3.4.2 Hiérarchie des Dépendances 10
3.4.3 Exemples d'Enregistrements 11
3.4.4 Avantages de cette Approche 11

4 Domaine et Entités **12**
4.1 Entités Principales 12
4.1.1 Gestion des Utilisateurs (3 entités) 12
4.1.2 Gestion des Associations (3 entités) 13
4.1.3 Gestion des Livres (2 entités) 13
4.1.4 Gestion des Salles (2 entités) 14
4.1.5 Gestion de la Cantine (2 entités) 14
4.1.6 Gestion de la Messagerie (2 entités) 14
4.1.7 Gestion des Horaires (1 entité) 14
4.2 Relations entre Entités 16
4.2.1 Relations Many-to-Many 16
4.2.2 Relations One-to-Many 16
4.2.3 Relations de Dépendance 16
4.3 Intégrité des Données 17
4.3.1 Contraintes Métier 17

4.3.2 Validation des Données 17
4.3.3 Gestion des États 17

5 Cas d'Utilisation **18**
5.1 Étudiant (17 cas d'usage) 20
5.1.1 Gestion des Associations (4 cas) 20
5.1.2 Gestion des Livres (4 cas) 20
5.1.3 Gestion des Salles (4 cas) 20
5.1.4 Gestion du Profil (3 cas) 20
5.1.5 Communication (2 cas) 21
5.2 Administrateur (15 cas d'usage) 21
5.2.1 Gestion des Comptes (4 cas) 21
5.2.2 Gestion de la Cantine (4 cas) 21
5.2.3 Gestion des Associations (4 cas) 21
5.2.4 Gestion des Salles (2 cas) 21
5.2.5 Gestion des Privilèges (1 cas) 21
5.3 Association (5 cas d'usage) 22
5.3.1 Gestion du Contenu (3 cas) 22
5.3.2 Gestion des Adhésions (2 cas) 22
5.4 Flux de Travail et Interactions 22
5.4.1 Workflow d'Adhésion 22
5.4.2 Workflow de Réservation de Salle 22
5.4.3 Workflow de Transaction de Livre 22
5.5 Règles Métier et Contraintes 23
5.5.1 Contraintes d'Inscription 23
5.5.2 Contraintes de Réservation 23
5.5.3 Contraintes de Transaction 23

6 Repositories et Services **23**
6.1 Interfaces de Repositories (Domain) 23
6.1.1 Pattern Repository Standard 23
6.1.2 Exemples d'Interfaces Implémentées 24
6.2 Services Métier (Presentation) 25
6.2.1 AuthentificationService 25
6.2.2 StatistiquesService 25
6.2.3 TransactionsService 26
6.2.4 AdhesionsService 26
6.2.5 GestionMembresService 27
6.2.6 MeteoService 27
6.2.7 MessagerieService 27
6.3 Implémentation des Services 27
6.3.1 Pattern Singleton 27

6.3.2 Gestion des Erreurs 27
6.3.3 Performance et Optimisation 28
6.3.4 Sécurité et Validation 28

7 Écrans et Widgets **28**
7.1 Écrans Principaux 29
7.1.1 Écrans Utilisateur (18 écrans) 29
7.1.2 Écrans Administrateur (7 écrans) 31
7.2 Widgets Réutilisables 32
7.2.1 WidgetCarte 32
7.2.2 WidgetCollection 33
7.2.3 WidgetSectionStatistiques 33
7.2.4 Widgets de Navigation 33
7.2.5 Widgets Spécialisés 34
7.3 Architecture des Widgets 34
7.3.1 Hiérarchie des Composants 34
7.3.2 Système de Thème 35
7.3.3 Gestion des États 35
7.3.4 Performance et Optimisation 35

8 Qualité UI/UX **35**
8.1 Règles de Design Appliquées 35
8.1.1 Respect de la Charte UQAR 35
8.1.2 Gestion des États 36
8.1.3 Responsive Design 37
8.2 Composants UI Modernes 37
8.2.1 AppBar Personnalisées 37
8.2.2 Formulaires Intelligents 37
8.2.3 Navigation Fluide 38
8.3 Accessibilité et Inclusion 38
8.3.1 Standards d'Accessibilité 38
8.3.2 Inclusion et Diversité 39
8.4 Performance et Optimisation 39
8.4.1 Rendu Optimisé 39
8.4.2 Expérience Utilisateur 39

9 Gestion des Données **39**
9.1 Sources de Données 39
9.1.1 Datasources Locales (15 sources) 39
9.1.2 Datasources Distantes (1 source actuelle) 41
9.2 Modèles de Données 42
9.2.1 Structure des Modèles 42
9.2.2 Exemples de Modèles Implémentés 42

10 Références **42**
10.1 Bibliographie Principale 42
10.2 Vidéos YouTube 43
10.3 Outils et Ressources Numériques 43

1 Vue d'Ensemble

L'application **UqarLive** est une solution mobile développée avec le framework Flutter pour répondre aux besoins de la communauté universitaire de l'UQAR. Elle centralise plusieurs services de la vie étudiante et administrative au sein d'une même interface conviviale.
Video de presentation de l'application : `https://youtu.be/4MYa2c9ddGw`

1.1 Fonctionnalités Principales

— **Associations** : découverte des associations, suivi des actualités, consultation et inscription aux événements, gestion des adhésions
— **Cantine** : affichage des menus, menu du jour et horaires d'ouverture
— **Salles** : consultation et réservation des salles disponibles
— **Livres** : marketplace d'échange ou de vente de manuels scolaires
— **Comptes** : gestion des utilisateurs et des profils

1.2 Cibles et Technologies

— **Plateformes** : Android, iOS, Web et Desktop (Windows, macOS, Linux)
— **Framework** : Flutter (Dart)
— **Architecture** : Clean Architecture (Presentation / Domain / Data / Core)
— **Injection de dépendances** : GetIt (Service Locator)
— **Base de données** : Système hybride local/cloud
— **Sécurité** : Authentification multi-facteurs, chiffrement des données

2 Identité Visuelle et Thème

Le thème de l'application est inspiré de la charte graphique officielle de l'UQAR. La palette de couleurs comprend :

— **Couleur principale** : `#005499` (bleu UQAR)
— **Couleur accent** : `#00A1E4` (bleu ciel UQAR)
— **Couleur de fond** : `#F8F9FA` (gris très clair)
— **Texte foncé** : `#2C2C2C` (gris foncé)
— **Blanc** : `#FFFFFF` (pour les contrastes)

FIGURE 1 – logo de notre application

2.1 Principes de Design

— **Design "Roundy"** : Coins arrondis systématiquement appliqués (16-24px)
— **Ombres discrètes** : Élévation subtile avec `boxShadow` et `alpha: 0.08`
— **Typographie cohérente** : Styles centralisés via `StylesTexteApp`
— **Responsive Design** : Adaptation à toutes les tailles d'écran avec `MediaQuery`
— **Accessibilité** : Contrastes appropriés et navigation intuitive

3 Architecture Logicielle

L'application suit une architecture en couches distinctes basée sur les principes de Clean Architecture, implémentant une séparation claire des responsabilités et une inversion de dépendances.

3.1 Structure des Couches

L'architecture est organisée en quatre couches principales, chacune ayant des responsabilités bien définies :

3.1.1 Core Layer (Couche Noyau)

— **Dependency Injection** : ServiceLocator basé sur GetIt pour la gestion centralisée des dépendances

— **Thème et Styles** : Système de design centralisé respectant la charte graphique UQAR
— **Utilitaires** : Helpers métier et fonctions communes réutilisables
— **Configuration** : Paramètres globaux et constantes de l'application

3.1.2 Domain Layer (Couche Métier)

— **Entités Métier** : 15 entités pures représentant les concepts métier (Utilisateur, Association, Livre, etc.)
— **Interfaces Repository** : 14 contrats définissant les opérations CRUD pour chaque entité
— **Règles Métier** : Logique applicative et validation des données
— **Cas d'Usage** : Orchestration des opérations métier complexes

3.1.3 Data Layer (Couche Données)

— **DataSources** : 15 sources de données locales et distantes (MeteoDatasourceRemote)
— **Modèles de Données** : 15 modèles adaptés au stockage local
— **Implémentations Repository** : 14 implémentations concrètes des interfaces
— **Stockage Local** : Système de persistance des données utilisateur

3.1.4 Presentation Layer (Couche Présentation)

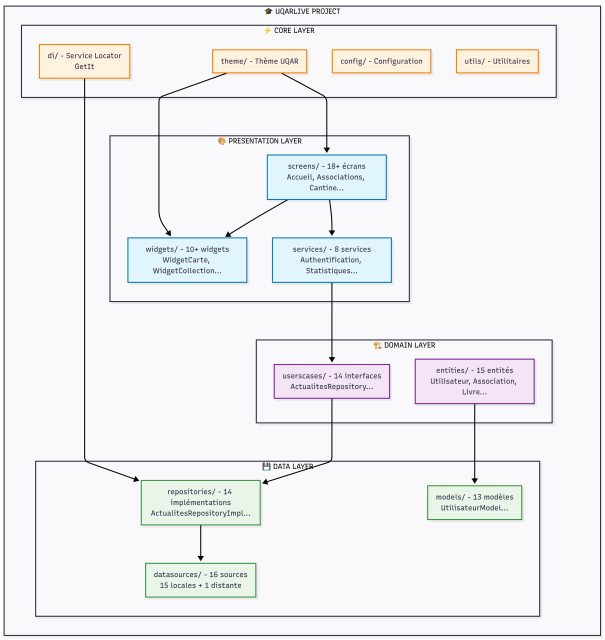

FIGURE 2 – structure de l'application UqarLive

— **Écrans** : 25+ écrans organisés par module fonctionnel
— **Services Métier** : 8 services orchestrant la logique d'interface
— **Widgets Réutilisables** : 10+ composants UI standardisés
— **Navigation** : Système de routage et gestion des transitions

3.2 Principes d'Architecture

3.2.1 Séparation des Responsabilités

Chaque couche a un rôle spécifique et ne peut pas être contournée. La couche Presentation gère uniquement l'interface utilisateur, la couche Domain contient la logique métier pure, et la couche Data s'occupe de la persistance et de l'accès aux données.

