

# Rapport de projet Portfolio

## Contributeur

- Gabriel Picard
- Guillaume Dorges
- Ludovic Dumet

## But du projet:

Faire un portfolio en html / css autre technologies autorisées, le back-end sera en go et la gestion de la BDD se fera via SQL ou sqlite3.

Dans le cas présent le front est un mélange HTML5/CSS et bootstraps

Repo : [https://github.com/MaoenD/Projet\\_Portefolio](https://github.com/MaoenD/Projet_Portefolio)

## Fonctionnalités

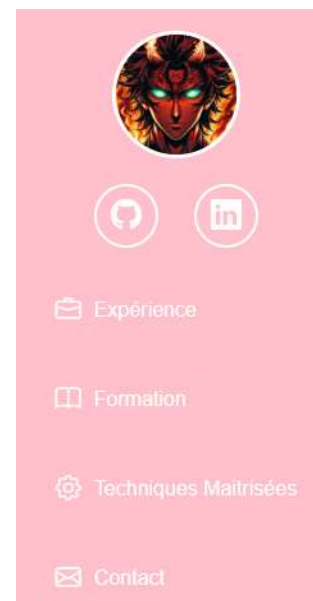
### Zone gauche

#### //sommaire

Photo de profil avec encadrement

2 boutons qui redirige vers le Github et le linkdin

Un sommaire composé de nav-link vers les différentes section via un scroll en JavaScript



### Zone droite

#### //Section 1

Bg image importée

Script d'auto écriture en boucle en JavaScript



## //Section 2 et 3

Appel de la base de données

Manipulation de la position du texte via CSS

Exemple 1  
de 06/2025 à 09/2026 sur 15 mois  
test

Exemple 2  
de 10/2026 à 12/2026 sur 2 mois  
Test

Expérience

## //Section 4/5

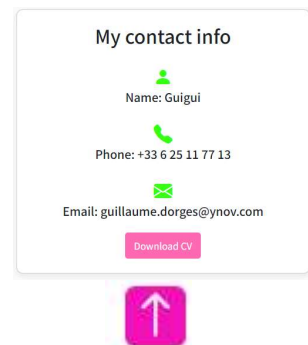
En bootstrap.

et de l'affichage / import image / bouton de téléchargement.

## //Flèche back to the top

Clic simple script JavaScript qui retourne à la première section.

Clic double redirection vers page login admin.

A contact information form titled "My contact info". It contains fields for Name (Guigui), Phone (+33 6 25 11 77 13), and Email (guillaume.dorges@ynov.com). There is a "Download CV" button and a "back to top" button with an upward arrow icon.

## Administrateur

### // Accès

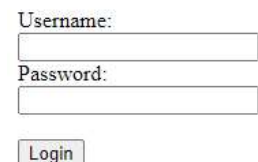
Possible uniquement via le double clic sur la flèche .

Si l'adresse est rentrée telle quelle, la page sera blanche.

### // Page de login

Si login ou password ne correspond pas pas de connexion possible.

### Login

A login form with fields for "Username:" and "Password:". Below the fields is a "Login" button.

Admin Expériences

### // Page adminXP/admin Formation

En bootstrap.

Un get all qui va récupérer l'ID et l'intitulé.

Un get by ID qui va permettre:

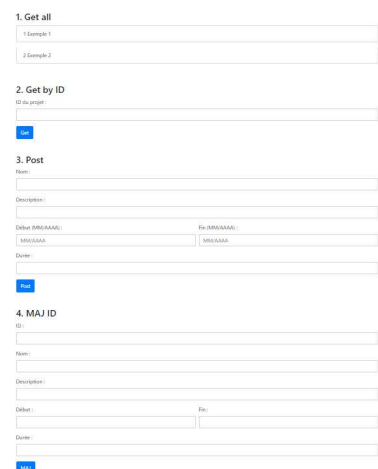
- de récupérer les informations .
- de la précharger pour l'update.

Un post pour ajouter à la base de données.

L'update afin de modifier.

2 boutons supplémentaires:

- 1 Pour changer de page admin.
- 1 pour se déconnecter.

A form titled "Admin Expériences" with four sections: 1. Get all (with a table showing examples), 2. Get by ID (with a search field and a "Get" button), 3. Post (with fields for Name, Description, Debut (YYYY-MM), Fin (YYYY-MM), and Date), and 4. MAJ ID (with fields for ID, Name, Description, Debut, Fin, and Date). Each section has a "New" button.

