ENSAM
ÉCOLE NATIONALE SUPÉRIEURE D'ARTS ET MÉTIERS
UNIVERSITÉ HASSAN II DE CASABLANCA

2023/2024

Programmation Orientée Objet

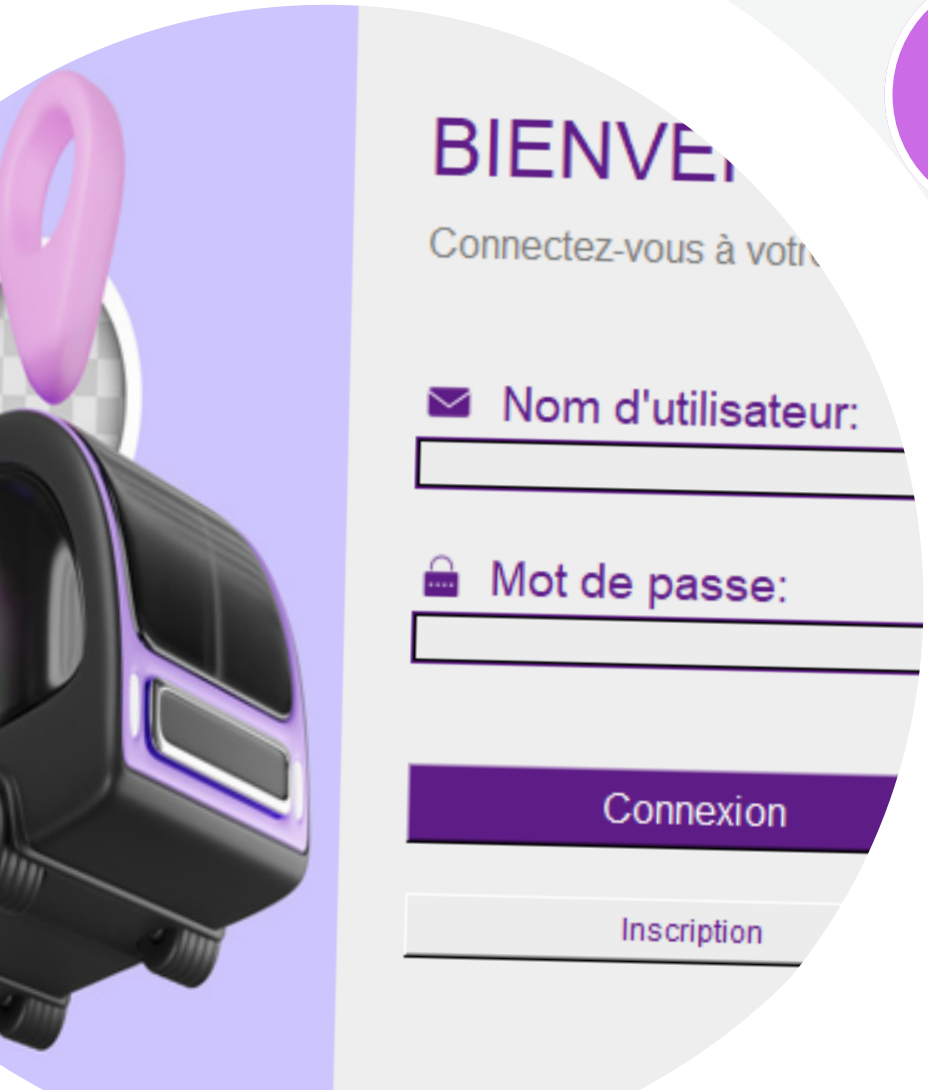
Application de gestion de flotte de véhicules

Réalisée par:

- EL MOHADI Assia
- DOUCH Mariam
- ID OUFKIR Salma

Sous la prévision de :

- Mr. HAIN Mustapha



INTRODUCTION

Pour améliorer l'administration de leurs ressources et fournir un rendement quantifiable de leur gestion interne, les entreprises informatisent et automatisent leurs différents processus administratifs.

