Suite du projet concessionnaire automobile - Base de donnée et interface Web

Pour ces exercices, nous allons réutiliser le code de l'exercice précédent, pour l'améliorer.

03-fonctions-packages.4.2

Exercice 1 : Intégration de la Base de Données et Mise à Jour des Structures

Objectif: Intégrer une base de données pour gérer l'inventaire et mettre à jour les structures de données existantes.

Instructions :

1. Modification des Structures :

- Ouvrez `car.go`.
- Ajoutez un champ `ID` à la structure `Car` pour gérer un identifiant unique pour chaque voiture.
- Exemple de code pour la structure modifiée :

```
type Car struct {
    ID    int    `json:"id"`
    Brand string `json:"brand"`
    // autres champs...
}
```

2. Installation du Driver PostgreSQL :

- Si ce n'est pas déjà fait, installez le driver PostgreSQL : `go get -u github.com/lib/pq` dans votre module.

3. Initialisation de la Connexion à la Base de Données :

- Créez un fichier `db.go`.
- À l'intérieur, écrivez une fonction `InitDB` pour établir une connexion à la base de données.
- Utilisez la chaîne de connexion appropriée pour votre base de données PostgreSQL.
- Exemple de code pour `ConnectDB` :

```
import (
    "database/sql"
    _ "github.com/lib/pq"
)

var db *sql.DB

func ConnectDB(dataSourceName string) {
    var err error
    db, err = sql.Open("postgres", dataSourceName)
    if err != nil {
        log.Fatal(err)
    }

    if err = db.Ping(); err != nil {
        log.Fatal(err)
```

```
    }
}
```

4. Script SQL pour la Table des Voitures :

- Créez un fichier `init.sql`.
- Écrivez un script SQL pour créer une table `cars` correspondant à la structure `Car`.
- Exemple de script SQL :

```
CREATE TABLE IF NOT EXISTS cars (
    id SERIAL PRIMARY KEY,
    brand VARCHAR NOT NULL,
    // autres champs...
);
```

- Dans `main.go`, rajoutez la commande `initdb` à la CLI.
- Cette dernière exécute le script SQL pour créer la table `cars` dans votre base de données, depuis votre code Go (créez une fonction `InitDB` dans `db.go` pour cela).

5. Mise à Jour des Fonctions de Gestion de l'Inventaire :

- Dans `dealership.go`, remplacez les fonctions qui géraient l'inventaire avec des fichiers JSON par des fonctions qui interagissent avec la base de données.
- Par exemple, remplacez `SaveToFile` par `SaveToDB` pour sauvegarder les voitures dans la base de données.
- Exemple de fonction `SaveToDB` :

```
func (d *Dealership) SaveToDB() error {
  for _, car := range d.Cars {
      _, err := db.Exec("INSERT INTO cars (brand) VALUES ($1)", car.Brand)
      if err != nil {
          return err
      }
  }
  return nil
}
```

Workflow mis à jour :

- Initiez le processus en établissant une connexion avec la base de données. Assurez-vous de clôturer cette connexion une fois le programme terminé.
- Chargez les données des voitures depuis la base de données au démarrage.
- Enregistrez uniquement les voitures dont les informations ont été modifiées dans la base de données après chaque changement dans l'inventaire.
- Mettez à jour l'inventaire en mémoire et sauvegardez les changements dans la base de données à l'ajout ou la suppression d'une voiture.
- Fermez la connexion à la base de données à la conclusion du programme, sans nécessité de sauvegarder les données de l'inventaire.

Exercice 2 : Ajout d'une Commande 'serve' et Création d'une Interface Web Simple

Objectif: Modifier la CLI pour inclure une commande `serve` qui lance un serveur web et créer une interface web simple pour visualiser l'inventaire.

Instructions :

1. Modification de la CLI :

- Dans `main.go`, ajoutez une nouvelle commande `serve` à la CLI.
- Cette commande appelle une fonction `serve` qui démarre un serveur web.

2. Configuration du Serveur Web :

- Configurez un serveur web en utilisant `net/http`.
- Exemple de code :

```
func showInventory(w http.ResponseWriter, r *http.Request) {
    // Logique pour récupérer et afficher l'inventaire
}
```

3. Intégration de la Commande Serve :

- La commande `serve` doit démarrer le serveur web.
- Exemple de code :

```
http.HandleFunc("/inventory", showInventory)
http.ListenAndServe(":8080", nil)
```

4. Template HTML pour l'Affichage :

- En Go, vous pouvez utiliser le package `html/template` pour gérer des templates HTML.
- Créez un fichier `inventory.html` qui servira de template pour afficher l'inventaire.
- Exemple de code pour `inventory.html` :

```
<!DOCTYPE html>
<html>
<head>
  <title>Inventory</title>
</head>
<body>
  <h1>Car Inventory</h1>
  {{range .}}
  <div>
      <p>{{.ID}} {{.Brand}}</p>
  </div>
  {{end}}
</body>
</html>
```

- Dans `main.go`, utilisez ce template pour afficher l'inventaire de voitures.
- Exemple de code pour intégrer le template dans Go :

```
import "html/template"

var tmpl = template.Must(template.ParseFiles("inventory.html"))

func showInventory(w http.ResponseWriter, r *http.Request) {
  cars := // Récupération de l'inventaire
  err := tmpl.Execute(w, cars)
  if err != nil {
```

```
        http.Error(w, err.Error(), http.StatusInternalServerError)
    }
}
```