

# Synthèse - Troisième Semaine

Lors de cette troisième semaine de stage , ma première tâche a également été de réaliser des tests afin de vérifier que toutes les fonctionnalités du site qui est en cours de production par l'entreprise fonctionnaient correctement.

Ma seconde tâche à été l'apprentissage du langage Golang qui est un langage de programmation open source, qui est un langage compilé.

Pour cela j'ai d'abord configuré mon environnement Golang sur Windows et lancer un premier programme Go:

## Prérequis:

Pour exécuter un programme Go il faut :

- Un éditeur de texte

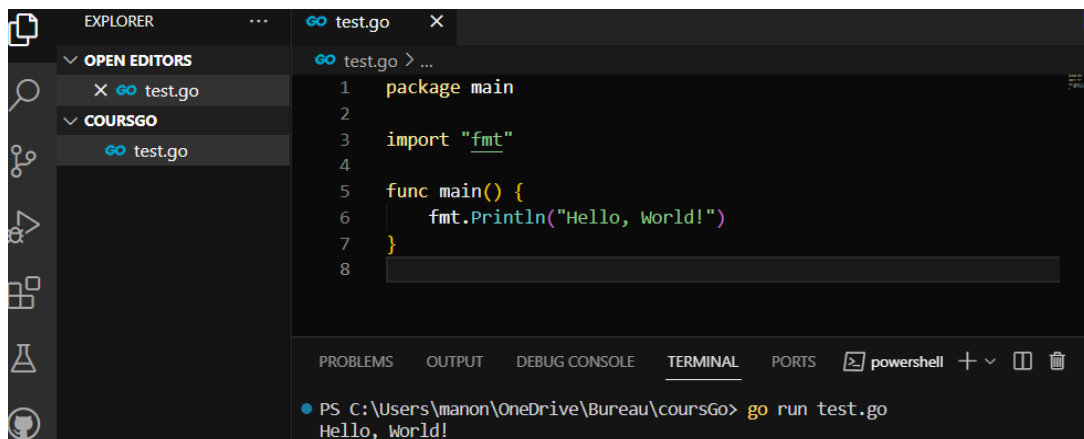
J'utilise un éditeur de texte gratuit et open source à savoir [Visual studio Code](#) avec l'extension [Go](#).

- Un compilateur GO

J'ai téléchargé la dernière version du compilateur [en cliquant ici](#) .

## Tester le compilateur Go:

J'ai créer un fichier, nommez le `test.go`, mis le code suivant puis sauvegardez :



The screenshot shows the Visual Studio Code interface with a Go file named `test.go` open. The code in the editor is as follows:

```
1 package main
2
3 import "fmt"
4
5 func main() {
6     fmt.Println("Hello, World!")
7 }
8
```

Below the editor, the TERMINAL panel is active, showing the command `go run test.go` being executed in a PowerShell terminal. The output of the command is `Hello, World!`.

J'ai revu les notions de base en langage Golang (variables, conditions, les boucles, les fonctions, les tableaux...) pour voir la syntaxe spécifique au langage et comment il fonctionne.