Dans ce cadre, nous nous sommes intéressées à la conception et à la mise en place d'une application qui informatise les opérations liées à la gestion de flotte de véhicules des entreprises.

Ce rapport se consacre à la présentation de notre application, élaborée avec les langages de programmation Python, Tkinter pour l'interface graphique, et SQLite3 pour la gestion de la base de données. Cette initiative découle de notre désir d'appliquer les compétences acquises au cours de notre formation en programmation orientée objets et d'autres concepts pertinents.

Nous explorerons en détail les différentes facettes de l'application, mettant en lumière son architecture basée sur des principes de programmation orientée objets.

Pour vous connecter et essayer notre application de gestion de flotte de véhicules, vous pouvez utiliser votre nom d'utilisateur et mot de passe :

id	nom_utilisateur	mot_de_passe
Filtre	Filtre	Filtre
3	HAIN Mustapha	HAIN

INTERFACES:

1

Interface de connexion/inscription :

Cette interface comporte deux champs de saisie pour le nom d'utilisateur et le mot de passe, permettant ainsi d'effectuer la connexion. De plus, elle est équipée de deux boutons, «Connexion » et «Inscription », servant à valider la connexion et à effectuer l'inscription dans le cas où l'utilisateur n'a pas encore de compte, respectivement.

Lorsque l'utilisateur clique sur le bouton «Connexion » sans remplir tous les champs requis, un message d'erreur s'affiche dans une boîte de dialogue, indiquant : "Erreur, veuillez remplir tous les champs."

En cas de tentative de connexion avec un nom d'utilisateur ou un mot de passe incorrect, un message d'erreur spécifique apparaît : "Erreur, nom d'utilisateur ou mot de passe incorrect."

En appuyant sur le bouton “Inscription“, l'utilisateur accède à une autre interface graphique dédiée au processus d'inscription.

Si l'utilisateur tente de s'inscrire avec un nom d'utilisateur déjà existant, un message d'erreur s'affiche : "Utilisateur déjà existant."

Après une inscription réussie, un message de succès s'affiche, indiquant : "Inscription réussie, vous pouvez maintenant vous connecter."

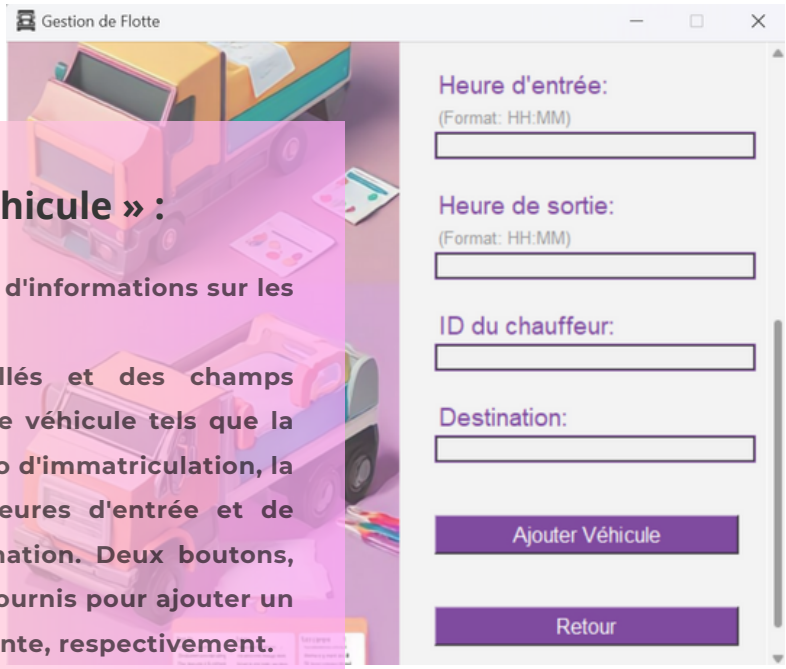
INTERFACES:

2

Interface de Contrôle :

Cette fenêtre s'ouvre après la connexion de l'utilisateur par son nom d'utilisateur et mot de passe, avec une mise en page soigneusement organisée, comprenant une image illustrative sur le côté gauche et un panneau de commandes sur le côté droit. Ce dernier contient des boutons qui déclenchent des actions spécifiques liées à la gestion de la flotte.

