Projet

- Eco-Mobil
  - Présentation du projet Eco-Mobil
- Examen
  - Site Web + API
  - Application Mobile

Eco-Mobil

Présentation du projet Eco-Mobil

Contexte du projet

Eco-Mobil est une filiale créée par un acteur mondial de la location de véhicules pour proposer une offre de « mobilité verte » sur la région Auvergne-Rhône-Alpes, axe Annecy–Grenoble–Valence. Après un déploiement initial couronné de succès (4 agences opérationnelles et un atelier central à Meylan), l'ambition est d'étendre l'activité sur l'axe Saint-Étienne–Bourg-en-Bresse et de déménager la direction à Bron, avec un nouveau site plus vaste intégrant bureaux, entrepôt et ateliers .

Objectifs

1. **Offrir un service de location multimodal** (vélos, VTT, trottinettes, gyropodes, hoverboards, skateboards) en combinant :
   - Un **site web unique** pour la réservation publique.
   - Une **application mobile client** pour le suivi des réservations.
2. **Gérer l'intégralité du cycle de vie des véhicules** :
   - Location (réservation, paiement, emargement, restitution).
   - Maintenance (flux agence ↔ atelier, suivi des interventions).
3. **Fournir un outil de pilotage décisionnel** pour la direction : indicateurs de fréquentation, d'usage par type de véhicule, et de chiffre d'affaires consolidé ou par agence.
4. **Concevoir et déployer l'infrastructure** pour héberger les applications (« clé en main » interne), en assurant sécurité, évolutivité et haute disponibilité (≥ 95 %) .

Périmètre fonctionnel

- **Site web de réservation**
  - Création de compte (email + mot de passe fort ANSSI).

    - Sélection d’agence, de type et nombre de véhicules, date/heure et durée (1 h à N jours).
    - Gestion de l’empreinte CB via prestataire externe.
    - Notification email avec numéro de réservation.
    - Module d’annulation par numéro de réservation.
- **Application mobile client (iOS/Android)**
    - Connexion par email + numéro de réservation.
    - Affichage détaillé de la réservation et bouton « incident » (panne, vol, sinistre).
    - Pré-notion d’évolution hors périmètre initial (gestion des incidents).
- **Logiciel Atelier**
    - Flux logistique agences ↔ atelier (sortie/entrée maintenance).
    - Qualification et suivi des interventions (réparation sur place ou constructeur, assignation technicien, états d’avancement).
    - Gestion des anomalies (véhicule manquant, volé/détruit).
- **Application de pilotage direction**
    - Vues quantitatives et financières par agence et consolidées :
        - Nombre de locations (valeur, % par durée).
        - Répartition par type de véhicule.
        - Chiffre d’affaires global et par type.
        - Taux de « sortie location » par parc pour anticiper la maintenance.

Architecture du Système d’Information

1. **Front-end Web** : site de réservation en responsive design.
2. **Back-end unifié** (API REST) pour gérer :
    - Authentification/Sessions.
    - Gestion des réservations et disponibilités.
    - Suivi des interventions atelier.
    - Tableau de bord direction.
3. **Base de données centrale** (PostgreSQL), répliquée entre sites pour résilience.
4. **Mobile Application** : client léger communiquant avec l’API pour suivi de réservation et signalement d’incident.
5. **Infrastructure** :
    - Hébergement interne (VMs / conteneurs) sur deux sites (Bron & Meylan), interconnectés en VPN.
    - Supervision et sauvegardes automatisées.
    - Plan de reprise d’activité et sécurité renforcée (firewall, IDS/IPS, chiffrement).

Technologies et outils

- **Langages** : Typescript + React pour le backend et framework NextJS pour le front.
- **BDD** : PostgreSQL; Redis pour cache.
- **Infrastructure** : Ferme de serveur de Gab et Nahia.
- **Sécurité** : conformité ANSSI pour mots de passe, chiffrement TLS.

Équipe et organisation

- **Côté SLAM**, développement full-stack :
  - 3 développeurs polyvalents (Web & Mobile).
- **Pas de sous-groupes** : chaque membre contribue sur l'ensemble des périmètres (réservation, atelier, pilotage).
- **Méthode** : Agile Scrum, sprints de 5 semaines, revue et démo en fin de sprint.