## Code

### // BDD

Un fichier database configuré via sqlite3

Fonction de lecture, connection et création pour la gestion de la BDD

### BackEnd

### // CRUD

Dans le cas présent il s'agit plutôt d'un CRU, je prends une fonction de chaque fichier  
Create (Post)

```
func PostProjet(db *sql.DB, NomProjet string, Description string,
    DateDebut string, DateFin string, Duree string) {
    _, err := db.Exec("INSERT INTO Projet ( Nom_Projet,
        Description, Date_Debut, Date_Fin, Duree) VALUES ( ?, ?, ?, ?, ?)",
        NomProjet, Description, DateDebut, DateFin, Duree)
    if err != nil {
        log.Println(err)
    }
}
```

- Permet l'insertion d'un nouveau projet dans la db selon les paramètres indiqués
- Exécution d'une requête SQL insert, selon les paramètres donnés.
- db.exec est utilisé pour exécuter une requête qui ne renvoie pas de ligne.
- les "?" sont des placeholder remplacer par les variables qui seront fournies et servent de "sécurité" afin d'éviter des injections SQL.
- Si il y a échec de l'insertion, l'erreur sera capturée dans err.
- Si err n'est pas nulle elle sera enregistrée dans le journal via le print.

### //Read (Get):

```
func GetAllProjet(db *sql.DB) ([]Projet, error) {
    rows, err := db.Query("SELECT * FROM Projet")
    if err != nil {
        return nil, err
    }
    defer rows.Close()
    var Projets []Projet
    for rows.Next() {
        var Projet Projet
        if err := rows.Scan(&Projet.Id_Projet,
            &Projet.Nom_Projet, &Projet.Description,
            &Projet.Date_Debut, &Projet.Date_Fin, &Projet.Durée);
            err != nil {
                return nil, err
            }
        Projets = append(Projets, Projet)
    }
    return Projets, nil
}
```

- Cela renverra une liste des projet et une erreur.
- Envoie d'une requête SQL qui renverra le résultat ou une erreur.
- le defer pour garantir que row.close sera exécutée en fin de fonction.
- une boucle pour avancer vers la ligne suivante.
- row.Scan pour extraire les colonnes de la ligne courante et les stocker dans la structure projet Projet si err arrêt immédiat.
- Ajout de projet à la liste Projet .
- Renvoi de la liste.

## //Update

```
func UpdateProjetNameById(db *sql.DB, id int, newProjetName string) {  
    _, err := db.Exec("UPDATE Projet SET Nom_Projet = ? WHERE Id_Projet = ?", newProjetName, id)  
    if err != nil {  
        log.Println(err)  
    }  
}
```

- Permet la mise à jour selon les paramètres demandés.
- Exécution d'une requête SQL update du Nom\_Projet d'une ligne spécifique de db.
- Même chose pour les placeholders que le post.
- Même chose que post si l'erreur n'est pas nulle.

## // handler

Fonction: HandleIndexPage

```
func HandleIndexPage(w http.ResponseWriter, r *http.Request) {  
    logged = false  
    datas := GetDatas()  
    Fdatas := GetFormationsDatas()  
    data := map[string]interface{}{  
        "Projets": datas,  
        "Formation": Fdatas,  
    }  
    if r.Method == http.MethodGet {  
    }  
    templates.RenderTemplate(w, "index", data)  
    return  
}
```

Paramètres:

- o w: Réponse HTTP que le serveur enverra au client.
- o r: Requête HTTP reçue du client.
- Retourne:
  - o Rien.
- Étapes:
  - o Charge le fichier de modèle index.html.
  - o En cas d'erreur lors du chargement, elle renvoie une erreur HTTP 500.
  - o Si le fichier est correctement chargé, il est rendu dans la réponse HTTP.

Fonction: LoginHandler

```
func HandleLoginPage(w http.ResponseWriter, r *http.Request) {  
    templates.RenderTemplate(w, "login", nil)  
}
```

- Paramètres:
  - o w: La réponse HTTP envoyée au client.
  - o r: La requête HTTP reçue du client.
- Retourne:
  - o Rien.
- Étapes:
  - o Si la méthode HTTP est un POST, on extrait les champs username et password du formulaire.
  - o Si les identifiants sont corrects (ici admin et 1234 en dur), l'utilisateur est redirigé vers /dashboard.
  - o Si les identifiants sont incorrects, une erreur 401 est renvoyée.
  - o Si la requête n'est pas un POST, le modèle login.html est chargé et affiché.

## // server

```
func Start() {  
    cfg := database.ReadDB()  
    database.ConnectDB(cfg.DatabaseURL)  
    database.CreateTable()  
  
    http.Handle("/Style/", http.StripPrefix("/Style/", http.FileServer(http.Dir("Style"))))  
    http.Handle("/ressource/", http.StripPrefix("/ressource/", http.FileServer(http.Dir("ressource"))))  
  
    http.HandleFunc("/adminXP", HtmlLink.HandleAdminPage)  
    http.HandleFunc("/adminFormations", HtmlLink.HandleFormationPage)  
    http.HandleFunc("/index", HtmlLink.HandleIndexPage)  
    http.HandleFunc("/login", HtmlLink.HandleLoginPage)  
  
    log.Println("Listening on :8080...")  
    // Starting the server on port 8080  
  
    if err := http.ListenAndServe(":8080", nil); err != nil {  
        log.Fatal(err)  
    }  
    // Starting the HTTP server.  
}
```

Lecture de la configuration de la base de données :

- Charge la configuration de la base de données depuis un fichier ou une source externe.