3.2.2 Dépendances Unidirectionnelles

Les dépendances suivent un flux strict : Presentation → Domain → Data. La couche Domain ne connaît pas la couche Data, et la couche Core est accessible par toutes les couches via le ServiceLocator.

3.2.3 Inversion de Dépendances

Les couches internes (Domain) définissent des interfaces que les couches externes (Data) implémentent. Cela permet de changer l'implémentation sans affecter la logique métier.

3.2.4 Testabilité

L'architecture facilite les tests unitaires en permettant la substitution des dépendances via des mocks. Chaque couche peut être testée indépendamment.

3.2.5 Maintenabilité

Le code est organisé de manière modulaire avec des responsabilités claires, facilitant la maintenance et l'évolution de l'application.

3.3 Avantages de cette Architecture

- **Évolutivité** : Facilite l'ajout de nouvelles fonctionnalités sans affecter l'existant
- **Réutilisabilité** : Les composants peuvent être réutilisés dans différents contextes
- **Sécurité** : Isolation des couches réduit les risques de propagation d'erreurs
- **Performance** : Optimisation possible de chaque couche indépendamment

3.4 Injection de Dépendances

Le système d'injection de dépendances est implémenté via le pattern Service Locator utilisant la bibliothèque GetIt, permettant une gestion centralisée et efficace des dépendances entre les couches de l'application.

3.4.1 Configuration du Service Locator

Le ServiceLocator est configuré au démarrage de l'application via la méthode `configurerDependances()` qui enregistre toutes les relations entre interfaces et implémentations. Cette approche centralisée facilite la maintenance et permet de changer facilement les implémentations.

3.4.2 Hiérarchie des Dépendances

L'architecture respecte une hiérarchie stricte où chaque couche ne dépend que des couches qui lui sont autorisées :

- **Services Métier** → **Interfaces Repository** → **Implémentations Repository** → **DataSources**
- **Écrans** → **Services Métier** → **Interfaces Repository**
- **Widgets** → **Services Métier** ou **Directement aux Repositories**

3.4.3 Exemples d'Enregistrements

Le système enregistre 14 paires interface-implémentation :

- **ActualitesRepository → ActualitesRepositoryImpl → ActualitesDatasourceLocal**
- **AssociationsRepository → AssociationsRepositoryImpl → AssociationsDatasourceLocal**
- **EvenementsRepository → EvenementsRepositoryImpl → EvenementsDatasourceLocal**
- **LivresRepository → LivresRepositoryImpl → LivresDatasourceLocal**
- **MenusRepository → MenusRepositoryImpl → MenusDatasourceLocal**
- **SallesRepository → SallesRepositoryImpl → SallesDatasourceLocal**
- **UtilisateursRepository → UtilisateursRepositoryImpl → UtilisateursDatasourceLocal**
- **MembresAssociationRepository → MembresAssociationRepositoryImpl → MembresAssociationDatasourceLocal**
- **ReservationsSalleRepository → ReservationsSalleRepositoryImpl → ReservationsSalleDatasourceLocal**
- **TransactionsRepository → TransactionsRepositoryImpl → TransactionsDatasourceLocal**
- **DemandesAdhesionRepository → DemandesAdhesionRepositoryImpl → DemandesAdhesionDatasourceLocal**
- **MessagesRepository → MessagesRepositoryImpl → MessagesDatasourceLocal**
- **HorairesRepository → HorairesRepositoryImpl → HorairesDatasourceLocal**
- **MeteoRepository → MeteoRepositoryImpl → MeteoDatasourceRemote**

3.4.4 Avantages de cette Approche

- **Flexibilité** : Possibilité de changer d'implémentation sans modifier le code client
- **Testabilité** : Facilite l'injection de mocks pour les tests
- **Maintenabilité** : Configuration centralisée des dépendances
- **Performance** : Instances partagées et gestion optimisée de la mémoire

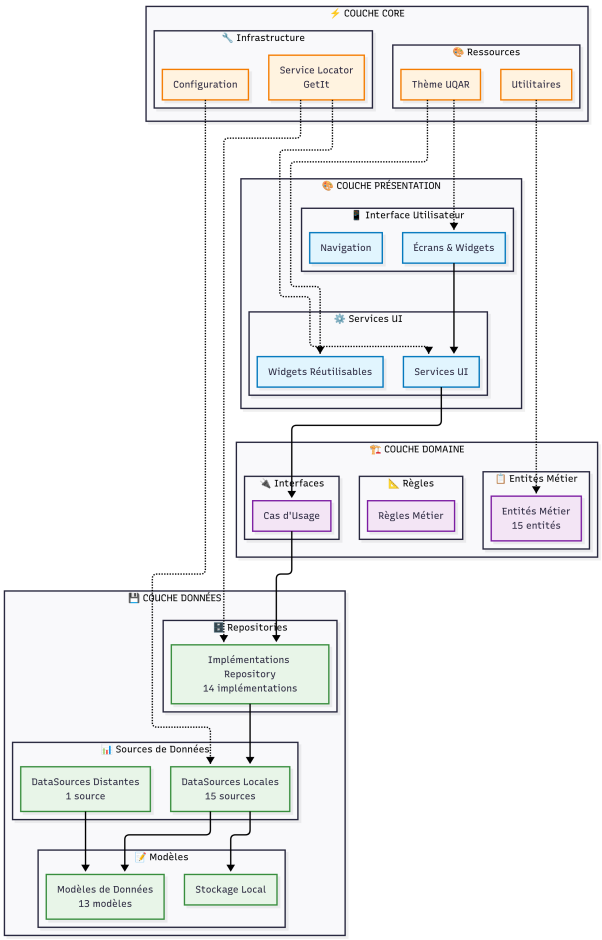

FIGURE 3 – Architecture de l'application UqarLive

4 Domaine et Entités

Le domaine métier de l'application UqarLive est structuré autour de 15 entités principales, chacune représentant un concept métier spécifique de la vie étudiante universitaire.

4.1 Entités Principales

4.1.1 Gestion des Utilisateurs (3 entités)

— **Utilisateur** : Entité centrale représentant un étudiant ou membre du personnel UQAR
  — **Attributs** : 15 propriétés incluant identité, informations académiques, privilèges et métadonnées

  - **Méthodes métier** : `estAdmin()`, `aPrivilege()`, `estMembreAssociations()`, `nombreAssociations()`
  - **Validation** : Vérification des formats email, téléphone et codes étudiants
  - **Sécurité** : Gestion des privilèges et contrôle d'accès
- **MembreAssociation** : Entité de liaison entre utilisateur et association
  - **Attributs** : Rôle, date d'adhésion, statut actif, commentaires
  - **Relations** : Clés étrangères vers Utilisateur et Association
  - **Métadonnées** : Suivi de l'historique d'adhésion
- **DemandeAdhesion** : Gestion des candidatures aux associations
  - **Attributs** : Statut de la demande, date de soumission, motivation
  - **Workflow** : États de traitement (en attente, approuvée, rejetée)

4.1.2 Gestion des Associations (3 entités)

- **Association** : Entité représentant une association étudiante UQAR
  - **Attributs** : 25+ propriétés incluant informations de contact, réseaux sociaux, activités
  - **Méthodes métier** : `nombreMembresFormatte()`, `aDesMembres()`, `aDesContacts()`
  - **Fonctionnalités** : Gestion des présidents, vice-présidents, cotisations annuelles
  - **Médias** : Support des logos, descriptions longues et bénéfices membres
- **Actualite** : Publications d'actualités par les associations
  - **Attributs** : Contenu, priorité, tags, épinglage, images
  - **Fonctionnalités** : Système de priorité et épinglage des actualités importantes
  - **Métadonnées** : Suivi des publications et engagement
- **Evenement** : Organisation d'événements par les associations
  - **Attributs** : 15+ propriétés incluant lieu, dates, capacité, inscriptions
  - **Méthodes métier** : `estAVenir()`, `estEnCours()`, `estTermine()`, `estComplet()`
  - **Types** : Conférences, ateliers, événements sociaux, sportifs, culturels, académiques
  - **Gestion** : Système d'inscription avec capacité maximale et suivi des participants

4.1.3 Gestion des Livres (2 entités)

- **Livre** : Manuels scolaires disponibles à l'échange
  - **Attributs** : 18 propriétés incluant métadonnées académiques et commerciales
  - **Méthodes métier** : `codeCours()` (alias pour coursAssocies)
  - **Fonctionnalités** : Support des mots-clés, images, états et prix
  - **Validation** : Vérification de la disponibilité et des informations propriétaire
- **Transaction** : Opérations d'achat/vente/échange de livres
  - **Attributs** : Type de transaction, montant, statut, dates
  - **Types** : Achat, vente, échange avec ou sans compensation
  - **Suivi** : Historique complet des transactions utilisateur