Ces actions comprennent l'ajout de véhicules, la modification, la suppression, l'affichage de la liste des véhicules et la déconnexion. Ainsi, chaque bouton a un rôle spécifique :



a

Bouton « Ajouter véhicule » :

Permet d'ouvrir une interface d'ajout d'informations sur les véhicules.
Cette interface contient des libellés et des champs d'entrée pour saisir des détails sur le véhicule tels que la marque, le modèle, l'année, le numéro d'immatriculation, la date de mise en circulation, les heures d'entrée et de sortie, l'ID du chauffeur et la destination. Deux boutons, "Ajouter Véhicule" et "Retour", sont fournis pour ajouter un véhicule et revenir à la page précédente, respectivement.

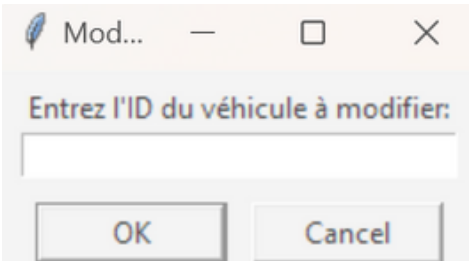
Il est important de noter que la date doit être saisie selon le format année-mois-jour, et l'heure selon le format heures:minutes. En cas de saisie incorrecte de la date, un message d'erreur s'affichera dans une boîte de messages : "Erreur", "Format de date/heure invalide."

b

Bouton « Modifier véhicule »

En cliquant sur ce bouton, l'utilisateur est invité à entrer l'ID du véhicule qu'il souhaite modifier à l'aide d'une boîte de dialogue simple. Si l'ID est invalide, un message d'erreur s'affichera dans une boîte de messages indiquant : "Erreur, Aucun véhicule trouvé avec cet ID". Dans le cas contraire , une nouvelle interface s'ouvre contenant des champs d'entrée permettant de modifier l'heure d'entrée, l'heure de sortie, l'ID du chauffeur et la destination .
À noter que la marque, le modèle, l'année, le numéro d'immatriculation et la date de mise en circulation restent inchangés, car ce sont des champs généralement fixes qui ne nécessitaient pas de modification, raison pour laquelle ils ne sont pas inclus dans cette interface.

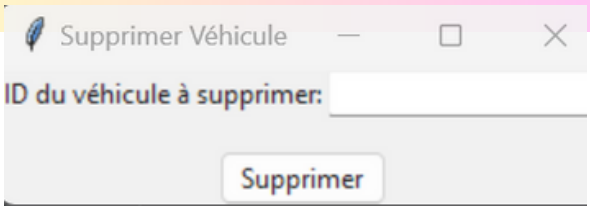
Ainsi, dans cette interface, deux boutons, " Modifier " et "Retour", sont fournis pour confirmer la modification et revenir à la page précédente, respectivement.



c

Bouton « Supprimer véhicule »

Après avoir cliqué sur le bouton "Supprimer Véhicule", l'utilisateur est invité à entrer l'ID du véhicule à supprimer. Une fois l'ID saisi, l'utilisateur peut confirmer la suppression dans une autre fenêtre de dialogue. Si l'ID correspond à un véhicule existant dans la base de données, la suppression est effectuée, et un message de succès est affiché. Si l'ID n'est pas trouvé, un message d'erreur est affiché, informant l'utilisateur qu'aucun véhicule correspondant à cet ID n'a été trouvé.
Il est à noter que dans le cas où l'ID du véhicule saisi n'est pas un nombre entier, une fenêtre d'erreur s'affiche pour informer l'utilisateur que l'ID du véhicule doit être un nombre entier. Cette vérification prévient les erreurs liées à des saisies incorrectes.



