Documentation technique – Projet Porfolio

Projet : Développement d'une application full stack en 5 Jours (Portfolio)

Groupe *: Raphaël Caron, Jordan Milleville Lino, Oscar Li, Lucas Gerard, Yanis Fronteau*

Introduction

Ce projet a été réalisé dans un cadre de développement rapide, où l'objectif était de créer une **application web Full Stack** avec un délai restreint de 5 jours. Le but de l'application est de permettre à un utilisateur de **gérer un portfolio personnel**, incluant la modification d'informations telles que la photo de profil, le nom, les descriptions, et les projets via une interface d'administration. Cette application a été développée en utilisant **Go** pour la gestion du back-end et une base de données **SQLite3**, tandis que le frontend est composé de pages **HTML, CSS, et JavaScript**.

Le projet, réalisé en groupe, a permis d'expérimenter le **travail collaboratif**, la **gestion de version avec GitHub**, et l'application des **bonnes pratiques** de développement. L'application est conçue pour être facilement **utilisable** et **personnalisable**, tout en répondant aux attentes d'une **interface moderne** et **responsive**.

Planification et Répartition des Rôles

La planification des tâches a été faite via des discussions de groupe, en utilisant des outils pour organiser les user stories et suivre les tâches dans un tableau Kanban. Chaque membre a choisi des responsabilités en fonction de ses compétences (backend, frontend, documentation, etc.). Cela a permis d'assurer une bonne répartition du travail.

Répartition des tâches principales:

Backend : Jordan, Oscar et Raphael

Frontend : Lucas et Yanis

Documentation : Raphael, Lucas

Notion, gestion de projet

Trello, gestion de projet en ligne

Discord, plateforme de messagerie instantanée

1. Architecture de l'Application

L'architecture de l'application est basée sur une approche **multi-page** avec une gestion back-end en **Go** et une base de données **SQLite3**. Voici les éléments clés de l'architecture :

- **Front-end** : Composé de pages HTML, CSS et JavaScript avec un design inspiré d'un design d'actualité, le ***Bento Grid*** pour un portfolio moderne et épuré.