4.1.4 Gestion des Salles (2 entités)

- **Salle** : Espaces de révision et d'étude disponibles
  - **Attributs** : Capacité, bâtiment, équipements, disponibilité
  - **Fonctionnalités** : Gestion des équipements et localisation
  - **Validation** : Vérification de la disponibilité et des conflits
- **ReservationSalle** : Réservations de salles par les étudiants
  - **Attributs** : Dates de début/fin, motif, statut, utilisateur
  - **Validation** : Vérification des conflits de réservation
  - **Gestion** : Système de réservation avec motifs et statuts

4.1.5 Gestion de la Cantine (2 entités)

- **Menu** : Plats et menus disponibles à la cantine
  - **Attributs** : 12 propriétés incluant informations nutritionnelles et restrictions
  - **Fonctionnalités** : Support des régimes (végétarien, sans gluten, halal)
  - **Gestion** : Menu du jour et disponibilité par jour
- **Horaire** : Horaires d'ouverture et de service de la cantine
  - **Attributs** : Jours, heures d'ouverture, fermeture, pauses
  - **Fonctionnalités** : Gestion des horaires variables selon les périodes

4.1.6 Gestion de la Messagerie (2 entités)

- **Message** : Communication entre utilisateurs
  - **Attributs** : Contenu, expéditeur, destinataire, date, statut
  - **Fonctionnalités** : Système de messagerie privée intégrée
- **Contact** : Gestion des contacts et conversations
  - **Attributs** : Relations entre utilisateurs, statut de contact
  - **Fonctionnalités** : Gestion des listes de contacts

4.1.7 Gestion des Horaires (1 entité)

- **Horaire** : Horaires généraux de l'université
  - **Attributs** : Périodes, jours, heures, type d'activité
  - **Fonctionnalités** : Gestion des horaires académiques et administratifs

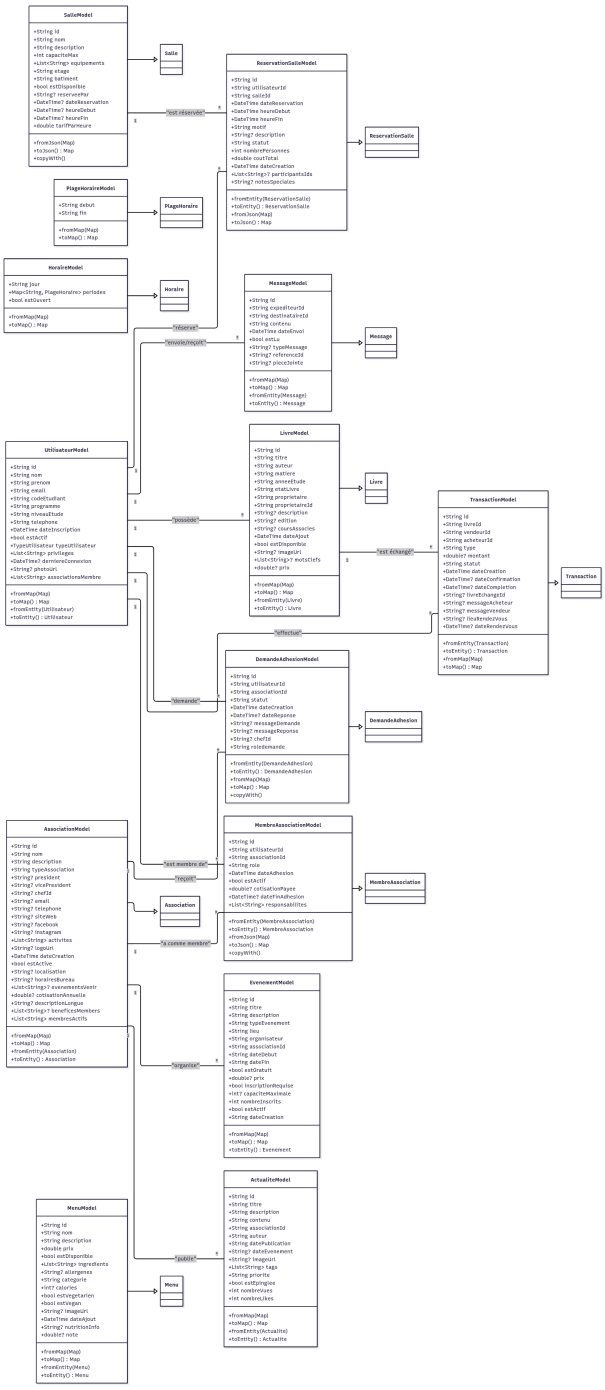

FIGURE 4 – Diagramme de classes des entités métier UqarLive

4.2 Relations entre Entités

Le modèle de données implémente un système de relations complexes respectant les règles métier universitaires :

4.2.1 Relations Many-to-Many

— **Utilisateur Association** : Via MembreAssociation avec rôles et statuts
— **Utilisateur Livre** : Via Transaction avec historique complet
— **Utilisateur Salle** : Via ReservationSalle avec gestion des conflits
— **Utilisateur Evenement** : Via système d'inscription avec capacité

4.2.2 Relations One-to-Many

— **Association** → **Actualite** : Une association peut publier plusieurs actualités
— **Association** → **Evenement** : Une association peut organiser plusieurs événements
— **Utilisateur** → **Livre** : Un utilisateur peut posséder plusieurs livres
— **Utilisateur** → **ReservationSalle** : Un utilisateur peut avoir plusieurs réservations

4.2.3 Relations de Dépendance

— **MembreAssociation** : Dépend de Utilisateur et Association
— **Transaction** : Dépend de Livre, Utilisateur (acheteur/vendeur)
— **ReservationSalle** : Dépend de Salle et Utilisateur
— **Actualite/Evenement** : Dépendent de Association

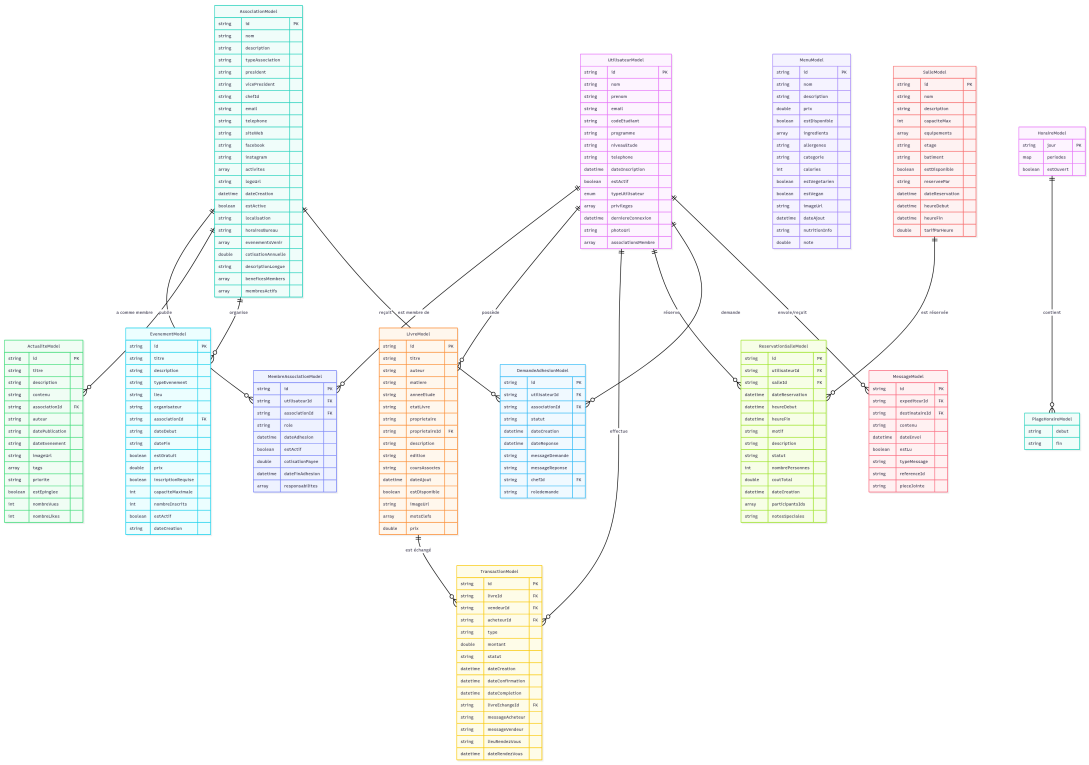

FIGURE 5 – Modèle relationnel de la base de données UqarLive

4.3 Intégrité des Données

4.3.1 Contraintes Métier

— Un utilisateur ne peut pas réserver une salle déjà réservée sur le même créneau
— Un livre ne peut être vendu/échangé que par son propriétaire
— Un événement ne peut dépasser sa capacité maximale
— Une association doit avoir au moins un président actif

4.3.2 Validation des Données

— Formats des emails et téléphones
— Dates cohérentes (début ¡ fin pour événements/réservations)
— Prix positifs pour les transactions
— Capacités positives pour les salles et événements

