# Question 1

## Q :

Peux tu me faire une page pour gérer les horaires de mes employé qui parle avec ma base de données dont ma table horaire qui ressemble à … , je voudrais aller chercher les données et les afficher, ainsi que pouvoir les changer, je dois pouvoir changer l’heure d’arrivée et l’heure de sortie pour chaque jour et avoir un bouton en haut a droite pour revenir en arrière. Je veux un bouton enregistrer qui enregistre dans ma base de donnée mes valeurs

## Rep :

import sqlite3

from PySide6.QtCore import Qt, QTime

from PySide6.QtWidgets import QWidget, QVBoxLayout, QHBoxLayout, QLabel, QLineEdit, QTimeEdit, QPushButton, QComboBox, QGridLayout

from PySide6.QtGui import QIntValidator

from ERP\_data\_base import DatabaseManager

from ERP\_emplacement import Emplacement

class QHoraire(QWidget):

    def \_\_init\_\_(self, parent=None):

            super().\_\_init\_\_(parent)

            # Layout principal (vertical)

            layout = QVBoxLayout(self)

            # Layout pour le bouton de retour (horizontal)

            back\_button\_layout = QHBoxLayout()

            back\_button = QPushButton("<-")

            back\_button.clicked.connect(parent.basculer\_before)

            back\_button\_layout.addWidget(back\_button)

            # Assurez-vous que le bouton est aligné à gauche

            back\_button\_layout.setAlignment(Qt.AlignmentFlag.AlignLeft)

            # Ajouter le layout du bouton de retour en haut

            layout.addLayout(back\_button\_layout)

            # Titre

            layout.addWidget(QLabel("Gestion des Horaires"))

            # Création du formulaire pour les horaires de chaque jour de la semaine

            grid\_layout = QGridLayout()

            self.inputs = {}

            jours = ["Lundi", "Mardi", "Mercredi", "Jeudi", "Vendredi"]

            for i, jour in enumerate(jours):

                # Labels pour chaque jour

                grid\_layout.addWidget(QLabel(jour), i, 0)

                # Champs pour l'heure d'entrée et l'heure de sortie

                heure\_entree = QTimeEdit(self)

                heure\_sortie = QTimeEdit(self)

                # Ajouter les widgets à la grille

                grid\_layout.addWidget(heure\_entree, i, 1)

                grid\_layout.addWidget(heure\_sortie, i, 2)

                # Stocker ces champs dans un dictionnaire

                self.inputs[f'heure\_entree\_{jour.lower()}'] = heure\_entree

                self.inputs[f'heure\_sortie\_{jour.lower()}'] = heure\_sortie

            # Bouton pour enregistrer les horaires

            save\_button = QPushButton("Enregistrer", self)

            save\_button.clicked.connect(self.save\_horaire)

            # Ajouter la grille et le bouton enregistrer

            layout.addLayout(grid\_layout)

            layout.addWidget(save\_button)

    def load\_horaires(self):

        """Récupérer les horaires de l'employé et de la succursale depuis la base de données et les afficher."""

        print("récupération des horaires")

        try:

            db\_manager = DatabaseManager('erp\_database.db')

            # Récupérer l'horaire de l'employé pour cette succursale

            query = """

                SELECT heure\_entree\_lundi, heure\_sortie\_lundi,

                       heure\_entree\_mardi, heure\_sortie\_mardi,

                       heure\_entree\_mercredi, heure\_sortie\_mercredi,

                       heure\_entree\_jeudi, heure\_sortie\_jeudi,

                       heure\_entree\_vendredi, heure\_sortie\_vendredi

                FROM Horaires

                WHERE id\_employe = ?

            """

            print((Emplacement.employeHoraire))

            result = db\_manager.execute\_query(query, (int(Emplacement.employeHoraire),))

            if result:

                horaires = result[0]

                jours = ["lundi", "mardi", "mercredi", "jeudi", "vendredi"]

                # Remplir les champs de l'interface avec les horaires récupérés

                for i, jour in enumerate(jours):

                    # Convertir les chaînes d'heure en objets QTime

                    heure\_entree = QTime.fromString(horaires[i\*2], "HH:mm")

                    heure\_sortie = QTime.fromString(horaires[i\*2 + 1], "HH:mm")

                    # Assigner ces objets QTime aux champs correspondants

                    self.inputs[f'heure\_entree\_{jour}'].setTime(heure\_entree)  # Heure d'entrée

                    self.inputs[f'heure\_sortie\_{jour}'].setTime(heure\_sortie)  # Heure de sortie

            else:

                print("Aucun horaire trouvé pour cet employé dans cette succursale.")

        except sqlite3.Error as e:

            print(f"Une erreur est survenue lors de la récupération des horaires : {e}")

    def save\_horaire(self):

        """Enregistrer les horaires modifiés dans la base de données."""

