BUT MMI
2022 - 2023

métiers du multimédia et de l'internet

Università di Corsica Pasquale Paoli
IUT DI CORSICA
Institut Universitaire de Technologie

Mehdi Ghoulam

SAE 203

Conception d'un site web

Tables des matières

1. Introduction

Le sujet de cette SAÉ s'appuie sur la démarche Portfolio et est la suite directe de la SAÉ 105 proposée au premier semestre. Nous étions invités à développer notre propre site Internet en utilisant les connaissances acquises durant les cours et les autres SAÉ. La mise en avant de nos compétences est ainsi mise en valeur au travers de ce projet combinant les connaissances nécessaires à l'élaboration d'un site Internet statique à celles devant permettre de structurer l'approche Portfolio.

Dans celle-ci, nous devons prendre notre portfolio comme support de développement et ajouter une interface d'administration permettant de configurer et d'alimenter le portfolio de projets et de contenu multimédia répertoriés dans une base de données. Le site sera donc dynamique et devra implémenter le patron de conception Modèle-Vue-Contrôleur (MVC).

Nous verrons dans ce document les différentes analyses et solutions proposées permettant le développement d'un portfolio dynamique et efficace.

2. Analyses et solutions techniques

1. Modélisation de la base de données

Avant même de commencer le développement de notre site web, nous devons réfléchir en amont sur comment modéliser notre base de données.

Le diagramme entité-association nous permet de conceptualiser simplement les différentes entités et éventuelles relations. Dans le cadre de mon portfolio, voici comment j'ai mis en œuvre ce modèle.

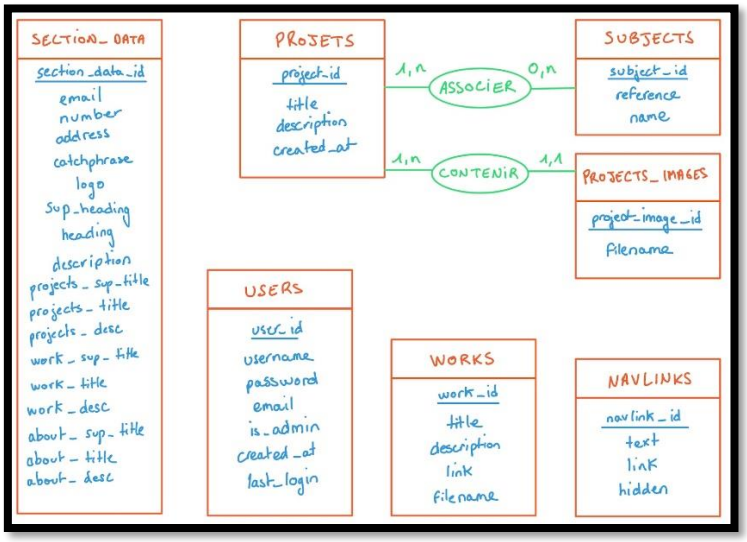


Figure 1 - Diagramme entité-association

La figure 1 représente le schéma conceptuel que j'ai défini pour ma base de données. Il contient plusieurs entités telles que 'Users', 'Projects' ou encore 'Subjects'. Nous retrouvons également des relations entre ces entités comme le montre la figure 2 ci-dessous.

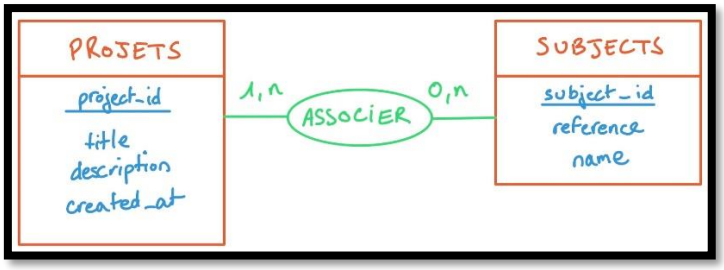


Figure 2 - Exemple d'entité-association

On identifie dans la figure 2 deux entités 'Projects' et 'Subjects' liées par l'association « Associer ».

L'entité 'Projects' contient les attributs *title*, *description* et *created_at* en plus d'un identifiant unique nommé *project_id*.

L'entité 'Subjects' comprend l'identifiant unique, un nom et une référence.

Nous pouvons décrire les entités dans le monde réel par les phrases suivantes :

- Un projet est associé à une matière au minimum (1, n).

A l'inverse:

- Une matière peut être associée à plusieurs projets ou aucun (0, n).

Certaines entités n'ont pour but que le stockage de données et ne sont donc pas soumises à des relations. C'est le cas des entités 'Sections_data', 'Works' ou 'Nav_links'.