4.3.3 Gestion des États

— Statuts des demandes d'adhésion
— Disponibilité des livres et salles

— Actif/inactif des associations et utilisateurs
— Statuts des transactions et réservations

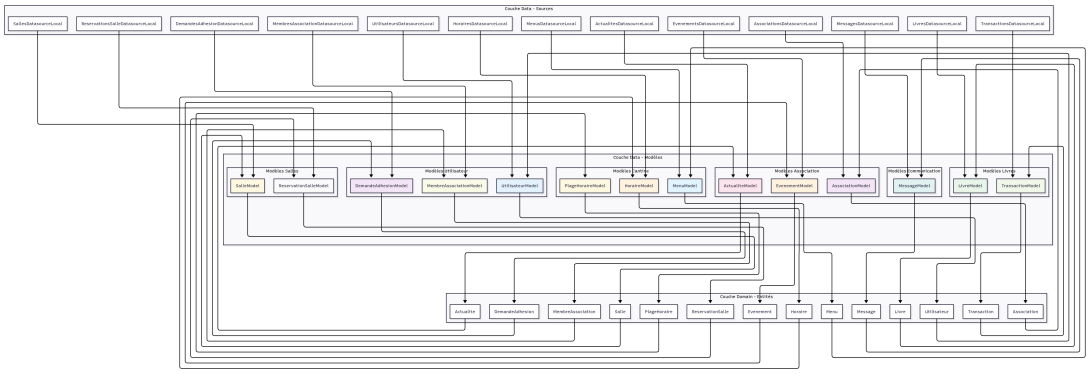

FIGURE 6 – Modele d'architecture de l'application UqarLive

5 Cas d'Utilisation

L'application UqarLive implémente 36 cas d'utilisation répartis entre trois acteurs principaux, couvrant l'ensemble des fonctionnalités de la vie étudiante universitaire.

FIGURE 7 – FIGURE 7 - Diagramme des cas d'usage de l'application UqarLive

5.1 Étudiant (17 cas d'usage)

5.1.1 Gestion des Associations (4 cas)

— **Consulter les actualités** : Accès au flux d'actualités des associations avec filtrage par priorité et épinglage
— **S'inscrire à un événement** : Inscription aux événements avec vérification de capacité et gestion des conflits
— **Demander l'adhésion** : Soumission de candidature avec motivation et suivi du statut
— **Consulter les détails** : Accès aux informations complètes des associations (contacts, activités, bénéfices)

5.1.2 Gestion des Livres (4 cas)

— **Parcourir le marketplace** : Navigation dans le catalogue avec filtres par matière, année, état et prix
— **Acheter/échanger un livre** : Transaction sécurisée avec validation des propriétaires et gestion des échanges
— **Gérer sa collection** : Ajout, modification et suppression de livres personnels
— **Consulter l'historique** : Suivi complet des transactions (achats, ventes, échanges)

5.1.3 Gestion des Salles (4 cas)

— **Consulter la disponibilité** : Vue des salles libres avec filtres par capacité, équipements et bâtiment
— **Réserver une salle** : Réservation avec sélection de créneaux et validation des conflits
— **Gérer ses réservations** : Consultation, modification et annulation des réservations actives
— **Consulter les horaires** : Accès aux horaires d'ouverture et de fermeture

5.1.4 Gestion du Profil (3 cas)

— **Consulter le profil** : Affichage des informations personnelles, académiques et statistiques
— **Modifier les informations** : Mise à jour des données personnelles avec validation
— **Consulter les statistiques** : Métriques d'utilisation et d'engagement dans l'application

5.1.5 Communication (2 cas)

— **Envoyer/recevoir des messages** : Messagerie privée avec gestion des contacts
— **Gérer les contacts** : Ajout et organisation de la liste de contacts

5.2 Administrateur (15 cas d'usage)

5.2.1 Gestion des Comptes (4 cas)

— **Créer des comptes** : Création de nouveaux utilisateurs avec attribution de rôles et privilèges
— **Modifier les informations** : Mise à jour des profils utilisateurs et gestion des permissions
— **Suspendre/réactiver** : Gestion du statut actif des comptes avec justification
— **Consulter les statistiques** : Métriques système et utilisation des fonctionnalités

5.2.2 Gestion de la Cantine (4 cas)

— **Ajouter des menus** : Création de nouveaux plats avec informations nutritionnelles et restrictions
— **Modifier les menus** : Mise à jour des informations et disponibilité des plats
— **Gérer les horaires** : Configuration des heures d'ouverture et de fermeture
— **Définir le menu du jour** : Sélection et promotion du menu principal quotidien

5.2.3 Gestion des Associations (4 cas)

— **Créer des associations** : Création de nouvelles associations avec attribution des responsables
— **Modifier les associations** : Mise à jour des informations et gestion des contacts
— **Gérer le contenu** : Supervision des actualités et événements publiés
— **Approuver les adhésions** : Validation des demandes d'adhésion avec notifications

5.2.4 Gestion des Salles (2 cas)

— **Gérer les salles** : Ajout, modification et suppression des espaces disponibles
— **Superviser les réservations** : Consultation des réservations actives et gestion des conflits

5.2.5 Gestion des Privilèges (1 cas)

— **Gérer les privilèges** : Attribution et retrait des permissions utilisateurs

5.3 Association (5 cas d'usage)

5.3.1 Gestion du Contenu (3 cas)

— **Publier des actualités** : Création et publication d'actualités avec système de priorité
— **Créer des événements** : Organisation d'événements avec gestion des inscriptions
— **Gérer les membres** : Consultation de la liste des membres et gestion des rôles

5.3.2 Gestion des Adhésions (2 cas)

— **Approuver les adhésions** : Validation des candidatures avec notifications automatiques
— **Consulter les statistiques** : Métriques d'engagement et d'activité de l'association

5.4 Flux de Travail et Interactions

5.4.1 Workflow d'Adhésion

1. **Soumission** : Étudiant soumet une demande d'adhésion
2. **Notification** : Association reçoit la notification de candidature
3. **Évaluation** : Responsables évaluent la motivation et le profil
4. **Décision** : Approbation ou rejet avec justification
5. **Notification** : Étudiant reçoit la décision par l'application

5.4.2 Workflow de Réservation de Salle

1. **Consultation** : Étudiant consulte la disponibilité des salles
2. **Sélection** : Choix du créneau et de la salle appropriée
3. **Validation** : Vérification des conflits et disponibilité
4. **Confirmation** : Réservation confirmée avec notification
5. **Gestion** : Possibilité de modification ou annulation

5.4.3 Workflow de Transaction de Livre

1. **Recherche** : Étudiant parcourt le marketplace
2. **Sélection** : Choix du livre avec consultation des détails
3. **Négociation** : Contact avec le propriétaire pour les conditions
4. **Transaction** : Finalisation de l'échange ou de l'achat
5. **Suivi** : Mise à jour des statuts et historique

5.5 Règles Métier et Contraintes

5.5.1 Contraintes d'Inscription

— Un étudiant ne peut s'inscrire qu'à un événement à la fois sur le même créneau
— La capacité maximale des événements doit être respectée
— Les inscriptions sont closes avant le début de l'événement

5.5.2 Contraintes de Réservation

— Une salle ne peut être réservée que par un seul utilisateur sur un créneau
— Les réservations ne peuvent pas déborder sur les horaires de fermeture
— Un utilisateur ne peut avoir qu'une réservation active par salle

5.5.3 Contraintes de Transaction

— Un livre ne peut être vendu que par son propriétaire
— Les transactions doivent respecter les conditions d'échange définies
— Le statut du livre doit être mis à jour après chaque transaction

6 Repositories et Services

L'architecture de l'application implémente un système complet de repositories et de services métier, organisant la logique applicative et l'accès aux données selon les principes de Clean Architecture.

6.1 Interfaces de Repositories (Domain)

Le domaine définit 14 interfaces de repositories, chacune spécifiant les opérations CRUD et métier pour une entité donnée. Ces interfaces garantissent la séparation des préoccupations et facilitent les tests unitaires.

6.1.1 Pattern Repository Standard

Chaque interface implémente un ensemble standard d'opérations :
— **Opérations CRUD** : Création, lecture, mise à jour et suppression des entités
— **Opérations de Recherche** : Filtrage, pagination et recherche textuelle
— **Opérations Métier** : Actions spécifiques au domaine (inscription, réservation, transaction)
— **Gestion des Relations** : Opérations sur les entités liées et jointures

6.1.2 Exemples d'Interfaces Implémentées

ActualitesRepository

- `obtenirActualites()` : Récupération de toutes les actualités avec pagination
- `obtenirActualiteParId(String id)` : Récupération d'une actualité spécifique
- `ajouterActualite(Actualite actualite)` : Création d'une nouvelle actualité
- `modifierActualite(Actualite actualite)` : Mise à jour d'une actualité existante
- `supprimerActualite(String id)` : Suppression logique d'une actualité
- `obtenirActualitesParAssociation(String associationId)` : Filtrage par association
- `obtenirActualitesEpinglees()` : Récupération des actualités prioritaires

AssociationsRepository

- `obtenirToutesLesAssociations()` : Liste complète des associations actives
- `obtenirAssociationParId(String id)` : Détails d'une association spécifique
- `filtrerAssociations(String type)` : Filtrage par type d'association
- `ajouterAssociation(Association association)` : Création d'une nouvelle association
- `modifierAssociation(Association association)` : Mise à jour des informations
- `supprimerAssociation(String id)` : Désactivation d'une association
- `obtenirAssociationsParUtilisateur(String utilisateurId)` : Associations d'un utilisateur