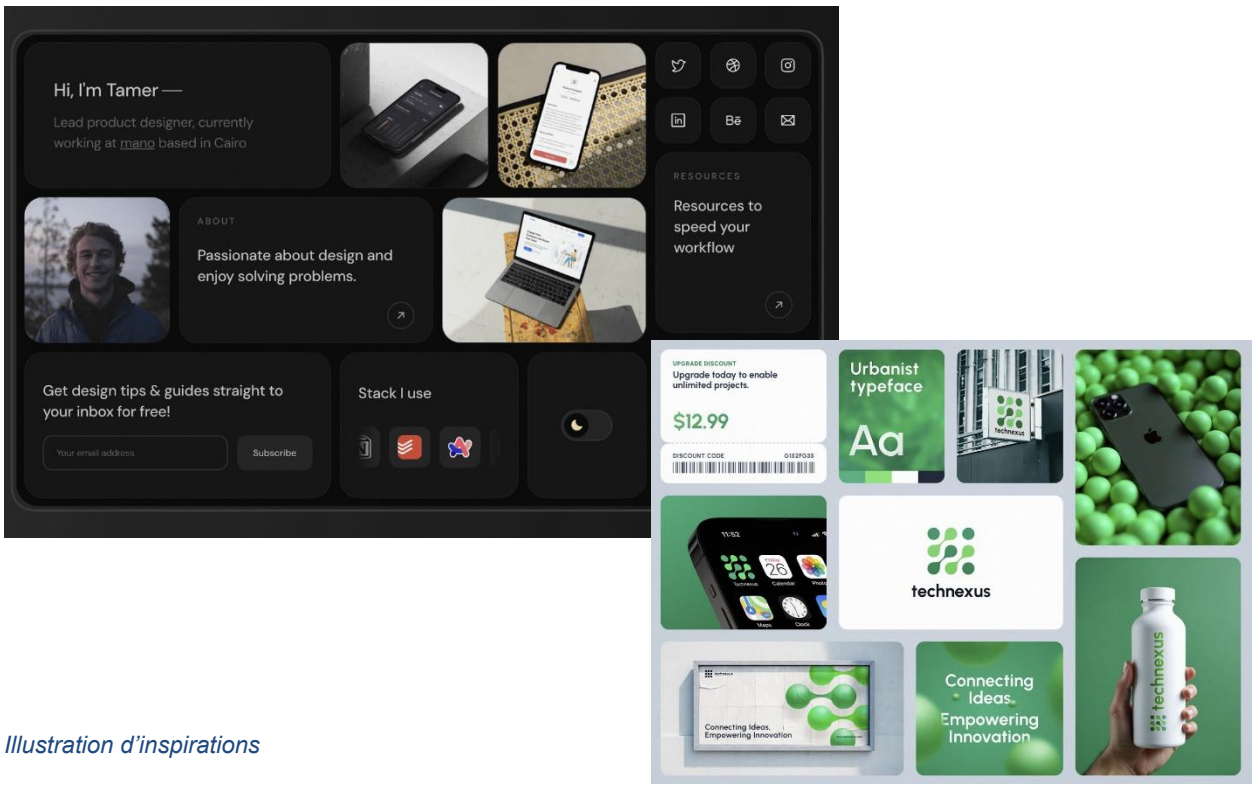


Illustration d'inspirations

- Pages principales :
  - **Index** : Présente un portfolio avec la plupart des informations (Nom, description, Photos), des éléments cliquables comme les pages permettant de voir nos différents projets et formations ainsi que nos différents réseaux sociaux.

  - **Formation** : Montre les différentes formations avec les informations utiles comme le nom de la formation, la période ainsi qu'une description et photo facultative.

  - **Projets** : Affiche une liste de projets avec des informations détaillées. Comme pour la page formation, il y a les détails et informations utiles comme le nom du projet, la date, une description, lien du projet et photo facultative.

    - **Popup :** Page dans laquelle il nous faut rentrer les identifiants pour accéder à la page Admin.

    - **Admin** : Accessible uniquement aux utilisateurs avec les bons identifiants. Cette page permet de modifier le contenu du portfolio, incluant les images, les descriptions et les informations des réseaux sociaux. Les modifications sont effectuées via des pop-ups et des formulaires.

- **Back-end** : Utilise **Go** pour la gestion des requêtes (GET, POST, PUT, DELETE). Le back-end traite les requêtes envoyées par l'interface d'administration et interagit avec la base de données SQLite3 pour mettre à jour les informations.
- **Base de données** : Le projet utilise **SQLite3** pour stocker les informations de l'utilisateur (nom, description, photos, etc.). Chaque mise à jour effectuée via l'interface admin est enregistrée dans la base de données et reflétée en HTML.
- **Sécurité** : Les mots de passe sont hashés pour sécuriser l'accès à la page d'administration.

2. Endpoints API (Go)

L'application utilise Go pour gérer les requêtes HTTP. Voici les principaux **endpoints** définis :

- **GET** : Récupère les informations à afficher dans les différentes pages du portfolio.
    - Exemple : `GET /api/user` pour récupérer les informations de l'utilisateur.
- **POST** : Permet l'envoi de nouvelles informations (ex. mise à jour de l'image ou des informations de description).
    - Exemple : `POST /api/update/image` pour envoyer une nouvelle image à enregistrer.
- **PUT** : Permet de mettre à jour les données existantes.
    - Exemple : `PUT /api/update/profile` pour modifier la description ou les informations personnelles de l'utilisateur.
- **DELETE** : Supprime des éléments si nécessaire, comme des images ou des projets.

Tous ces endpoints interagissent directement avec la base de données SQLite3.

3. Instructions pour Cloner, Installer et Exécuter l'Application

Voici les étapes pour cloner et exécuter l'application en local :

1. **Cloner le projet depuis GitHub** :

Ouvrir le terminal et exécuter la commande suivante :
bash
Copier le code

```
git clone https://github.com/LucasGYnov/portfolio_project.git
```

-

2. **Pré-requis** :
    a. **Go** : Assurez-vous d'avoir Go installé sur votre machine. Si ce n'est pas le cas, vous pouvez l'installer à partir du site officiel : https://golang.org/dl/
    b. **SQLite3** : SQLite3 est intégré, mais il est préférable de s'assurer qu'il est configuré correctement sur votre environnement.
3. **Installer les dépendances** :

Naviguer dans le dossier du projet :
bash
Copier le code

```
cd portfolio_project
```

-

Exécuter le fichier Go :
bash
Copier le code

```
go run main.go
```

-

4. **Exécuter l'application** :
    a. Une fois le serveur lancé, vous pouvez accéder à l'application via votre navigateur à l'adresse : `http://localhost:6969`.

4. Problème et Solution Implémentée

Problème à Résoudre :

Créer une application de **portfolio personnel** permettant de modifier et d'actualiser facilement les informations du CV, y compris les images, descriptions et projets. L'objectif est de proposer un **portfolio adaptable** et facile à mettre à jour via une interface d'administration simple.

Solution Implémentée :

L'interface admin permet à l'utilisateur de modifier les différentes sections de son portfolio via des formulaires, en gérant directement les informations dans la base de données SQLite3 via des requêtes Go. Cette interface est sécurisée par un système de connexion avec mots de passe hashés.