Connexion à la base de données :

- Se connecte à la base de données en utilisant l'URL fournie dans la configuration.

Création de la table :

- Crée les tables nécessaires dans la base de données si elles n'existent pas encore.

Gestion des fichiers statiques

- Les lignes avec http.Handle configurent le serveur pour servir des fichiers statiques.

Gestion des routes HTTP :

- http.HandleFunc associe des URL spécifiques à des fonctions de gestion, définies ailleurs dans le code, pour traiter les requêtes HTTP correspondantes.

Démarrage du serveur HTTP :

- Indique que le serveur démarre.
- Lance le serveur sur le port 8080 et attend les connexions entrantes.

- En cas d'échec, le serveur affiche une erreur et arrête le programme avec `log.Fatal(err)`.

## // template

```
var templates = template.Must(template.ParseGlob("web/*.html"))

func RenderTemplate(w http.ResponseWriter, tmpl string, data map[string]interface{}) {
    err := templates.ExecuteTemplate(w, tmpl+".html", data)
    if err != nil {
        log.Printf("Error rendering template %s: %v", tmpl, err)
        http.Error(w, "Internal Server Error(212)", http.StatusInternalServerError)
        return
    }
}
```

Chargement des templates :

- La ligne `var templates` charge tous les fichiers HTML du répertoire `web` en tant que templates pour les utiliser dans l'application.

Fonction `RenderTemplate` :

- Rendu d'un template HTML donné avec des données dynamiques. Elle prend comme paramètres :
  - o `w` : Réponse HTTP envoyée au client.
  - o `tmpl` : Nom du template à afficher.
  - o `data` : Données à injecter dans le template.

Exécution du template :

- Affiche le template correspondant au nom passé avec les données et l'envoie au client.

Gestion des erreurs :

- Si une erreur survient lors du rendu, elle est enregistrée dans les logs, et une erreur HTTP 500 (erreur serveur) est envoyée au client.



## FrontEnd

Le CSS est long car Guillaume n'avait pas encore saisi toutes les bases du bootstrap HTML pour gérer les 3 premières sections et pas assez de temps pour tout refaire. Je vais détailler une fonction JS et une ligne bootstrap.

### //JS le "Smooth Scrolling" vers une section

```
document.querySelectorAll('.nav-link').forEach(link => {
  link.addEventListener('click', function(e) {
    e.preventDefault();
    const targetID = this.getAttribute('href');
    const targetSection = document.querySelector(targetID);
    targetSection.scrollIntoView({
      behavior: 'smooth'
    });
  });
});
```

Sélection des liens de navigation :

- Sélectionne tous les éléments du DOM qui ont la classe nav-link.
- Renvoie une collection de tous les éléments correspondants.

Ajout d'un gestionnaire d'événements click :

- Pour chaque lien de navigation, ajout d'un écouteur d'événement qui déclenche une fonction lors du clic sur ce lien.
- Se déclenche au clic et reçoit l'événement e en paramètre.

Empêcher le comportement par défaut du lien :

- Cette ligne empêche le comportement par défaut du lien, qui est de rediriger vers une nouvelle page ou section instantanément via l'attribut href.

Récupération de l'ID de la section cible :

- Récupère l'attribut href du lien cliqué. Cet attribut contient généralement un identifiant de section sous forme de lien (par ex. #about, #contact).

Sélection de la section cible :

- Sélectionne la section du document qui correspond à l'ID récupéré (targetID). Par exemple, si href="#about", cette ligne sélectionne l'élément ayant id="about".

Défilement fluide vers la section cible :

- Cette méthode fait défiler la page jusqu'à la section cible.
- L'option behavior: 'smooth' assure que le défilement se fait de manière fluide et progressive.

## //Ligne bootstrap

```
<div class="skill-box p-4" style="width: 300px; height: 100px; border: 2px solid #ccc; border-radius: 15px; display: flex; align-items: center; justify-content: center;">
```

Classe skill-box et p-4 : Le div appartient à la classe skill-box, qui peut contenir des styles définis ailleurs dans le CSS. La classe p-4 est une classe Bootstrap qui applique un padding de 4 unités à l'intérieur de la boîte.

Dimensions (width, height) :

- La largeur de l'élément est fixée à 300 pixels.
- La hauteur est fixée à 100 pixels, ce qui crée un conteneur de forme rectangulaire.

Bordure (border) :

- L'élément a une bordure solide de 2 pixels d'épaisseur avec une couleur gris clair .

Coins arrondis (border-radius) :

- Les coins de l'élément sont arrondis avec un rayon de 15 pixels, donnant à la boîte un aspect plus doux.

Flexbox (display: flex) :

- La boîte utilise le modèle Flexbox pour organiser et centrer son contenu.
- Aligne verticalement le contenu au centre de la boîte.
- Aligne horizontalement le contenu au centre.