EvenementsRepository

- `obtenirTousLesEvenements()` : Liste des événements avec filtres
- `obtenirEvenementParId(String id)` : Détails d'un événement
- `ajouterEvenement(Evenement evenement)` : Création d'un nouvel événement
- `inscrireUtilisateur(String evenementId, String utilisateurId)` : Inscription
- `desinscrireUtilisateur(String evenementId, String utilisateurId)` : Désinscription
- `obtenirEvenementsParAssociation(String associationId)` : Événements d'une association
- `obtenirEvenementsAVenir()` : Événements futurs

LivresRepository

- `obtenirTousLesLivres()` : Catalogue complet des livres disponibles
- `obtenirLivreParId(String id)` : Détails d'un livre
- `ajouterLivre(Livre livre)` : Ajout d'un livre au marketplace
- `modifierLivre(Livre livre)` : Mise à jour des informations
- `supprimerLivre(String id)` : Retrait du marketplace
- `filtrerLivres(String matiere, String annee, double? prixMax)` : Filtrage avancé
- `obtenirLivresParProprietaire(String proprietaireId)` : Livres d'un utilisa-

teur

UtilisateursRepository

- `authentifierUtilisateur(String email, String motDePasse)` : Authentification
- `obtenirUtilisateurActuel()` : Récupération de l'utilisateur connecté
- `deconnecterUtilisateur()` : Déconnexion et nettoyage de session
- `ajouterUtilisateur(Utilisateur utilisateur)` : Création de compte
- `modifierUtilisateur(Utilisateur utilisateur)` : Mise à jour du profil
- `obtenirUtilisateurParId(String id)` : Récupération par identifiant
- `modifierPrivileges(String utilisateurId, List<String> privileges)` : Gestion des droits

6.2 Services Métier (Presentation)

L'application implémente 8 services métier qui orchestrent la logique applicative et coordonnent les interactions entre les différentes couches.

6.2.1 AuthentificationService

Service centralisé gérant l'ensemble du cycle de vie de l'authentification utilisateur.

Fonctionnalités Principales

- **Gestion des Sessions** : Maintien de l'état de connexion utilisateur
- **Validation des Identifiants** : Vérification des emails et mots de passe
- **Gestion des Rôles** : Attribution et vérification des privilèges utilisateur
- **Sécurisation des Données** : Protection des informations sensibles
- **Gestion des Déconnexions** : Nettoyage sécurisé des sessions

Méthodes Implémentées

- `authentifier(String email, String motDePasse)` : Connexion utilisateur
- `chargerUtilisateurActuel()` : Récupération de la session au démarrage
- `deconnecter()` : Déconnexion sécurisée
- `actualiserUtilisateurActuel()` : Mise à jour des informations utilisateur
- `aPrivilege(String privilege)` : Vérification des droits spécifiques
- `estAdministrateur` : Vérification du statut administrateur
- `modifierPrivileges(String utilisateurId, List<String> nouveauxPrivileges)` : Gestion des droits

6.2.2 StatistiquesService

Service d'agrégation et d'analyse des métriques d'utilisation de l'application.

Métriques Collectées

- **Utilisateurs** : Total, actifs, nouveaux, types et répartition
- **Associations** : Nombre, membres, activités et engagement

- **Livres** : Catalogue, transactions et échanges
- **Salles** : Réservations, utilisation et disponibilité
- **Cantine** : Menus, fréquentation et préférences
- **Système** : Performance, erreurs et temps de réponse

Fonctionnalités

- Agrégation automatique des données d'utilisation
- Calcul des statistiques de performance et d'engagement
- Génération de rapports d'activité détaillés
- Analyse des comportements utilisateurs et tendances

6.2.3 TransactionsService

Service de gestion des opérations commerciales sur le marketplace de livres.

Types de Transactions

- **Achat** : Transaction monétaire pour l'acquisition d'un livre
- **Vente** : Mise en vente d'un livre avec prix fixe
- **Échange** : Troc de livres entre utilisateurs
- **Échange avec Compensation** : Échange avec différence monétaire

Fonctionnalités

- Validation des propriétaires et disponibilité des livres
- Gestion des conditions d'échange et négociations
- Suivi des statuts de transaction (en cours, finalisée, annulée)
- Historique complet des opérations utilisateur
- Gestion des conflits et litiges

6.2.4 AdhesionsService

Service de gestion des demandes d'adhésion aux associations étudiantes.

Workflow d'Adhésion

1. **Soumission** : Validation de la candidature et des informations
2. **Notification** : Alerte automatique aux responsables d'association
3. **Évaluation** : Processus de validation par les administrateurs
4. **Décision** : Approbation ou rejet avec justification
5. **Notification** : Communication du résultat au candidat

Fonctionnalités

- Validation des informations de candidature
- Gestion des statuts de demande (en attente, approuvée, rejetée)
- Notifications automatiques aux parties prenantes
- Suivi de l'historique des candidatures
- Gestion des quotas et restrictions d'adhésion

6.2.5 GestionMembresService

Service de gestion des membres des associations et de leurs rôles.
Fonctionnalités
- Gestion des adhésions et désadhésions
- Attribution et modification des rôles membres
- Suivi des statuts d'adhésion (actif, inactif, suspendu)
- Gestion des permissions et accès
- Communication avec les membres

6.2.6 MeteoService

Service d'intégration avec l'API météo externe pour afficher les conditions météorologiques.
Fonctionnalités
- Récupération des données météo en temps réel
- Affichage des conditions actuelles et prévisions
- Intégration dans l'interface utilisateur
- Mise à jour automatique des informations

6.2.7 MessagerieService

Service de gestion de la communication entre utilisateurs.
Fonctionnalités
- Envoi et réception de messages privés
- Gestion des contacts et conversations
- Notifications de nouveaux messages
- Historique des conversations
- Gestion des statuts de lecture

6.3 Implémentation des Services

6.3.1 Pattern Singleton

La plupart des services utilisent le pattern Singleton pour garantir une instance unique et partagée dans l'application.

6.3.2 Gestion des Erreurs

Chaque service implémente une gestion robuste des erreurs avec :
- Try-catch appropriés pour les opérations asynchrones
- Messages d'erreur informatifs pour l'utilisateur
- Logs détaillés pour le débogage
- Fallbacks et états de récupération

6.3.3 Performance et Optimisation

— Mise en cache des données fréquemment consultées
— Opérations asynchrones pour éviter le blocage de l'UI
— Pagination des résultats volumineux
— Optimisation des requêtes de base de données

6.3.4 Sécurité et Validation

— Validation des entrées utilisateur
— Vérification des permissions et autorisations
— Protection contre les injections et attaques
— Chiffrement des données sensibles

7 Écrans et Widgets

L'interface utilisateur de UqarLive est organisée en 25+ écrans principaux, organisés par module fonctionnel, et utilise un système de widgets réutilisables pour garantir la cohérence visuelle et la maintenabilité du code.