Il accède à la page admin, en se rendant directement sur l'URL localhost:6969/admin. Il a ensuite accès à un formulaire d'authentification (username, mot de passe). En se connectant avec son identifiant et mot de passe correspondant (username : admin, mot de passe: admin pour l'exemple) puis pourra accéder à la page admin. L'authentification est stockée pour chaque session de la page.

Si l'utilisateur n'a pas les identifiants correct il n'a pas accès à la page admin et est renvoyé vers la page du formulaire pour réessayer.

5. Défis Rencontrés

Les principaux défis rencontrés incluent :

- **Gestion du merge des fichiers front-end et back-end** : Avec des délais serrés, coordonner les modifications et intégrer les différentes parties a été compliqué.
- **Travail en groupe** : La répartition des tâches et la coordination entre les membres ont nécessité une communication continue pour éviter les conflits et résoudre les problèmes techniques rencontrés, en particulier lors des **debugs** collectifs.

6. Fonctionnalités de l'Application

L'application permet :

- De visualiser un portfolio via une interface moderne et responsive.
- De **modifier les informations** via une page admin sécurisée, incluant les images, textes, et réseaux sociaux.
- **Hashing des mots de passe** pour sécuriser l'accès à la page d'administration.
- **Sauvegarde automatique** des modifications dans une base de données SQLite3 et mise à jour instantanée du front-end.

7. Qualité du Code

- Le code est bien structuré :
  - **Go** pour le back-end, avec une séparation claire des routes et des fichiers de gestion des requêtes.
  - **Front-end** : Organisé avec des fichiers HTML et CSS dans un dossier `static`, facilitant le chargement des images et des styles.
- Des **commentaires** ont été ajoutés pour les sections importantes, et des **revues de code** ont été réalisées entre les membres de l'équipe.
- Le projet suit des pratiques standards avec des commits réguliers et descriptifs, ainsi que des **pull requests** bien gérées.

8. Utilisation de GitHub

- Utilisation de plusieurs branches, dont une principale (`main`) et une dédiée au front-end (`front`).
- Les commits sont effectués régulièrement et incluent des descriptions claires des modifications.
- Les **pull requests** sont faites en équipe et discutées, soit lors de sessions de travail, soit via Discord pour assurer la synchronisation.

9. Travail d'Équipe

- Tâches réparties selon les compétences et préférences de chaque membre.
- Utilisation de **Notion** pour suivre l'avancement (Kanban, tableau blanc) et de **Figma** pour les maquettes.
- Collaboration en temps réel avec le partage de code sur Visual Studio Code, facilitant l'édition simultanée.
- Une **bonne communication** a été maintenue tout au long du projet, avec des discussions régulières sur Discord.

10. Test et Validation

Pendant le développement, des tests unitaires ont été effectués sur les routes back-end pour vérifier la bonne gestion des requêtes (GET, POST, PUT, DELETE). Ces tests permettent de s'assurer que les données sont bien transmises et que les erreurs sont correctement gérées.

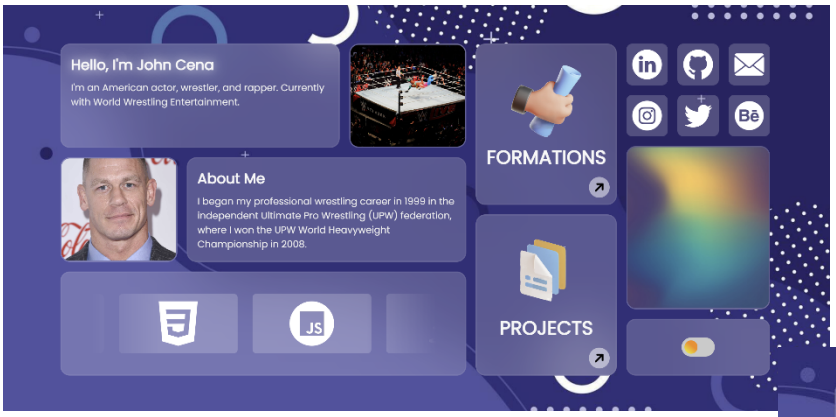
- **Tests Front-end :** Nous avons testé manuellement chaque fonctionnalité sur différents navigateurs pour garantir une compatibilité cross-browser (Chrome, Firefox, Edge).
- **Tests Back-end :** Les endpoints ont été testés avec des outils comme Postman pour vérifier la bonne communication entre le front-end et le back-end.