Vous pouvez retrouver le cahier des charges ici.

Le code source du projet est disponible sur GitHub :

https://github.com/qoyri/v0-eco-mobile

Le site du projet Eco Mobil est accessible à l'adresse suivante :

https://eco.qoyri.fr

user: admin@example.com

mdp: password123

(Hébergé sur la ferme de serveurs, sur la VM du groupe de Nahia et Gabriel.)

Découvrez ici la technologie que nous avons utilisée pour mettre en place le système d'accès externe au site.

Pour la partie création de l'application mobile nous avons utiliser CapacitorJS pour exporter et compiler notre site web en version application mobile. Vous pouvez retrouver le tutoriel ici et retrouver l'APK de l'application téléchargeable ici.

Examen

Examen

Site Web + API

Documentation du Portail Professeurs

Table des matières

1. Introduction

Le Portail Professeurs est une application web conçue pour aider les enseignants à gérer efficacement leurs classes, les absences des élèves, les réservations de salles, et à communiquer avec les autres membres de l'établissement. Cette plateforme centralisée offre une interface intuitive et responsive qui s'adapte à tous les appareils.

Objectifs du projet

- Simplifier la gestion administrative pour les enseignants
- Améliorer le suivi des absences des élèves
- Faciliter la réservation des salles

- Offrir des outils de communication entre les membres de l'équipe éducative
- Fournir des statistiques et des rapports sur les performances et l'assiduité

2. Architecture technique

Frontend

- **Framework** : Next.js 14 (App Router)
- **Langage** : TypeScript
- **UI/UX** : Tailwind CSS avec shadcn/ui
- **État client** : React Hooks
- **Requêtes API** : Fetch API

Backend

- **Framework** : ASP.NET Core 9
- **Langage** : C#
- **Architecture** : API RESTful
- **Authentification** : JWT (JSON Web Tokens)
- **WebSockets** : Pour la messagerie en temps réel

Base de données

- **SGBD** : MySQL
- **ORM** : Entity Framework Core

3. Structure du projet

Frontend (Next.js)

```
/app                # Dossier principal des routes (App Router)
  /absences           # Gestion des absences
  /classes            # Gestion des classes
  /login              # Page de connexion
  /messages           # Système de messagerie
```

```
  /notifications         # Centre de notifications
  /parametres            # Paramètres utilisateur
  /points                # Système de points
  /rapports              # Génération de rapports
  /reservations          # Réservation de salles
  /salles                # Gestion des salles
  /statistiques          # Tableaux de bord et statistiques
  /layout.tsx            # Layout principal
  /globals.css           # Styles globaux
  /page.tsx              # Page d'accueil

/components              # Composants réutilisables
  /ui                    # Composants UI (shadcn)
  /header.tsx            # En-tête de l'application
  /sidebar.tsx           # Barre latérale de navigation
  /auth-guard.tsx        # Protection des routes authentifiées

/lib                     # Services et utilitaires
  /api-client.ts         # Client API centralisé
  /auth-service.ts       # Service d'authentification
  /absence-service.ts    # Service de gestion des absences
  /class-service.ts      # Service de gestion des classes
  /message-service.ts    # Service de messagerie
  /points-service.ts     # Service de gestion des points
  /room-service.ts       # Service de gestion des salles
  /settings-service.ts   # Service de gestion des paramètres

/public                  # Fichiers statiques
```