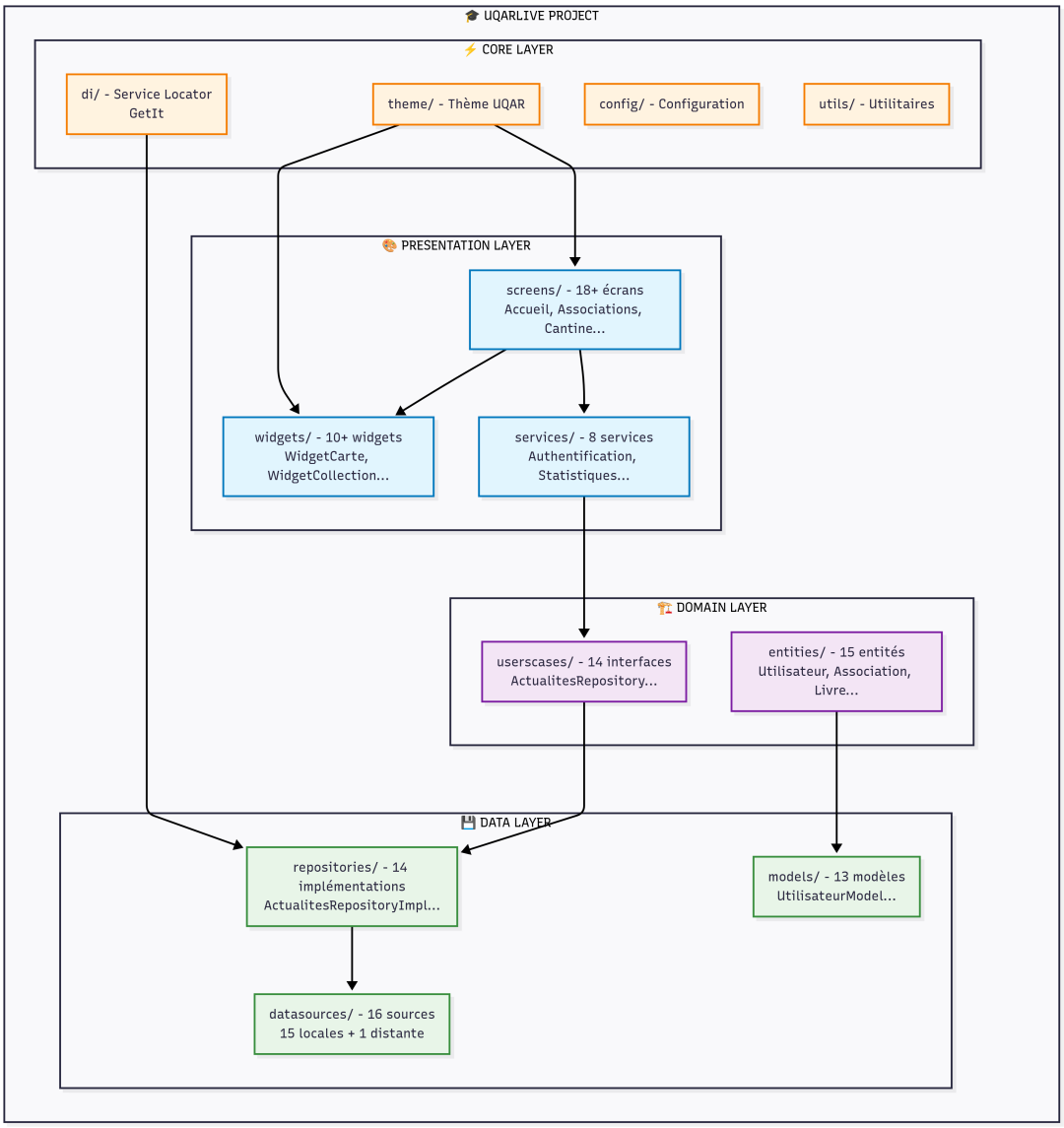

FIGURE 8 – Structure générale de l'interface utilisateur UqarLive

7.1 Écrans Principaux

7.1.1 Écrans Utilisateur (18 écrans)

Module Accueil (1 écran)

- **AccueilEcran** : Point d'entrée principal avec navigation vers tous les modules
  - Sections principales : Associations, Cantine, Livres, Salles, Messagerie
  - Widgets de statistiques personnelles et actualités récentes
  - Navigation rapide vers les fonctionnalités les plus utilisées

Module Associations (4 écrans)

- **AssociationsEcran** : Liste des associations avec filtres et recherche
  - Grille responsive des associations avec informations essentielles
  - Filtrage par type d'association et statut
  - Barre de recherche textuelle intégrée
- **DetailsAssociationEcran** : Informations détaillées d'une association
  - Présentation complète : description, contacts, réseaux sociaux
  - Liste des membres et activités récentes
  - Bouton de demande d'adhésion avec statut
- **AjouterActualiteEcran** : Création d'actualités pour les associations
  - Formulaire complet avec validation des champs
  - Support des images et tags de priorité
  - Prévisualisation avant publication
- **AjouterEvenementEcran** : Création d'événements associatifs
  - Configuration des dates, lieu et capacité
  - Gestion des inscriptions et conditions
  - Support des types d'événements multiples

Module Cantine (2 écrans)

- **CantineEcran** : Vue d'ensemble de la cantine universitaire
  - Menu du jour mis en avant
  - Liste des menus disponibles avec filtres
  - Horaires d'ouverture et informations pratiques
- **DetailsMenuEcran** : Informations détaillées d'un menu
  - Composition complète et valeurs nutritionnelles
  - Restrictions alimentaires (végétarien, sans gluten, halal)
  - Prix et disponibilité

Module Livres (4 écrans)

- **MarketplaceEcran** : Catalogue complet des livres disponibles
  - Grille responsive avec filtres avancés
  - Recherche par titre, auteur, matière, année
  - Tri par prix, état et date d'ajout
- **DetailsLivreEcran** : Informations complètes d'un livre
  - Détails académiques et commerciaux
  - Informations du propriétaire et conditions
  - Boutons d'action (acheter, échanger, contacter)
- **GererLivresEcran** : Gestion de la collection personnelle
  - Ajout, modification et suppression de livres
  - Gestion des statuts de disponibilité
  - Historique des transactions
- **SelectionnerLivreEchangeEcran** : Sélection pour échange
  - Interface de choix de livre à échanger
  - Validation des conditions d'échange
  - Confirmation de la transaction

Module Salles (1 écran)
- **SallesEcran** : Gestion des espaces de révision
  - Consultation de la disponibilité en temps réel
  - Interface de réservation avec sélection de créneaux
  - Gestion des réservations actives

Module Messagerie (2 écrans)
- **ConversationsEcran** : Liste des conversations actives
  - Conversations récentes avec aperçu des derniers messages
  - Indicateurs de nouveaux messages non lus
  - Recherche dans l'historique des conversations
- **ConversationEcran** : Interface de chat individuel
  - Envoi et réception de messages en temps réel
  - Historique complet de la conversation
  - Gestion des pièces jointes et emojis

Module Profil (2 écrans)
- **ProfilEcran** : Affichage du profil utilisateur
  - Informations personnelles et académiques
  - Statistiques d'utilisation et d'engagement
  - Accès aux paramètres et préférences
- **ModifierProfilEcran** : Édition des informations personnelles
  - Formulaire de modification avec validation
  - Upload de photo de profil
  - Gestion des préférences de confidentialité

Module Authentification (2 écrans)
- **ConnexionEcran** : Interface de connexion
  - Formulaire de connexion avec validation
  - Gestion des erreurs d'authentification
  - Lien vers l'inscription
- **InscriptionEcran** : Création de compte utilisateur
  - Formulaire d'inscription complet
  - Validation des informations et acceptation des conditions
  - Confirmation de création de compte

7.1.2 Écrans Administrateur (7 écrans)

Module Dashboard (1 écran)
- **AdminDashboardEcran** : Vue d'ensemble administrative
  - Statistiques système et métriques d'utilisation
  - Accès rapide aux fonctions d'administration
  - Alertes et notifications importantes

Module Gestion des Comptes (1 écran)
- **AdminGestionComptesEcran** : Administration des utilisateurs

  - Liste des utilisateurs avec filtres et recherche
  - Création, modification et suspension de comptes
  - Gestion des privilèges et rôles

Module Gestion de la Cantine (2 écrans)

- **AdminGestionCantineEcran** : Administration de la cantine
  - Gestion des menus et horaires
  - Statistiques de fréquentation
  - Configuration des paramètres
- **AdminAjouterMenuEcran** : Création de nouveaux menus
  - Formulaire d'ajout avec validation
  - Configuration des informations nutritionnelles
  - Gestion des restrictions alimentaires

Module Gestion des Associations (2 écrans)

- **AdminGestionAssociationsEcran** : Administration des associations
  - Liste des associations avec statuts
  - Gestion des demandes d'adhésion
  - Supervision des activités
- **AdminAjouterAssociationEcran** : Création de nouvelles associations
  - Formulaire de création avec validation
  - Attribution des responsables et contacts
  - Configuration des paramètres

Module Gestion des Horaires (1 écran)

- **AdminModifierHorairesEcran** : Configuration des horaires
  - Gestion des horaires d'ouverture
  - Configuration des périodes spéciales
  - Synchronisation avec le calendrier académique

7.2 Widgets Réutilisables

L'application utilise un système de 10+ widgets réutilisables pour garantir la cohérence visuelle et faciliter la maintenance du code.

7.2.1 WidgetCarte

Widget principal pour l'affichage de tous les contenus de l'application.

Fonctionnalités

- **Modes d'affichage** : Horizontal et vertical avec adaptation automatique
- **Gestion responsive** : Dimensions adaptatives selon l'écran
- **Thème UQAR** : Application automatique des couleurs et styles
- **États multiples** : Chargement, vide, erreur et normal
- **Interactions** : Support des tap, long press et swipe

Types de Cartes Implémentés

— **CarteAssociation** : Affichage des informations d'association
— **CarteLivre** : Présentation des livres du marketplace
— **CarteMenu** : Affichage des menus de cantine
— **CarteSalle** : Information sur les salles disponibles
— **CarteUtilisateur** : Profils utilisateurs et contacts

7.2.2 WidgetCollection

Widget de gestion des collections et listes de contenu.
Fonctionnalités
— **Orientations** : Horizontale et verticale avec configuration
— **Espacements** : Gestion automatique des marges et paddings
— **États de chargement** : Indicateurs visuels pendant le chargement
— **Gestion du vide** : Placeholders informatifs quand aucune donnée
— **Responsive** : Adaptation automatique selon la taille d'écran
Collections Implémentées
— **CollectionHorizontale** : Navigation horizontale (carrousel)
— **CollectionVerticale** : Liste verticale avec scroll
— **Grille** : Affichage en grille responsive

7.2.3 WidgetSectionStatistiques

Widget spécialisé pour l'affichage des métriques et statistiques.
Fonctionnalités
— **Métriques clé-valeur** : Affichage structuré des données
— **Couleurs thématiques** : Code couleur par type de métrique
— **Gestion des débordements** : Protection contre le texte trop long
— **Animations** : Transitions fluides pour les changements de valeurs
— **Responsive** : Adaptation selon l'espace disponible

7.2.4 Widgets de Navigation

WidgetBarreAppPersonnalisee Barre d'application personnalisée pour les utilisateurs standards.
Fonctionnalités
— Navigation par onglets entre les modules principaux
— Indicateurs de notifications et alertes
— Intégration avec le thème UQAR
— Gestion des états actifs et inactifs
WidgetBarreAppNavigationAdmin Barre d'application spécialisée pour les administrateurs.
Fonctionnalités

— Accès aux fonctions d'administration
— Indicateurs de tâches en attente
— Navigation rapide vers les modules critiques
— Gestion des privilèges et permissions

NavbarWidget Widget de navigation principale de l'application.