11. Expériences Apprises

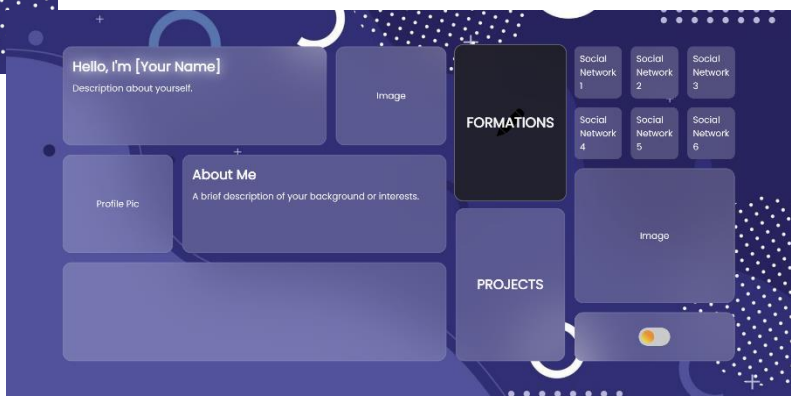
Ce projet a permis à l'équipe de renforcer certaines compétences techniques et collaboratives :

- **Gestion de version avec Git :** Nous avons appris à mieux gérer les branches et à résoudre les conflits lors des fusions de code.
- **Travail en équipe et communication :** La répartition des tâches a montré l'importance de la communication continue, notamment à travers les outils de collaboration comme Notion et Discord.
- **Debugging collectif :** Travailler en groupe sur le débogage de problèmes complexes a renforcé notre capacité à identifier rapidement les bugs et à y remédier de manière efficace

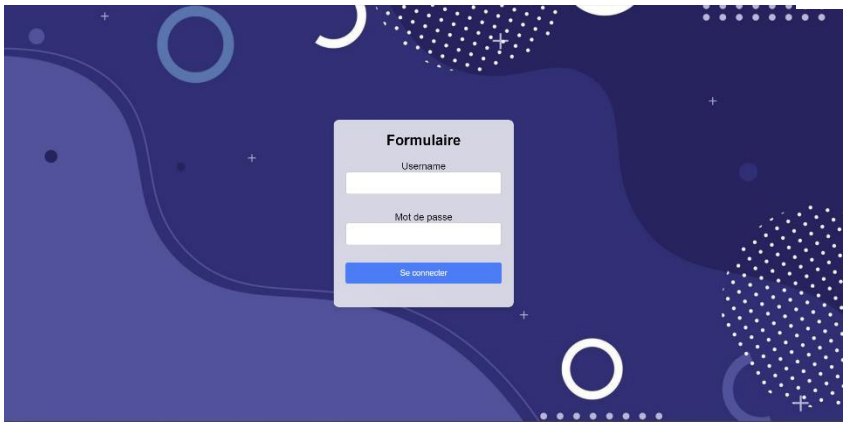
12. Visuel de l’application



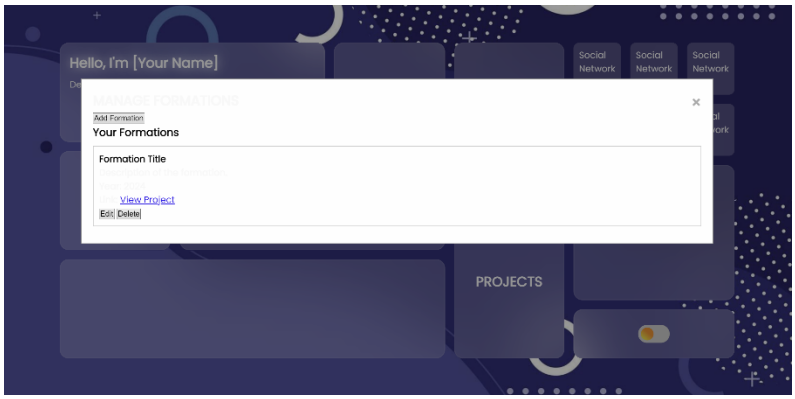
Page principale



Page admin pour les modification des informations



Page du formulaire pour l’accès à la page admin



Formulaire adapté pour modifier les informations

Conclusion

Ce projet a été une expérience enrichissante tant sur le plan technique que collaboratif. En seulement 5 jours, nous avons réussi à concevoir une application full stack fonctionnelle et sécurisée, avec une interface utilisateur moderne et une gestion simple des informations de portfolio via une page d'administration. Malgré les défis rencontrés, nous avons su trouver des solutions efficaces et garantir la qualité de l'application. Il y a bien sûr des possibilités d'améliorations futures, notamment en termes de design et de fonctionnalités, mais nous sommes satisfaits du résultat atteint dans le temps imparti.