d

Bouton « Afficher la liste des véhicules »

En cliquant sur ce bouton, une nouvelle fenêtre se crée pour présenter une liste de véhicules ajoutés par l'utilisateur. Cette fenêtre utilise un widget ‘Treeview’ pour organiser les données de manière tabulaire avec des en-têtes de colonnes. Chaque colonne représente une caractéristique spécifique du véhicule, comme l'ID, la matricule, la marque, le modèle, l'année, la date de mise en circulation, l'heure d'entrée, l'heure de sortie, l'ID du chauffeur et la destination.

Si aucun véhicule n'est trouvé pour le propriétaire actuel, une boîte de dialogue informative s'affiche pour informer l'utilisateur qu'aucun véhicule n'a été trouvé.

CODE PYTHON

Notre code de réalisation de l'application de gestion de flotte en Python utilise la bibliothèque Tkinter pour l'interface graphique et se connecte à une base de données SQLite. Cette base de données relationnelle légère, stocke les informations sur les utilisateurs et les véhicules dans deux tables distinctes.

Ainsi, le code comporte les éléments suivants :

- Classe "Application" :

La classe Application est une sous-classe de tk.Tk. Elle représente la fenêtre principale de notre application. Elle comporte un constructeur _init_ qui initialise la fenêtre avec des paramètres tels que le titre, la taille, l'icône, et établit la connexion à la base de données SQLite.
La base de données contient deux tables :

. Table Utilisateurs :

- id: Identifiant unique pour chaque utilisateur (clé primaire).
- nom_utilisateur: Nom d'utilisateur.
- mot_de_passe: Mot de passe associé à l'utilisateur.

. Table Vehicules :

- id: Identifiant unique pour chaque véhicule (clé primaire).
- immatriculation: Numéro dimmatriculation du véhicule.
- marque: Marque du véhicule.
- modele: Modèle du véhicule.
- year: Année du véhicule.
- date_entree: Date de mise en circulation du véhicule.
- heure-entree: Heure d'entrée du véhicule.
- heure-sortie: Heure de sortie du véhicule.
- chauffeur_id: ID du chauffeur associé au véhicule.
- destination: Destination du véhicule.
- proprietaire_id: ID de l'utilisateur propriétaire du véhicule (clé étrangère faisant référence à la table Users).

```python
import tkinter as tk
from tkinter import ttk, simpledialog, messagebox
import sqlite3
from datetime import datetime
from PIL import Image, ImageTk

class Application(tk.Tk):

    def __init__(self):
        super().__init__()

        self.title("Gestion de Flotte")
        self.config(bg="black")
        self.iconbitmap("b.ico")
        self.geometry("600x480")
        self.resizable(0, 0)
        # Base de données SQLite
        self.conn = sqlite3.connect("gestion_flotte.db")
        self.cursor = self.conn.cursor()
        self.cursor.execute('''
            CREATE TABLE IF NOT EXISTS Utilisateurs (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            nom_utilisateur TEXT NOT NULL,
            mot_de_passe TEXT NOT NULL
        )
    ''')

    # Création de la table Véhicules si elle n'existe pas
    self.cursor.execute('''
        CREATE TABLE IF NOT EXISTS Vehicules (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            immatriculation INTEGER NOT NULL,
            marque TEXT NOT NULL,
            modele TEXT NOT NULL,
            annee INTEGER NOT NULL,
            date_entree TEXT,
            heure_entree TEXT,
            heure_sortie TEXT,
            chauffeur_id INTEGER NOT NULL,
            destination TEXT,
            proprietaire_id INTEGER,
            FOREIGN KEY (proprietaire_id) REFERENCES Utilisateurs(id)
        `
            ''')
        self.is_authenticated = False
        self.connexion()
```

id	immatriculation	marque	modele	annee	:e_entr	heure_entree	heure_sortie	chauffeur_id	destination	proprietaire_id
Filtre	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre
3	123456	RENAULT	T520	2015	2015...	12:00:00	12:30:00	12345	CASABLANCA	3
4	987654	Hyundai	H100	2019	2019...	11:00:00	13:00:00	876543	EL JADIDA	3

