📅 Gestion d'Emploi du Temps Universitaire

Projet ICT 203 - Année Académique 2025/2026

Délai: 14 jours **Équipe: 6 étudiants**

1. Contexte du Projet

🎯 Objectif

Développer une application de gestion d'emploi du temps pour un établissement universitaire permettant la planification, la modification et la consultation des emplois du temps par différents acteurs (enseignants, administrateurs, étudiants).

📦 Stack Technique Imposée

⚛️ Frontend

React.js avec Material-UI pour une interface moderne et responsive

🟢 Backend

Node.js avec Express.js pour l'API REST

🗄️ Base de Données

SQLite avec triggers et procédures stockées pour la logique métier

💻 Desktop

Electron pour packaging .exe (Windows) et .AppImage (Linux)

⏰ **Délai de réalisation:** 14 jours à compter de la publication du projet

👥 **Équipe:** 6 étudiants

2. Contraintes Principales

- **Logique métier dans la BD:** Utilisation obligatoire de triggers et procédures SQLite pour les contraintes
- **Optimisation automatique:** Attribution des salles selon la capacité et l'effectif
- **Traçabilité complète:** Historique de toutes les modifications
- **Alertes système:** Notification automatique des emplois du temps complets
- **Multi-plateforme:** Exécutables pour Windows et Linux

3. Livrables Attendus

1. **Cahier de conception et d'implémentation** avec spécifications et consignes d'installation
2. **Exécutable avec code source** (.exe + .AppImage + sources)
3. **Démonstration ICT-L2:** Emploi du temps du semestre 1 2025/2026 avec 2 plages horaires
4. **Documentation complète:** Installation, utilisation, conception technique

Acteurs du Système

👨‍🏫 Enseignant

Permissions:

- Soumettre ses préférences de plages horaires
- Modifier sa programmation (pendant une période définie)
- Consulter son emploi du temps finalisé
- Visualiser l'historique de ses modifications

Contraintes:

- Période de modification limitée dans le temps
- Ne peut modifier que ses propres désidératas
- Consultation uniquement après validation admin

👨‍💼 Administrateur

Permissions:

- Paramétrer l'application (salles, classes, effectifs, années académiques)
- Modifier/supprimer les choix des enseignants
- Effectuer les arbitrages nécessaires
- Recevoir des alertes pour les emplois du temps complets
- Visualiser toutes les modifications (traçabilité complète)
- Créer et valider les emplois du temps
- Gérer les utilisateurs du système

👁️ Consultation (Vue Publique)

Accès:

- Afficher l'emploi du temps selon différents critères

- Filtrer par département, filière, classe
- Vues multiples : par temps, groupes, espaces, enseignants
- Export PDF des emplois du temps

Matrice des Permissions

Fonctionnalité	Enseignant	Administrateur	Consultation
Consulter EDT	✅ (personnel)	✅ (tous)	✅ (tous)
Soumettre désidératas	✅	✅	❌
Modifier désidératas	✅ (période limitée)	✅	❌
Créer/Modifier EDT	❌	✅	❌
Paramétrage système	❌	✅	❌
Voir historique	✅ (personnel)	✅ (complet)	❌
Recevoir alertes	❌	✅	❌

Fonctionnalités Détaillées

🗓️ Gestion des Emplois du Temps

Affichage Multi-vues

L'application doit permettre l'affichage selon:

- **Vue temporelle:** Semestre, Semaine, Jour
- **Vue par groupes:** Classe spécifique, Groupes d'étudiants
- **Vue par espace:** Par salle
- **Vue par enseignant:** Emploi du temps individuel

> ### 📊 Informations Obligatoires par Créneaux
>
> Chaque créneau doit afficher:
>
> - ⏰ Plage horaire réservée
> - 📚 Code de l'UE (Unité d'Enseignement)
> - 👨‍🏫 Nom de l'enseignant
> - 🏫 Salle assignée
> - 📅 Date de passage de l'UE

👨‍🏫 Interface Enseignant

Soumission des Désidératas

- Formulaire de sélection des plages horaires préférées
- Choix des jours de la semaine
- Système de préférences (préféré, acceptable, à éviter)
- Validation temporaire (période de modification active)

Modification de la Programmation

- Possibilité de modifier ses choix
- Limitation temporelle (période définie par l'admin)
- Historique des modifications personnelles
- Notification des changements effectués par l'admin

👨‍💼 Interface Administrateur

Paramétrage du Système

🏫 Gestion des Salles

Création, modification, suppression de salles avec définition de capacité

👥 Gestion des Classes

Effectif par classe, semestre, année académique, filière

📚 Gestion des UE

Code, intitulé, enseignant assigné, volume horaire

⏰ Gestion des Périodes

Semestres, période de modification, plages horaires

Arbitrage et Modification

- Modifier les choix des enseignants
- Supprimer des créneaux
- Réassigner des salles/horaires
- Traçabilité complète (qui a modifié quoi et quand)

🔔 Système d'Alertes

- Alerte automatique quand un emploi du temps est complet
- Notification des conflits (salle, enseignant, horaire)
- Dashboard des emplois du temps en cours
- Suivi de progression par classe/semestre