Backend (ASP.NET Core)

```
/Controllers             # Contrôleurs API
  /AuthController.cs     # Authentification
  /Teacher               # Contrôleurs spécifiques aux enseignants
    /TeacherAbsenceController.cs
    /TeacherClassController.cs
    /TeacherProfileController.cs
    /TeacherRoomController.cs
    /TeacherSettingsController.cs
```

```
    /TeacherStatController.cs
  /Message              # Contrôleurs de messagerie
    /MessageController.cs
  /Points               # Contrôleurs de gestion des points
    /PointsController.cs

/Models                 # Modèles de données
  /Absence.cs
  /Class.cs
  /Message.cs
  /Notification.cs
  /PointsConfig.cs
  /PointsHistory.cs
  /Reservation.cs
  /Room.cs
  /Schedule.cs
  /Settings.cs
  /Student.cs
  /StudentPoints.cs
  /Teacher.cs
  /User.cs
  /GestionAbsencesContext.cs  # Contexte EF Core

/DTO                    # Objets de transfert de données
  /AbsenceDTO.cs
  /AuthDTO.cs
  /ClassDTO.cs
  /Message
    /MessageDTO.cs
    /ConversationDTO.cs
  /Points
    /StudentPointsDTO.cs
    /PointsHistoryDTO.cs
  /Teacher
    /TeacherProfileDTO.cs

/Services               # Services métier
  /Teacher
    /TeacherAbsenceService.cs
    /TeacherClassService.cs
```

```
    /TeacherProfileService.cs
    /TeacherRoomService.cs
    /TeacherSettingsService.cs
    /TeacherStatService.cs
  /Message
    /MessageService.cs
  /Points
    /PointsService.cs

/WebSocket              # Gestion des WebSockets
  /WebSocketHandler.cs
  /WebSocketMiddleware.cs
  /WebSocketExtensions.cs

/Utils                  # Utilitaires
  /DateOnlyJsonConverter.cs
  /DateUtils.cs

/Middleware             # Middlewares personnalisés
  /BearerPrefixMiddleware.cs
  /RequestResponseLoggingMiddleware.cs

/Scripts                # Scripts SQL
```

4. Fonctionnalités principales

Authentification et sécurité

- Connexion sécurisée avec JWT
- Protection des routes authentifiées
- Gestion des rôles (professeur, admin)
- Déconnexion automatique après inactivité

Gestion des classes

- Liste des classes assignées à l'enseignant
- Détails de chaque classe avec liste des élèves
- Statistiques de présence par classe

- Recherche et filtrage des classes

Gestion des absences

- Saisie des absences par classe ou par élève
- Historique des absences
- Justification des absences
- Statistiques d'assiduité
- Exportation des données d'absence

Système de points

- Attribution de points aux élèves
- Classement des élèves par points
- Historique des points attribués
- Configuration des règles d'attribution

Réservation de salles

- Consultation des salles disponibles
- Réservation de créneaux horaires
- Gestion des caractéristiques des salles
- Calendrier des réservations

Messagerie

- Communication en temps réel via WebSockets
- Conversations privées entre utilisateurs
- Notifications de nouveaux messages
- Statut des messages (envoyé, livré, lu)

Rapports et statistiques

- Génération de rapports personnalisés
- Tableaux de bord avec indicateurs clés
- Graphiques d'assiduité et de performance
- Exportation des rapports en différents formats

Paramètres utilisateur

- Personnalisation du profil
- Choix du thème (clair/sombre)
- Préférences de notification
- Gestion du mot de passe

5. Guide d'utilisation

Connexion au portail

1. Accédez à la page de connexion
2. Entrez les identifiants suivants :
   - Email : `teacher@teacher.fr`
   - Mot de passe : `admin123`
3. Cliquez sur "Se connecter"

Navigation dans l'application

- Utilisez la barre latérale pour accéder aux différentes sections
- Le menu supérieur contient les notifications et l'accès au profil
- Le tableau de bord principal affiche un résumé des informations importantes

Gestion des classes

1. Accédez à la section "Classes" depuis la barre latérale
2. Consultez la liste de vos classes
3. Cliquez sur une classe pour voir les détails et la liste des élèves
4. Utilisez la barre de recherche pour filtrer les classes

Saisie des absences

1. Accédez à la section "Absences" depuis la barre latérale
2. Cliquez sur "Saisir des absences"
3. Sélectionnez la classe concernée
4. Cochez les élèves absents
5. Renseignez la date et le motif si connu
6. Validez la saisie

Réservation d'une salle

1. Accédez à la section "Salles" depuis la barre latérale
2. Consultez les salles disponibles
3. Cliquez sur "Nouvelle réservation"
4. Sélectionnez la salle, la date et l'horaire
5. Indiquez le motif de la réservation
6. Validez la réservation

Utilisation du système de points

1. Accédez à la section "Points" depuis la barre latérale
2. Consultez le classement des élèves
3. Pour attribuer des points :
   - Accédez à la fiche d'un élève
   - Cliquez sur "Ajouter des points"
   - Saisissez le nombre de points et le motif
   - Validez l'attribution