CODE PYTHON

Les méthodes utilisées dans cette classe sont les suivantes:

```
def connexion(self):
```

Crée l'interface utilisateur de la page de connexion avec des champs pour le nom d'utilisateur et le mot de passe, ainsi que des boutons pour la connexion et l'inscription.

```
def inscription(self):
```

Crée l'interface utilisateur pour la page d'inscription, avec des champs pour le nouveau nom d'utilisateur et le nouveau mot de passe.

```
def gestion_flotte(self):
```

Affiche l'interface utilisateur principale après la connexion réussie, avec des boutons pour les différentes opérations liées à la gestion de la flotte (ajout, modification, suppression de véhicules, affichage de la liste des véhicules, déconnexion).

```
def ajouter_vehicule(self):
```

```
def ajouter_vehicule_2(self):
```

Permettent d'ajouter un nouveau véhicule à la base de données. La première méthode initialise l'interface utilisateur, la deuxième traite les données saisies, et effectue l'insertion dans la base de données.

```
def connexion2(self):
```

La fonction `connexion2` gère la connexion d'un utilisateur. Elle récupère le nom d'utilisateur et le mot de passe, vérifie s'ils sont renseignés, puis exécute une requête SQL pour vérifier l'existence d'un utilisateur correspondant dans la base de données. Si une correspondance est trouvée, elle met à jour les attributs de la classe pour indiquer que l'utilisateur est authentifié, stocke l'identifiant de l'utilisateur actuel, et appelle une autre fonction (`gestion_flotte`). En cas d'échec, elle affiche un message d'erreur.

```
def modifer_vehicule(self):
```

```
def modifer_2(self,vehicle_id):
```

```
def modifer_3(self,vehicle_id):
```

Permettent de modifier les détails d'un véhicule existant. La première méthode demande à l'utilisateur de saisir l'ID du véhicule, la deuxième initialise l'interface utilisateur de modification, et la troisième effectue la modification dans la base de données.

CODE PYTHON

```
def afficher_vehicule(self):
```

Crée une nouvelle fenêtre pour afficher la liste des véhicules sous forme de tableau.

```
def inscription2(self):
```

Traite les données saisies lors de l'inscription et insère un nouvel utilisateur dans la base de données.

```
def deconnexion(self):
```

Réinitialise l'interface utilisateur à la page de connexion.

```
def initialiser_fenetre(self):
```

Détruit tous les widgets dans la fenêtre pour initialiser une nouvelle page.

```
def supprimer_vehicule(self):
```

```
def confirmer_suppression(self, vehicle_id):
```

Permettent de supprimer un véhicule de la base de données.

Bloc principal if _name_ == "_main_":

Crée une instance de la classe Application et lance la boucle principale de l'interface graphique.

```
if __name__ == "__main__":
    app = Application()
    app.mainloop()
```

CONCLUSION

En conclusion, la réalisation de cette application de gestion de flotte de véhicules a été une expérience enrichissante qui nous a permis de consolider nos compétences acquises durant notre formation en programmation orientée objet .

Ce projet a été l'occasion d'appliquer concrètement nos connaissances, tout en nous confrontant à de nouvelles techniques. La mise en œuvre s'est effectuée principalement à l'aide du langage de programmation Python, avec l'interface utilisateur développée grâce à Tkinter, et la gestion de la base de données assurée par SQLite3. Ces outils ont joué un rôle crucial dans le succès de notre projet, offrant une approche robuste et efficace pour répondre aux besoins de gestion de la flotte de véhicules.