Fonctionnalités

— Navigation entre les écrans principaux
— Gestion du breadcrumb et de l'historique
— Boutons d'action contextuels
— Intégration avec le système de thème

7.2.5 Widgets Spécialisés

WidgetBoutonConversations Bouton d'accès rapide aux conversations.

Fonctionnalités

— Indicateur de nouveaux messages
— Accès direct aux conversations récentes
— Intégration avec le système de notifications
— Design cohérent avec le thème

7.3 Architecture des Widgets

7.3.1 Hiérarchie des Composants

```
WidgetCarte (Base)
 CarteAssociation
 CarteLivre
 CarteMenu
 CarteSalle
 CarteUtilisateur

WidgetCollection (Base)
 CollectionHorizontale
 CollectionVerticale
 Grille

Widgets de Navigation
 WidgetBarreAppPersonnalisee
 WidgetBarreAppNavigationAdmin
 NavbarWidget
```

7.3.2 Système de Thème

Tous les widgets utilisent le système de thème centralisé :
- **Couleurs** : Palette UQAR (#005499, #00A1E4, #F8F9FA, #2C2C2C)
- **Typographie** : Styles centralisés via `StylesTexteApp`
- **Décorations** : Coins arrondis et ombres cohérentes
- **Espacements** : Marges et paddings standardisés

7.3.3 Gestion des États

Chaque widget gère plusieurs états :
- **Normal** : Affichage standard du contenu
- **Chargement** : Indicateurs visuels pendant les opérations
- **Vide** : Placeholders quand aucune donnée
- **Erreur** : Messages d'erreur informatifs
- **Interactif** : États de hover, focus et press

7.3.4 Performance et Optimisation

- **Rebuilds optimisés** : Utilisation de `const` constructors
- **Lazy loading** : Chargement différé des contenus
- **Mise en cache** : Cache des widgets fréquemment utilisés
- **Responsive** : Adaptation automatique selon l'écran

8 Qualité UI/UX

L'application UqarLive implémente un système de design moderne et cohérent, respectant strictement la charte graphique officielle de l'UQAR tout en offrant une expérience utilisateur intuitive et accessible.

8.1 Règles de Design Appliquées

8.1.1 Respect de la Charte UQAR

L'application respecte scrupuleusement l'identité visuelle officielle de l'Université du Québec à Rimouski.

Palette de Couleurs Officielle
- **Couleur principale** : `#005499` (bleu UQAR) - Utilisée pour les éléments principaux, boutons et accents
- **Couleur accent** : `#00A1E4` (bleu ciel UQAR) - Utilisée pour les liens, actions secondaires et highlights

- **Couleur de fond** : `#F8F9FA` (gris très clair) - Arrière-plan principal des écrans et sections
- **Texte foncé** : `#2C2C2C` (gris foncé) - Texte principal pour une excellente lisibilité
- **Blanc** : `#FFFFFF` - Utilisé pour les contrastes et les éléments sur fond coloré

Typographie Cohérente

- **Système centralisé** : Tous les styles de texte sont définis via `StylesTexteApp`
- **Hiérarchie claire** : Titres, sous-titres, corps de texte et légendes bien définis
- **Accessibilité** : Tailles de police appropriées pour tous les écrans
- **Cohérence** : Même famille de police dans toute l'application

Design "Roundy"

- **Coins arrondis systématiques** : 16px pour les champs, 20px pour les cartes, 24px pour les containers
- **Ombres discrètes** : Utilisation d'ombres avec `alpha: 0.08` et `blur: 24px`
- **Profondeur visuelle** : Élévation subtile pour créer une hiérarchie visuelle
- **Modernité** : Design contemporain respectant les tendances UI/UX actuelles

8.1.2 Gestion des États

L'application gère de manière élégante tous les états possibles de l'interface utilisateur.

État de Chargement

- **Indicateurs visuels** : Spinners, skeletons et progress bars appropriés
- **Feedback immédiat** : Réponse instantanée aux actions utilisateur
- **États de transition** : Animations fluides entre les différents états
- **Gestion des timeouts** : Fallbacks en cas de chargement trop long

État Vide

- **Placeholders informatifs** : Messages clairs expliquant l'absence de contenu
- **Suggestions d'action** : Boutons et liens pour résoudre l'état vide
- **Design attrayant** : Illustrations et icônes pour rendre l'état vide moins austère
- **Contexte approprié** : Messages adaptés au contexte de l'écran

État d'Erreur

- **Messages clairs** : Explications compréhensibles des erreurs
- **Solutions proposées** : Suggestions d'actions pour résoudre le problème
- **Design non-intrusif** : Affichage des erreurs sans perturber l'expérience
- **Logs techniques** : Informations détaillées pour le débogage

État de Succès

- **Confirmations visuelles** : Feedback immédiat des actions réussies
- **Animations de succès** : Transitions et micro-interactions satisfaisantes
- **Prochaines étapes** : Indications sur les actions suivantes possibles
- **Persistance appropriée** : Affichage temporaire des confirmations

8.1.3 Responsive Design

L'application s'adapte parfaitement à toutes les tailles d'écran et orientations.
Adaptation Automatique
- **MediaQuery** : Utilisation systématique pour adapter les dimensions
- **Breakpoints intelligents** : Adaptation selon les standards de l'industrie
- **Layout flexible** : Grilles et colonnes qui s'adaptent automatiquement
- **Navigation adaptative** : Menus et navigation qui s'ajustent à l'écran

Gestion de l'Espace
- **Espacement intelligent** : Marges et paddings qui s'adaptent à l'écran
- **Hiérarchie visuelle** : Importance des éléments préservée sur tous les écrans
- **Contenu prioritaire** : Affichage des informations essentielles en premier
- **Optimisation mobile** : Interface optimisée pour les appareils tactiles

Protection contre les Débordements
- **Gestion des textes longs** : Ellipsis et wrapping appropriés
- **Images responsives** : Adaptation automatique des dimensions
- **Containers flexibles** : Éléments qui s'adaptent au contenu
- **Scroll intelligent** : Navigation fluide sur tous les écrans

8.2 Composants UI Modernes

8.2.1 AppBar Personnalisées

L'application utilise des barres d'application spécialisées selon le contexte utilisateur.
WidgetBarreAppPersonnalisee
- **Navigation intuitive** : Onglets clairement identifiés pour les modules principaux
- **Indicateurs de notifications** : Badges et compteurs pour les alertes
- **Design cohérent** : Intégration parfaite avec le thème UQAR
- **États visuels** : Indication claire de l'onglet actif

WidgetBarreAppNavigationAdmin
- **Fonctions d'administration** : Accès rapide aux outils de gestion
- **Indicateurs de tâches** : Alertes sur les actions en attente
- **Navigation critique** : Accès prioritaire aux fonctions importantes
- **Gestion des privilèges** : Affichage conditionnel selon les droits

8.2.2 Formulaires Intelligents

Les formulaires de l'application offrent une expérience utilisateur moderne et intuitive.
Validation en Temps Réel
- **Feedback immédiat** : Validation des champs au fur et à mesure de la saisie
- **Messages contextuels** : Erreurs affichées près des champs concernés
- **Indicateurs visuels** : Couleurs et icônes pour indiquer l'état des champs
- **Prévention des erreurs** : Validation proactive avant soumission

États de Chargement et Succès
- **Indicateurs de progression** : Barres de progression et spinners
- **Confirmations visuelles** : Messages de succès et animations
- **Transitions fluides** : Changements d'état sans interruption
- **Feedback approprié** : Messages adaptés au contexte de l'action

Design Moderne
- **Coins arrondis** : Champs avec `BorderRadius` de 20px
- **Ombres subtiles** : Profondeur visuelle sans surcharge
- **Espacement cohérent** : Marges et paddings standardisés
- **Couleurs thématiques** : Utilisation de la palette UQAR

8.2.3 Navigation Fluide

L'application offre une navigation intuitive et cohérente entre tous les modules.