Envoi d'un message

1. Accédez à la section "Messages" depuis la barre latérale
2. Cliquez sur "Nouveau message"
3. Sélectionnez le destinataire
4. Rédigez votre message
5. Envoyez le message

Génération d'un rapport

1. Accédez à la section "Rapports" depuis la barre latérale
2. Sélectionnez le type de rapport
3. Définissez les paramètres (période, classe, etc.)
4. Générez le rapport
5. Consultez-le en ligne ou exportez-le

Modification des paramètres

1. Accédez à la section "Paramètres" depuis la barre latérale
2. Modifiez votre profil, thème ou préférences
3. Enregistrez les modifications

6. API et endpoints

Authentification

- `POST /api/auth/login` - Connexion utilisateur
- `GET /api/auth/test` - Test d'authentification
- `GET /api/auth/test-professeur` - Test d'autorisation professeur

Gestion des classes

- `GET /api/teacher/classes` - Liste des classes
- `GET /api/teacher/classes/{id}` - Détails d'une classe

Gestion des absences

- `GET /api/teacher/absences` - Liste des absences
- `GET /api/teacher/absences/{id}` - Détails d'une absence
- `POST /api/teacher/absences` - Création d'une absence
- `PUT /api/teacher/absences/{id}` - Mise à jour d'une absence
- `DELETE /api/teacher/absences/{id}` - Suppression d'une absence

Gestion des salles

- `GET /api/teacher/rooms` - Liste des salles
- `GET /api/teacher/rooms/{id}` - Détails d'une salle
- `GET /api/teacher/rooms/{id}/reservations` - Réservations d'une salle

Réservations

- `GET /api/teacher/reservations` - Liste des réservations
- `GET /api/teacher/reservations/{id}` - Détails d'une réservation
- `POST /api/teacher/reservations` - Création d'une réservation
- `PUT /api/teacher/reservations/{id}` - Mise à jour d'une réservation
- `DELETE /api/teacher/reservations/{id}` - Suppression d'une réservation

Messagerie

- `GET /api/messages` - Liste des conversations
- `GET /api/messages/{userId}` - Messages avec un utilisateur
- `POST /api/messages` - Envoi d'un message
- `PUT /api/messages/{id}/read` - Marquer un message comme lu
- `WebSocket /ws` - Communication en temps réel

Système de points

- `GET /api/points/students` - Points des élèves
- `GET /api/points/students/{id}` - Points d'un élève
- `GET /api/points/ranking` - Classement des élèves
- `POST /api/points/add` - Attribution de points
- `GET /api/points/history/{studentId}` - Historique des points

Profil et paramètres

- `GET /api/teacher/profile` - Profil de l'enseignant
- `PUT /api/teacher/profile` - Mise à jour du profil
- `GET /api/teacher/settings` - Paramètres utilisateur
- `PUT /api/teacher/settings` - Mise à jour des paramètres

7. Base de données

Schéma de la base de données

La base de données est organisée autour des entités principales suivantes :

- **Users** : Utilisateurs du système (enseignants, administrateurs)
- **Teachers** : Informations spécifiques aux enseignants
- **Students** : Informations sur les élèves
- **Classes** : Classes et leurs associations avec les enseignants
- **Absences** : Enregistrements des absences des élèves
- **Rooms** : Salles disponibles pour réservation
- **Reservations** : Réservations des salles
- **Messages** : Messages échangés entre utilisateurs
- **Notifications** : Notifications système
- **Settings** : Paramètres utilisateur
- **StudentPoints** : Points attribués aux élèves
- **PointsHistory** : Historique des attributions de points
- **PointsConfig** : Configuration du système de points

Relations principales

- Un enseignant peut avoir plusieurs classes
- Une classe peut avoir plusieurs élèves
- Un élève peut avoir plusieurs absences
- Un utilisateur peut avoir plusieurs réservations
- Un utilisateur peut envoyer/recevoir plusieurs messages
- Un élève peut avoir plusieurs entrées dans l'historique des points

8. Sécurité

Authentification

- Utilisation de JWT (JSON Web Tokens) pour l'authentification
- Stockage sécurisé des tokens côté client
- Expiration automatique des tokens
- Vérification des rôles pour l'accès aux ressources

Protection des données

- Hachage des mots de passe avec SHA-256
- Validation des entrées côté client et serveur
- Protection contre les attaques CSRF
- Utilisation de HTTPS pour toutes les communications