A partir du modèle entité-association, nous devons maintenant effectuer le passage au modèle relationnel. Chaque entité devient une relation.

- Sections_data (section_data_id, …)
- Users (user_id, …)
- Works (work_id, …)
- Navlinks (navlink_id, …)
- Projects (project_id, …)
- Projects_images (project_image_id, …, project_id)
- Subjects (subject_id, …)
- Projects_has_subjects (project_id, subject_id)
- ~~Contenir (project_id)~~

Les associations "1,1" n'ont pas d'intérêt à devenir relation, elles sont directement inscrites dans l'entité correspondante.

Maintenant que nous avons fait notre modèle relationnel, nous pouvons créer notre schéma des relations et construire nos tables à l'aide du logiciel MySQL Workbench (figure 3).

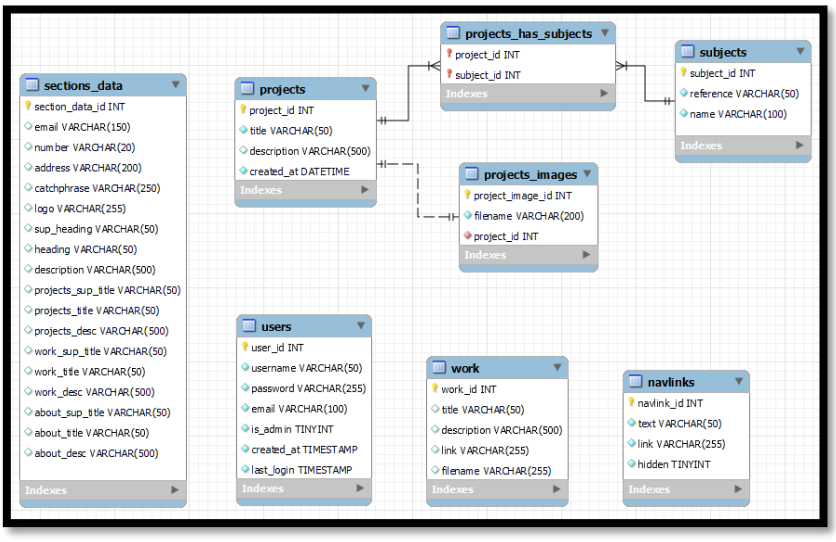


Figure 3 - Schéma MySQL Workbench

2. Implémentation de la base de données

Le serveur de base de données utilisé dans le cadre de ce projet est MySQL, c'est un outil puissant et open source utilisé à plus de 38% dans le web.

Pour implémenter le schéma de données dans le serveur, j'ai utilisé l'outil d'administration web PhpMyAdmin. Avec le plus gros du travail fait en amont, nous avons juste à définir les tables en suivant le schéma relationnel de données.

Lors du développement, ma base de données alimentée au travers d'une interface web simple et sécurisée par une connexion réservée aux administrateurs. Aussi appelé 'BackOffice', cette interface nous permettra de visualiser les entrées dans un tableau ainsi que d'insérer et de modifier les données grâce à un formulaire.

L'intégration d'une interface comme celle-ci nous évite d'avoir à se connecter et modifier les données depuis le système de base de données en lui-même. En outre, ce 'BackOffice' offre plus de sécurité et un confort d'utilisation non négligeable.

3. Développement du patron MVC

Maintenant que notre base de données est mise en place, nous pouvons passer à notre application. Le développement de notre portfolio devra suivre le modèle de conception MVC.

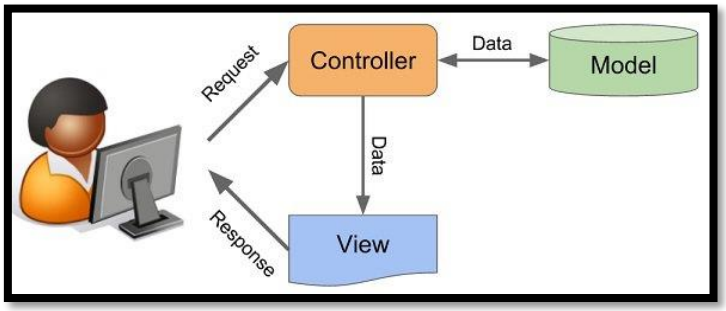


Figure 4 - Fonctionnement du modèle MVC

Comme le montre la figure 4, il permet de séparer les différentes parties de notre application en trois composants majeurs ayant chacuns ses propres fonctions. En plus du MVC, mon site utilise une programmation orientée objet afin de traiter mes données comme des objets. Prenons l'exemple de la classe 'Database' ci-dessous :

```php
class Database {

    private $db;
    private $error;

    protected function dbconnect() {
      // Set DSN
      $dsn = "mysql:host={$_ENV['DB_HOST']};dbname={$_ENV['DB_NAME']};";
      $options = array(
        PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
        PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC
      );
      try {
        // Create a new PDO instance
        $this->db = new PDO($dsn, $_ENV['DB_USER'], $_ENV['DB_PASS'], $options);
        return $this->db;
      }
      catch (PDOException $e) {
        $this->error = $e->getMessage();
        echo 'Database error : '. $this->error;
      }
    }

}
```

Figure 5 - Classe Database

C'est elle qui met en place la connexion avec la base de données avec la méthode 'dbconnect'.