        # Collecter les horaires saisis

        horaires = {}

        jours = ["lundi", "mardi", "mercredi", "jeudi", "vendredi"]

        for i, jour in enumerate(jours):

            heure\_entree = self.inputs[f'heure\_entree\_{jour}'].time().toString("HH:mm")

            heure\_sortie = self.inputs[f'heure\_sortie\_{jour}'].time().toString("HH:mm")

            horaires[f'heure\_entree\_{jour}'] = heure\_entree

            horaires[f'heure\_sortie\_{jour}'] = heure\_sortie

        # Insérer ou mettre à jour les horaires dans la base de données

        try:

            # Connexion à la base de données

            db\_manager = DatabaseManager('erp\_database.db')

            query = """

                UPDATE Horaires

                SET

                    heure\_entree\_lundi = ?, heure\_sortie\_lundi = ?,

                    heure\_entree\_mardi = ?, heure\_sortie\_mardi = ?,

                    heure\_entree\_mercredi = ?, heure\_sortie\_mercredi = ?,

                    heure\_entree\_jeudi = ?, heure\_sortie\_jeudi = ?,

                    heure\_entree\_vendredi = ?, heure\_sortie\_vendredi = ?,

                    statut = 'actif'

                WHERE id\_employe = ?  -- Critère pour l'employé spécifique

            """

            # Extraire les horaires pour chaque jour de la semaine (assurez-vous que `horaires` contient ces valeurs sous forme de dictionnaire)

            values = (

                horaires['heure\_entree\_lundi'], horaires['heure\_sortie\_lundi'],

                horaires['heure\_entree\_mardi'], horaires['heure\_sortie\_mardi'],

                horaires['heure\_entree\_mercredi'], horaires['heure\_sortie\_mercredi'],

                horaires['heure\_entree\_jeudi'], horaires['heure\_sortie\_jeudi'],

                horaires['heure\_entree\_vendredi'], horaires['heure\_sortie\_vendredi'],

                Emplacement.employeHoraire  # L'ID de l'employé à mettre à jour

            )

            # Exécuter la mise à jour dans la base de données

            db\_manager.execute\_update(query, values)

            print("Les horaires ont été mis à jour avec succès.")

        except sqlite3.Error as e:

            print(f"Une erreur est survenue lors de l'enregistrement des horaires : {e}")

## Conclu :

J’ai utiliser son blueprint de page avec les qtimedit, j’ai du changer l’accès a la base de donnée sinon tout machait quand même bien, je ne savais pas comment utiliser les qtime edit.

Sam

# Question 1

* + Tu peux faire une search bar pour chercher un employé dans une base de donne en utilisant la fonction employee\_dao.get\_employe\_by\_id(code\_employe)

# Réponse 1 :

* # Search box
* search\_layout = QHBoxLayout()
* self.search\_box = QLineEdit(self)
* self.search\_box.setPlaceholderText("Rechercher employé...")
* self.search\_box.textChanged.connect(self.search\_employee)
* search\_button = QPushButton("Rechercher", self)
* search\_button.clicked.connect(self.search\_employee)
* search\_layout.addWidget(self.search\_box)
* search\_layout.addWidget(search\_button)  
    
  def search\_employee(self):
* """Search and display employees matching the search text."""
* search\_text = self.search\_box.text().strip()
* employees = EmployeDAO().get\_employe\_by\_id(search\_text)
* self.update\_employee\_display(employees)

# Nuance

* + J’ai pris la class comme il m’avait envoyé

# Question 2

* + J’avais un erreur pour ouvrir la fenêtre midification donc j demande et il ma propose de:

# Réponse 2:

* + self.db\_manager = DatabaseManager('erp\_database.db')
  + Pour pouvoir accéder plus facilement a la base de donne.
* # Global
* self.vue = parent
* self.employee\_dao = EmployeDAO()
* self.db\_manager = DatabaseManager('erp\_database.db') # Make sure this line exists
* def open\_modify\_dialog(self, row):
* employee\_id = self.employee\_table.item(row, 0).text
* \* FROM Employes WHERE id\_employe = ?"
* result = self.db\_manager.execute\_query(query, (employee\_id,))
* # Step 3: Prepare the employee data (convert tuple to a list for easier access)
* employee\_data = []
* if result:
* employee\_data = list(result[0]) # Convert the tuple into a list (same as you did for succursales)
* # Step 4: Open the modify dialog and pass the employee data
* if employee\_data:
* dialog = AddModifyDialogEmploye(self, mode="Modifier", employee\_data=employee\_data)
* dialog.exec\_()
* else:
* QMessageBox.warning(self, "Erreur", "Aucun employé trouvé avec cet ID.")

# Nuance :

* + Il m’a conseillé d’ajouté self.db\_manager pour accéder facilement a la base de donnes et pouvoir modifier l’employé, donc j’ai implémenté sa suggestion.

# Question 3

* + J’avais un erreur et l’ID de l’employé n’est pas trouvé donc j’ai passé une journée avec chatgpt pour essayer de régler le problème mais ça pas marche. Donc j’ai demandé l’aide à Alek et il a réglé le problème en 15min