🤖 Optimisation Automatique

Attribution des Salles (via Triggers SQLite)

- Vérifier l'effectif de la classe
- Proposer uniquement les salles disponibles avec capacité suffisante
- Éviter les doubles réservations de salles

Détection des Conflits (via Triggers SQLite)

- Un enseignant ne peut être à deux endroits en même temps
- Une salle ne peut accueillir qu'un cours à la fois

- Vérification de la cohérence des horaires

🗄️ Modèle de Base de Données Complet

Schéma des Tables Principales

1. Départements

```
CREATE TABLE departements (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  code VARCHAR(10) UNIQUE NOT NULL,
  nom VARCHAR(100) NOT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

2. Filières

```
CREATE TABLE filieres (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  code VARCHAR(10) UNIQUE NOT NULL,
  nom VARCHAR(100) NOT NULL,
  departement_id INTEGER NOT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (departement_id) REFERENCES departements(id) ON DELETE CASCADE
);
```

3. Classes

```
CREATE TABLE classes (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  code VARCHAR(20) UNIQUE NOT NULL,
  nom VARCHAR(100) NOT NULL,
  niveau VARCHAR(10) NOT NULL, -- L1, L2, L3, M1, M2
  effectif INTEGER NOT NULL CHECK (effectif > 0),
  filiere_id INTEGER NOT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (filiere_id) REFERENCES filieres(id) ON DELETE CASCADE
);

-- Index pour optimisation
```

```
CREATE INDEX idx_classes_filiere ON classes(filiere_id);
CREATE INDEX idx_classes_niveau ON classes(niveau);
```

4. Années Académiques

```
CREATE TABLE annees_academiques (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  libelle VARCHAR(20) UNIQUE NOT NULL, -- 2025/2026
  date_debut DATE NOT NULL,
  date_fin DATE NOT NULL,
  active BOOLEAN DEFAULT 0,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  CHECK (date_fin > date_debut)
);

-- Trigger: Une seule année académique active à la fois
CREATE TRIGGER single_active_year
BEFORE UPDATE ON annees_academiques
WHEN NEW.active = 1
BEGIN
  UPDATE annees_academiques SET active = 0 WHERE id != NEW.id;
END;
```

5. Semestres

```
CREATE TABLE semestres (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  numero INTEGER NOT NULL CHECK (numero IN (1, 2)),
  annee_academique_id INTEGER NOT NULL,
  date_debut DATE NOT NULL,
  date_fin DATE NOT NULL,
  periode_modification_debut DATE,
  periode_modification_fin DATE,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (annee_academique_id) REFERENCES annees_academiques(id) ON DELETE CAS
  CHECK (date_fin > date_debut),
  UNIQUE (numero, annee_academique_id)
);

CREATE INDEX idx_semestres_annee ON semestres(annee_academique_id);
```

6. Salles

```
CREATE TABLE salles (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  code VARCHAR(20) UNIQUE NOT NULL,
  nom VARCHAR(100) NOT NULL,
  capacite INTEGER NOT NULL CHECK (capacite > 0),
  type VARCHAR(50) CHECK (type IN ('Amphi', 'TD', 'TP', 'Labo', 'Autre')),
  equipements TEXT, -- JSON: ['Vidéoprojecteur', 'Ordinateurs', ...]
  disponible BOOLEAN DEFAULT 1,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_salles_capacite ON salles(capacite);
CREATE INDEX idx_salles_type ON salles(type);
```

7. Enseignants

```
CREATE TABLE enseignants (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  matricule VARCHAR(20) UNIQUE NOT NULL,
  nom VARCHAR(100) NOT NULL,
  prenom VARCHAR(100) NOT NULL,
  email VARCHAR(100) UNIQUE NOT NULL,
  telephone VARCHAR(20),
  departement_id INTEGER,
  actif BOOLEAN DEFAULT 1,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (departement_id) REFERENCES departements(id) ON DELETE SET NULL
);

CREATE INDEX idx_enseignants_departement ON enseignants(departement_id);
CREATE INDEX idx_enseignants_actif ON enseignants(actif);
```

8. Unités d'Enseignement (UE)

```
CREATE TABLE ues (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  code VARCHAR(20) UNIQUE NOT NULL,
  intitule VARCHAR(200) NOT NULL,
  volume_horaire INTEGER NOT NULL CHECK (volume_horaire > 0),
  type VARCHAR(20) CHECK (type IN ('CM', 'TD', 'TP', 'CI')),
  semestre_id INTEGER NOT NULL,
  classe_id INTEGER NOT NULL,
  credits INTEGER CHECK (credits > 0),
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (semestre_id) REFERENCES semestres(id) ON DELETE CASCADE,
  FOREIGN KEY (classe_id) REFERENCES classes(id) ON DELETE CASCADE
);
```

```
CREATE INDEX idx_ues_semestre ON ues(semestre_id);
CREATE INDEX idx_ues_classe ON ues(classe_id);
```

9. Plages Horaires

```
CREATE TABLE plages_horaires (
```