TabController pour Écrans Complexes
- **Navigation par onglets** : Accès rapide aux sections principales
- **États persistants** : Préservation de l'état des onglets
- **Transitions fluides** : Changements d'onglet sans rechargement
- **Indicateurs visuels** : Indication claire de l'onglet actif

Navigation entre Sections et Détails
- **Hiérarchie claire** : Navigation logique entre les niveaux
- **Breadcrumbs** : Indication du chemin de navigation
- **Retours intuitifs** : Boutons de retour et navigation arrière
- **Contextes préservés** : Maintien de l'état lors de la navigation

Gestion des Retours et Fallbacks
- **Navigation arrière** : Gestion appropriée du bouton retour
- **États de récupération** : Restauration des données en cas d'erreur
- **Fallbacks intelligents** : Alternatives en cas d'échec
- **Gestion des erreurs** : Messages appropriés et solutions

Transitions Cohérentes
- **Animations uniformes** : Transitions similaires dans toute l'application
- **Timing approprié** : Durées d'animation cohérentes
- **Curves naturelles** : Courbes d'animation fluides et naturelles
- **Performance optimisée** : Animations fluides sur tous les appareils

8.3 Accessibilité et Inclusion

8.3.1 Standards d'Accessibilité

- **Contrastes appropriés** : Respect des ratios de contraste WCAG
- **Tailles de police** : Lisibilité sur tous les appareils
- **Navigation au clavier** : Support complet de la navigation clavier

— **Lecteurs d'écran** : Compatibilité avec les technologies d'assistance

8.3.2 Inclusion et Diversité

— **Langues multiples** : Support du français et de l'anglais
— **Cultures diverses** : Respect des différentes cultures universitaires
— **Capacités variées** : Interface adaptée à tous les niveaux d'expertise
— **Contexte universitaire** : Adaptation aux besoins spécifiques des étudiants

8.4 Performance et Optimisation

8.4.1 Rendu Optimisé

— **Rebuilds minimisés** : Utilisation de `const` constructors et `StatelessWidget`
— **Lazy loading** : Chargement différé des contenus non critiques
— **Cache intelligent** : Mise en cache des données fréquemment consultées
— **Images optimisées** : Compression et formats appropriés

8.4.2 Expérience Utilisateur

— **Réactivité immédiate** : Feedback instantané aux actions utilisateur
— **Chargement progressif** : Affichage progressif du contenu
— **Transitions fluides** : Animations sans interruption
— **Gestion des erreurs** : Récupération gracieuse en cas de problème

9 Gestion des Données

L'application UqarLive implémente un système de gestion des données robuste et sécurisé, utilisant une architecture hybride combinant stockage local et accès distant pour optimiser les performances et la fiabilité.

9.1 Sources de Données

9.1.1 Datasources Locales (15 sources)

L'application utilise un système de stockage local pour garantir l'accès aux données même hors ligne et optimiser les performances.
ActualitesDatasourceLocal
— **Stockage** : Actualités publiées par les associations
— **Fonctionnalités** : CRUD complet avec gestion des priorités et épinglage
— **Optimisations** : Cache des actualités récentes et fréquemment consultées
— **Synchronisation** : Mise à jour automatique lors de la reconnexion

AssociationsDatasourceLocal
- **Stockage** : Informations complètes sur les associations étudiantes
- **Fonctionnalités** : Gestion des membres, contacts et activités
- **Relations** : Liens avec actualités, événements et demandes d'adhésion
- **Validation** : Vérification de l'intégrité des données associatives

LivresDatasourceLocal
- **Stockage** : Catalogue complet des livres du marketplace
- **Fonctionnalités** : Gestion des propriétaires, états et disponibilité
- **Recherche** : Indexation pour recherche rapide par titre, auteur, matière
- **Transactions** : Historique des échanges et ventes

MenusDatasourceLocal
- **Stockage** : Informations détaillées des menus de cantine
- **Fonctionnalités** : Gestion des prix, calories et restrictions alimentaires
- **Temporalité** : Gestion des menus du jour et disponibilité par jour
- **Nutrition** : Informations nutritionnelles et allergènes

SallesDatasourceLocal
- **Stockage** : Données sur les salles de révision et d'étude
- **Fonctionnalités** : Gestion de la capacité, équipements et disponibilité
- **Réservations** : Système de réservation avec gestion des conflits
- **Géolocalisation** : Informations sur les bâtiments et localisation

UtilisateursDatasourceLocal
- **Stockage** : Profils utilisateurs et informations personnelles
- **Fonctionnalités** : Gestion des rôles, privilèges et sessions
- **Sécurité** : Chiffrement des mots de passe et données sensibles
- **Validation** : Vérification des formats et unicité des données

MembresAssociationDatasourceLocal
- **Stockage** : Relations entre utilisateurs et associations
- **Fonctionnalités** : Gestion des rôles, dates d'adhésion et statuts
- **Historique** : Suivi des adhésions et désadhésions
- **Permissions** : Gestion des droits d'accès aux associations

ReservationsSalleDatasourceLocal
- **Stockage** : Réservations de salles par les utilisateurs
- **Fonctionnalités** : Gestion des créneaux, motifs et statuts
- **Validation** : Vérification des conflits et disponibilité
- **Notifications** : Alertes de rappel et confirmation

TransactionsDatasourceLocal
- **Stockage** : Historique complet des transactions de livres
- **Fonctionnalités** : Suivi des achats, ventes et échanges
- **Statuts** : Gestion des états de transaction (en cours, finalisée, annulée)
- **Audit** : Traçabilité complète des opérations

DemandesAdhesionDatasourceLocal
- **Stockage** : Candidatures aux associations étudiantes

— **Fonctionnalités** : Workflow de validation et gestion des statuts
— **Notifications** : Alertes automatiques aux parties prenantes
— **Historique** : Suivi des demandes et décisions

MessagesDatasourceLocal

— **Stockage** : Conversations et messages privés entre utilisateurs
— **Fonctionnalités** : Gestion des contacts et conversations
— **Statuts** : Indicateurs de lecture et nouveaux messages
— **Recherche** : Indexation pour recherche dans l'historique

EvenementsDatasourceLocal

— **Stockage** : Événements organisés par les associations
— **Fonctionnalités** : Gestion des inscriptions, capacité et dates
— **Types** : Catégorisation des événements (conférences, ateliers, sociaux)
— **Notifications** : Alertes de rappel et mises à jour

HorairesDatasourceLocal

— **Stockage** : Horaires d'ouverture de l'université et cantine
— **Fonctionnalités** : Gestion des périodes spéciales et fermetures
— **Synchronisation** : Mise à jour avec le calendrier académique
— **Affichage** : Interface utilisateur adaptée aux horaires

MeteoDatasourceLocal

— **Stockage** : Données météo mises en cache localement
— **Fonctionnalités** : Conditions actuelles et prévisions
— **Mise à jour** : Actualisation automatique des informations
— **Optimisation** : Réduction des appels API externes

9.1.2 Datasources Distantes (1 source actuelle)

L'application intègre des sources de données externes pour enrichir l'expérience utilisateur.

MeteoDatasourceRemote

— **API** : Service météo externe pour conditions actuelles
— **Fonctionnalités** : Récupération des données en temps réel
— **Intégration** : Affichage dans l'interface utilisateur
— **Cache** : Mise en cache locale pour optimiser les performances

Futures Intégrations

— **Systèmes universitaires** : Intégration avec les bases de données UQAR
— **Calendrier académique** : Synchronisation avec les événements officiels
— **Bibliothèque** : Catalogue des livres de la bibliothèque universitaire
— **Services étudiants** : Intégration avec les services administratifs

9.2 Modèles de Données

Chaque entité dispose d'un modèle correspondant qui gère la conversion entre la couche métier et la couche de données.

9.2.1 Structure des Modèles

Les modèles implémentent des méthodes de conversion bidirectionnelles :

- **fromEntity()** : Conversion d'une entité métier vers le modèle de données
- **toEntity()** : Conversion du modèle de données vers l'entité métier
- **copyWith()** : Création de copies modifiées du modèle
- **fromJson()** : Désérialisation depuis le format JSON
- **toJson()** : Sérialisation vers le format JSON

9.2.2 Exemples de Modèles Implémentés

ActualiteModel

- **Attributs** : 10+ propriétés incluant contenu, priorité et métadonnées
- **Validation** : Vérification des formats et contraintes métier
- **Relations** : Liens avec les associations et utilisateurs
- **Optimisations** : Gestion des images et contenu riche

AssociationModel

- **Attributs** : 25+ propriétés incluant contacts et réseaux sociaux
- **Validation** : Vérification des informations de contact
- **Relations** : Liens avec membres, actualités et événements
- **Optimisations** : Gestion des logos et descriptions longues

LivreModel

- **Attributs** : 18 propriétés incluant métadonnées académiques
- **Validation** : Vérification des formats et contraintes
- **Relations** : Liens avec propriétaires et transactions
- **Optimisations** : Gestion des images et mots-clés

MenuModel

- **Attributs** : 12 propriétés incluant informations nutritionnelles
- **Validation** : Vérification des prix et calories
- **Relations** : Liens avec horaires et disponibilité
- **Optimisations** : Gestion des restrictions alimentaires

10 Références

10.1 Bibliographie Principale

Références

[1] Yu, Angela. *The Complete Flutter Development Bootcamp with Dart*. Udemy Course, 2024. `https://www.udemy.com/course/flutter-bootcamp-with-dart/` pages

[2] Martin, Robert C. *Clean Architecture : A Craftsman's Guide to Software Structure and Design*. Prentice Hall, 2017. ISBN : 978-0134494166 pages

10.2 Vidéos YouTube

— **The Ultimate Flutter Tutorial for Beginners - 2025 Full Course** : `https://www.youtube.com/watch?v=3kaGC_DrUnw` *Guide complet d Flutter*
— **Flutter Course for Beginners – 37-hour Cross Platform App Development Tutorial** : `https://www.youtube.com/watch?v=VPvVD8t02U8` *guide commplet pour debutant*
— **F**lutter Clean Architecture - Learn By A Project — Full Beginner's Tutorial `https://www.youtube.com/watch?v=VPvVD8t02U8` *Meilleures pratiques pour appliquer l'architecture clean dans flutter dans Flutter*

10.3 Outils et Ressources Numériques

— **OpenAI ChatGPT** : Assistant IA pour le développement et la résolution de problèmes `https://chat.openai.com/`

[3] Ismail, Chris. *Architecture clean*. Cours de Génie Logiciel II, Département de Mathématiques, Informatique et Génie, Université du Québec à Rimouski (UQAR), 2024. pages