TP - Fichiers

Vous devez développer un programme Go pour gérer une liste de contacts. Chaque contact est représenté par un nom et un numéro de téléphone. Le programme doit permettre à l'utilisateur de :

1. Ajouter un contact en saisissant son nom et son numéro de téléphone.
2. Rechercher un contact par son nom et afficher son numéro de téléphone.
3. Afficher la liste complète des contacts, en indiquant leur nom et leur numéro de téléphone.
4. Enregistrer les contacts dans un fichier.
5. Charger les contacts à partir d'un fichier lors du démarrage du programme.

Votre programme devrait afficher le message suivant : "Bienvenue dans le gestionnaire de contacts !"

Ensuite, il devrait afficher un menu avec les options suivantes :

1. Ajouter un contact
2. Rechercher un contact
3. Afficher la liste des contacts
4. Enregistrer les contacts dans un fichier
5. Charger les contacts à partir d'un fichier
6. Quitter

Après chaque action effectuée par l'utilisateur, le menu devrait être réaffiché jusqu'à ce que l'utilisateur choisisse l'option "Quitter".

Solution

Code solution expliqué :

```
package main

import (
	"bufio"
	"fmt"
	"os"
	"strings"
)

type Contact struct {
```

```
	Name  string
	Phone string
}

func main() {
	fmt.Println("Bienvenue dans le gestionnaire de contacts !")

	contacts := make(map[string]string)

	loadContactsFromFile(contacts)

	for {
		fmt.Println("\nMenu:")
		fmt.Println("1. Ajouter un contact")
		fmt.Println("2. Rechercher un contact")
		fmt.Println("3. Afficher la liste des contacts")
		fmt.Println("4. Enregistrer les contacts dans un fichier")
		fmt.Println("5. Charger les contacts à partir d'un fichier")
		fmt.Println("6. Quitter")

		var choice int
		fmt.Print("Votre choix : ")
		fmt.Scan(&choice)

		switch choice {
		case 1:
			var name, phone string
			fmt.Print("Nom du contact : ")
			scanner := bufio.NewScanner(os.Stdin)
			scanner.Scan()
			name = scanner.Text()
			fmt.Print("Numéro de téléphone : ")
			scanner.Scan()
			phone = scanner.Text()
			contacts[name] = phone
			fmt.Println("Contact ajouté avec succès !")

		case 2:
			var name string
			fmt.Print("Nom du contact à rechercher : ")
```

```
		scanner := bufio.NewScanner(os.Stdin)
		scanner.Scan()
		name = scanner.Text()
		phone, found := contacts[name]
		if found {
			fmt.Printf("Numéro de téléphone de %s : %s\n", name, phone)
		} else {
			fmt.Println("Contact non trouvé.")
		}

	case 3:
		fmt.Println("Liste des contacts :")
		for name, phone := range contacts {
			fmt.Printf("- %s : %s\n", name, phone)
		}

	case 4:
		saveContactsToFile(contacts)
		fmt.Println("Contacts enregistrés avec succès !")

	case 5:
		loadContactsFromFile(contacts)
		fmt.Println("Contacts chargés avec succès !")

	case 6:
		fmt.Println("Au revoir !")
		return

	default:
		fmt.Println("Choix invalide. Veuillez sélectionner une option valide.")
	}
}

func loadContactsFromFile(contacts map[string]string) {
	file, err := os.Open("contacts.txt")
	if err != nil {
		return
	}
	defer file.Close()
```

```
	scanner := bufio.NewScanner(file)
	for scanner.Scan() {
		line := scanner.Text()
		parts := strings.Split(line, ":")
		if len(parts) == 2 {
			name := strings.TrimSpace(parts[0])
			phone := strings.TrimSpace(parts[1])
			contacts[name] = phone
		}
	}

	if err := scanner.Err(); err != nil {
		fmt.Println("Erreur lors du chargement des contacts :", err)
	}
}

func saveContactsToFile(contacts map[string]string) {
	file, err := os.Create("contacts.txt")
	if err != nil {
		fmt.Println("Erreur lors de l'enregistrement des contacts :", err)
		return
	}
	defer file.Close()

	writer := bufio.NewWriter(file)
	for name, phone := range contacts {
		line := fmt.Sprintf("%s: %s\n", name, phone)
		_, err := writer.WriteString(line)
		if err != nil {
			fmt.Println("Erreur lors de l'écriture dans le fichier :", err)
		}
	}

	writer.Flush()
}
```

Explication du code :

- Une structure `Contact` est définie avec deux champs : `Name` pour le nom du contact et `Phone` pour son numéro de téléphone.

- La fonction `main()` est la fonction principale qui est exécutée lorsque le programme est lancé.
- Une map `contacts` est utilisée pour stocker les contacts, où la clé est le nom du contact et la valeur est son numéro de téléphone.
- La fonction `loadContactsFromFile()` est utilisée pour charger les contacts à partir d'un fichier `contacts.txt`. Chaque ligne du fichier doit être au format "Nom: Numéro".
- La fonction `saveContactsToFile()` est utilisée pour enregistrer les contacts dans le fichier `contacts.txt`. Chaque contact est écrit sur une ligne séparée avec le format "Nom: Numéro".
- Le programme utilise une boucle `for` pour afficher le menu et traiter les choix de l'utilisateur jusqu'à ce que l'option "Quitter" soit sélectionnée.
- Selon le choix de l'utilisateur, différentes actions sont effectuées :
  - Pour l'option 1, l'utilisateur est invité à saisir le nom et le numéro de téléphone du contact, puis le contact est ajouté à la map `contacts`.
  - Pour l'option 2, l'utilisateur est invité à saisir le nom du contact à rechercher. Si le contact est trouvé dans la map `contacts`, son numéro de téléphone est affiché.
  - Pour l'option 3, tous les contacts dans la map `contacts` sont affichés avec leur nom et leur numéro de téléphone.
  - Pour l'option 4, les contacts sont enregistrés dans le fichier `contacts.txt` en utilisant la fonction `saveContactsToFile()`.
  - Pour l'option 5, les contacts sont chargés à partir du fichier `contacts.txt` en utilisant la fonction `loadContactsFromFile()`.
  - Pour l'option 6, le programme se termine et affiche "Au revoir !".
  - Si l'utilisateur choisit une option invalide, un message d'erreur est affiché.

Revision #2
Created 2023-05-19 13:09:50 UTC by Noé Larrieu-Lacoste
Updated 2023-05-19 13:37:40 UTC by Noé Larrieu-Lacoste