Autorisations

- Système de rôles (professeur, admin)
- Vérification des autorisations pour chaque action
- Isolation des données par utilisateur
- Journalisation des actions sensibles

9. Déploiement

Prérequis

- Node.js 18+ pour le frontend
- .NET 9 SDK pour le backend
- MySQL 8+ pour la base de données
- Serveur web (Nginx recommandé)
- Proxy Reverse Caddy

Étapes de déploiement

1. **Base de données**
   - Créer une base de données MySQL
   - Exécuter les scripts de migration
2. **Backend**
   - Compiler l'application ASP.NET Core
   - Configurer les variables d'environnement
   - Déployer sur un serveur compatible .NET
3. **Frontend**
   - Construire l'application Next.js (`next build`)
   - Déployer les fichiers statiques
   - Configurer le serveur web pour servir l'application

Configuration

- Fichier `.env` pour les variables d'environnement frontend
- Fichier `appsettings.json` pour la configuration backend
- Configuration de la base de données dans `GestionAbsencesContext.cs`

10. Dépannage

Problèmes courants

Erreur de connexion

- Vérifier les identifiants (teacher@teacher.fr / admin123)
- Vérifier que le serveur backend est accessible
- Vérifier les logs du serveur pour des erreurs d'authentification

Données non affichées

- Vérifier la connexion au serveur backend
- Vérifier les requêtes dans la console du navigateur

- Vérifier que l'utilisateur a les permissions nécessaires

Erreurs WebSocket

- Vérifier que le serveur WebSocket est en cours d'exécution
- Vérifier la configuration du proxy si applicable
- Vérifier les logs du serveur pour des erreurs de connexion

Problèmes de performance

- Vérifier la charge du serveur
- Optimiser les requêtes à la base de données
- Vérifier les index de la base de données

Logs et monitoring

- Les logs frontend sont disponibles dans la console du navigateur
- Les logs backend sont stockés dans le dossier `/logs`
- Utiliser les outils de développement du navigateur pour déboguer les problèmes frontend
- Utiliser les outils de débogage de Visual Studio pour le backend

Examen

Application Mobile

Application Mobile de Gestion des Absences

Présentation

Cette application mobile Android permet aux enseignants de gérer les absences des élèves, consulter leurs classes, et accéder à diverses statistiques. Elle est conçue pour fonctionner en complément d'une API REST qui gère les données côté serveur.

Informations de connexion

Pour tester l'application, utilisez les identifiants suivants :

- **Email/Username** : teacher@teacher.fr
- **Mot de passe** : admin123

Architecture de l'application

L'application est développée selon le pattern MVVM (Model-View-ViewModel) et utilise les composants d'architecture Android :

- **Model** : Représente les données et la logique métier
- **View** : Activités et fragments qui affichent l'interface utilisateur
- **ViewModel** : Gère les données liées à l'UI et les interactions avec le modèle

- **Repository** : Couche intermédiaire qui gère les sources de données (API, base de données locale)

Structure des packages

```
com.example.exam/
├── adapters/          # Adaptateurs pour les RecyclerViews
├── api/               # Services API et configuration Retrofit
├── models/            # Classes de modèles de données
├── repository/        # Repositories pour accéder aux données
├── ui/                # Activités et fragments organisés par fonctionnalité
│   ├── absences/      # Gestion des absences
│   ├── classes/       # Gestion des classes
│   ├── home/          # Écran d'accueil et tableau de bord
│   └── ...
└── utils/             # Classes utilitaires
```

Fonctionnalités principales

1. Authentification

- Connexion avec email/mot de passe
- Stockage sécurisé du token d'authentification
- Déconnexion

2. Tableau de bord (Dashboard)

- Vue d'ensemble des statistiques (nombre d'élèves, classes, absences)
- Graphique des absences par mois
- Liste des absences récentes
- Fonctionnalité de rafraîchissement (swipe-to-refresh)

3. Gestion des classes

- Liste des classes de l'enseignant
- Détails d'une classe avec liste des élèves
- Statistiques par classe (taux d'absence, nombre d'élèves)

4. Gestion des absences

- Consultation des absences
- Saisie d'une nouvelle absence
- Possibilité de joindre un justificatif (document)
- Filtrage des absences