**Ma deuxième tâche a été de générer un fichier à partir d'un programme en Golang:**

```
package main

import (
    "fmt"
    "net/http"
    "os"
)

func main() {
    // Définition de la route "/hello" avec la fonction de gestion
    http.HandleFunc("/hello", helloHandler)

    // Démarrage du serveur sur le port 8080
    port := 8087
    fmt.Printf("Serveur écoutant sur le port %d...\n", port)
    err := http.ListenAndServe(fmt.Sprintf(":%d", port), nil)

    // Gestion des erreurs
    if err != nil {
        fmt.Println("Erreur:", err)
    }
}

// Fonction de gestion pour la route "/hello"
func helloHandler(w http.ResponseWriter, r *http.Request) {
    // Écriture du message "Hello, World!" dans un fichier nommé "output.txt"
    err := writeToFile("output.txt", "Hello, World!")

    // Vérification des erreurs
    if err != nil {
        http.Error(w, "Erreur interne du serveur", http.StatusInternalServerError)
        return
    }

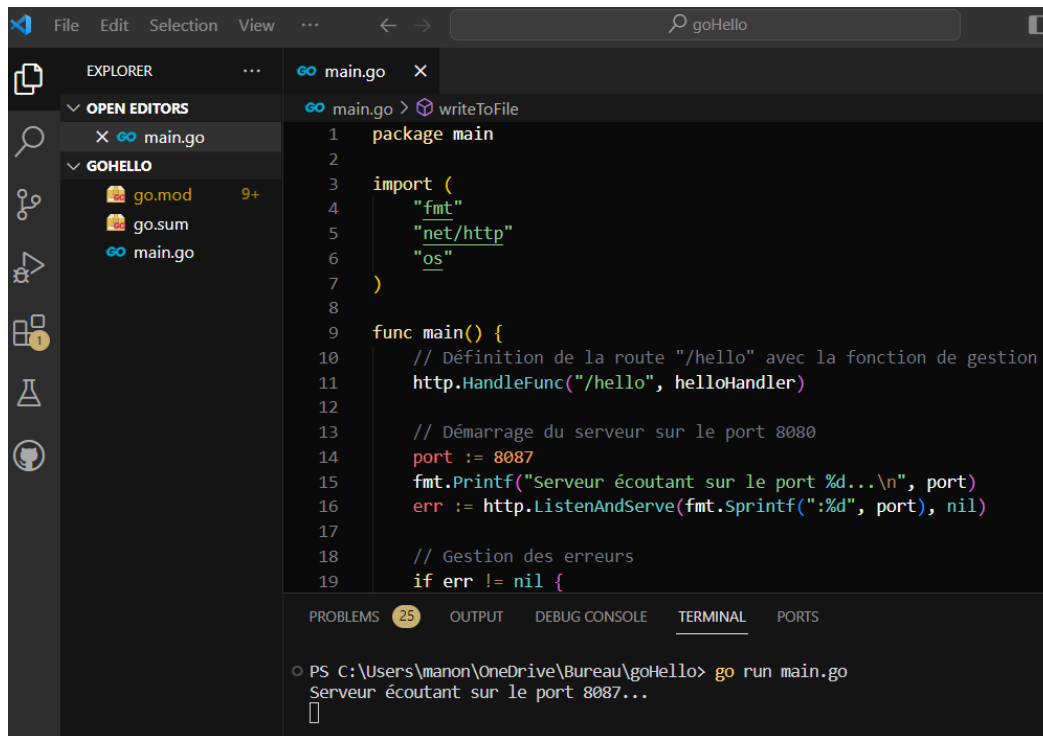
    // Écriture de la réponse dans le corps de la réponse
    fmt.Fprint(w, "Hello, World! Écriture dans le fichier réussie.")
}

// Fonction pour écrire dans un fichier
func writeToFile(filename, content string) error {
    // Ouverture du fichier en mode écriture, création s'il n'existe pas
    file, err := os.Create(filename)
    if err != nil {
        return err
    }
    defer file.Close()

    // Écriture du contenu dans le fichier
    _, err = file.WriteString(content)
    if err != nil {
        return err
    }

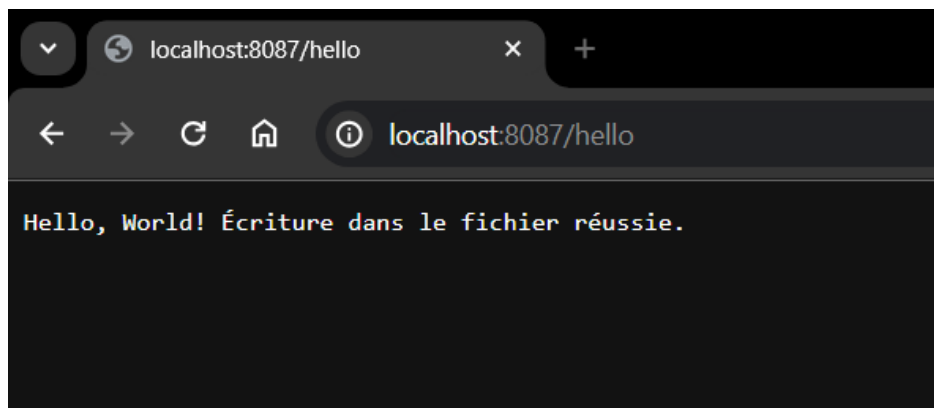
    return nil
}
```

Avant écriture dans le fichier=



```
1 package main
2
3 import (
4     "fmt"
5     "net/http"
6     "os"
7 )
8
9 func main() {
10     // Définition de la route "/hello" avec la fonction de gestion
11     http.HandleFunc("/hello", helloHandler)
12
13     // Démarrage du serveur sur le port 8080
14     port := 8087
15     fmt.Printf("Serveur écoutant sur le port %d...\n", port)
16     err := http.ListenAndServe(fmt.Sprintf(":%d", port), nil)
17
18     // Gestion des erreurs
19     if err != nil {
```

PS C:\Users\manon\OneDrive\Bureau\goHello> go run main.go  
Serveur écoutant sur le port 8087...



Après écriture dans le fichier=

The screenshot shows the Visual Studio Code interface with a Go project named 'goHello'. The Explorer sidebar on the left shows the file structure: 'main.go' is open in the editor, and 'go.mod', 'go.sum', and 'output.txt' are listed in the project. The main editor displays the following Go code:

```
1 package main
2
3 import (
4     "fmt"
5     "net/http"
6     "os"
7 )
8
9 func main() {
10     // Définition de la route "/hello" avec la fonction de gestion
11     http.HandleFunc("/hello", helloHandler)
12
13     // Démarrage du serveur sur le port 8080
14     port := 8087
15     fmt.Printf("Serveur écoutant sur le port %d...\n", port)
16     err := http.ListenAndServe(fmt.Sprintf(":%d", port), nil)
17
18     // Gestion des erreurs
19     if err != nil {
```

The bottom panel shows the 'TERMINAL' tab with the following output:

```
PS C:\Users\manon\OneDrive\Bureau\goHello> go run main.go
Serveur écoutant sur le port 8087...
```

The screenshot shows the 'output.txt' file in the editor, which contains the output of the Go program:

```
1 Hello, World!
```