Flux d'utilisation

Connexion et navigation

1. L'utilisateur lance l'application et arrive sur l'écran de connexion
2. Après authentification réussie, l'utilisateur est redirigé vers le tableau de bord
3. La navigation entre les différentes sections se fait via le menu latéral (Navigation Drawer)

Consultation des classes

1. L'utilisateur accède à la section "Classes" depuis le menu
2. La liste des classes s'affiche avec des informations de base
3. En cliquant sur une classe, l'utilisateur accède aux détails et à la liste des élèves
4. Depuis cette vue, l'utilisateur peut saisir une absence ou générer un rapport

Saisie d'une absence

1. Depuis la liste des classes ou les détails d'une classe, l'utilisateur peut cliquer sur "Saisir une absence"
2. Un formulaire s'affiche permettant de sélectionner l'élève, la date, le statut et la raison
3. L'utilisateur peut optionnellement joindre un document justificatif

4. Après validation, l'absence est enregistrée et l'utilisateur est redirigé vers l'écran précédent

Technologies utilisées

Bibliothèques principales

- **AndroidX** : Composants d'architecture Android (ViewModel, LiveData)
- **Material Design** : Composants UI modernes et cohérents
- **Retrofit** : Client HTTP pour les appels API
- **OkHttp** : Client HTTP sous-jacent avec intercepteurs pour logging
- **Gson** : Sérialisation/désérialisation JSON
- **MPAndroidChart** : Visualisation de données sous forme de graphiques
- **Lottie** : Animations vectorielles
- **Shimmer** : Effets de chargement
- **SwipeRefreshLayout** : Fonctionnalité de rafraîchissement par glissement

Gestion des données

- **SharedPreferences** : Stockage des informations de session (token, identifiants)
- **LiveData** : Notification des changements de données aux composants UI
- **ViewModel** : Persistance des données lors des changements de configuration

Détails techniques

Authentification et sécurité

L'application utilise un système d'authentification par token JWT :

1. L'utilisateur s'authentifie avec ses identifiants
2. Le serveur renvoie un token JWT
3. Ce token est stocké dans les SharedPreferences
4. Un intercepteur OkHttp ajoute automatiquement le token à chaque requête API

Gestion des erreurs

L'application gère différents types d'erreurs :

- Erreurs réseau (pas de connexion)
- Erreurs d'authentification (token expiré, identifiants incorrects)
- Erreurs serveur (500, 404, etc.)

Des écrans d'erreur spécifiques sont affichés avec possibilité de réessayer.

Optimisations UI/UX

- **Shimmer Effect** : Affichage d'un effet de chargement pendant les requêtes API
- **Animations** : Transitions fluides entre les écrans
- **Swipe-to-refresh** : Actualisation des données par glissement vers le bas
- **Mise en cache** : Certaines données sont mises en cache pour améliorer les performances

API et endpoints

L'application communique avec une API REST hébergée à l'adresse `https://api.slam.qoyri.fr/api/`.

Principaux endpoints utilisés

- **Authentification** : `/auth/login`
- **Dashboard** : `/teacher/dashboard`
- **Classes** : `/teacher/classes`
- **Détails d'une classe** : `/teacher/classes/{id}`
- **Absences** : `/teacher/absences`
- **Création d'absence** : `/teacher/absences/create`

Installation et déploiement

Prérequis

- Android Studio 4.0+
- JDK 11+
- SDK Android 33+ (Android 13)

Compilation et installation

1. Cloner le dépôt
2. Ouvrir le projet dans Android Studio
3. Synchroniser avec Gradle
4. Compiler et installer sur un appareil ou émulateur

Configuration

L'URL de l'API est configurée dans la classe `ApiClient.java`. Pour changer l'environnement (développement, production), modifiez la constante `BASE_URL`.

Dépannage

Problèmes courants

1. **Erreur de connexion** : Vérifiez votre connexion internet et que l'API est accessible
2. **Erreur d'authentification** : Vérifiez vos identifiants ou réessayez de vous connecter
3. **Données non affichées** : Utilisez la fonction de rafraîchissement (swipe-to-refresh)

Évolutions futures

- Implémentation du mode hors ligne avec synchronisation
- Ajout de notifications push pour les nouvelles absences
- Intégration de la biométrie pour l'authentification
- Support du mode sombre