Pour chaque modèle que l'on veut créer, nous devons le lier à la classe 'Database'. Ainsi lors de l'initialisation de celui-ci, nous pouvons simplement appeler la méthode 'dbconnect' comme le montre la figure 6, pour créer une nouvelle instance de 'PDO'.

```php
class UserModel extends Database {

  private $db;

  public function __construct() {
    $this->db = $this->dbconnect();
  }

  public function register($username, $password, $confirm_password, $email) {···
  }

  public function login($username, $password) {···
  }

}
```

Figure 6 - Utilisations de la méthode 'dbconnect'

Après avoir créé une base solide utile au bon développement du modèle MVC, j'ai utilisé la notion de routeur. Le principe du routeur est simple, c'est un carrefour qui renvoie à l'utilisateur le bon contrôleur et la bonne méthode en fonction de l'URL demandé.

Par défaut, la figure 7 ci-dessous nous montre que le routeur est réglé vers la page d'index.

```php
class Router {

  protected $currentController = 'Index';
  protected $currentMethod = 'index';
  protected $params = [];
```

Figure 7 - Paramètres du routeur

Admettons maintenant que nous voulons nous connecter au 'BackOffice', notre URL prendra la forme 'https://portfolio.gelk.studio**/admin**'.

Le routeur va filtrer l'URL et chercher un contrôleur correspondant au fragement de texte '**/admin**'. Voir l'exemple ci-contre.

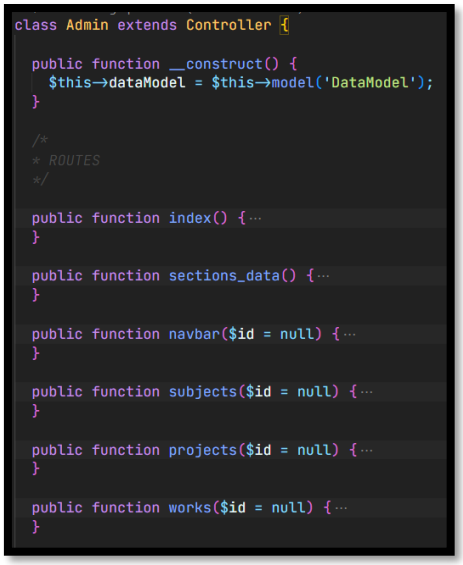
```php
public function __construct() {
  $url = $this->getUrl();

  // Look in controllers for first value
  if (isset($url[0])) {
    if (file_exists('../app/controllers/' . ucwords($url[0]) . '.php')) {
      $this->currentController = ucwords($url[0]);
      unset($url[0]);
    }
  }
  // Require the controller
  require_once '../app/controllers/' . $this->currentController . '.php';
  $this->currentController = new $this->currentController;
```

Figure 8 - Fonctionnement du routeur

S'il trouve un fichier correspondant au contrôleur recherché, il sera inclus dans le fichier et la propriété 'currentController' sera mise à jour avec une nouvelle instance de ce contrôleur.

Ensuite nous pouvons créer nos différentes routes. Voici celles du 'BackOffice'.



```php
class Admin extends Controller {

  public function __construct() {
    $this->dataModel = $this->model('DataModel');
  }

  /*
  * ROUTES
  */

  public function index() {···
  }

  public function sections_data() {···
  }

  public function navbar($id = null) {···
  }

  public function subjects($id = null) {···
  }

  public function projects($id = null) {···
  }

  public function works($id = null) {···
  }
```

Figure 9 - Routes du BackOffice

Essayons d'accéder à la page des projets dans le 'BackOffice'. L'URL aura la forme : **'/admin/projects'**. Après le travail du routeur, la méthode 'projects' ci-dessous est exécutée. Les données sont récupérées avec le modèle puis la vue est chargée avec les données.

```php
public function projects($id = null) {
  $this->checkUserConnection('/login');
  $data = [
    'mode' => isset($id) ? 'edit' : 'read',
    'action' => isset($id) ? 'editProject' : 'addProject',
    'projects' => isset($id) ? $this->dataModel->getProject($id) : $this->dataModel->getProjects(),
    'subjects' => $this->dataModel->getSubjects(),
    'sections_data' => $this->dataModel->getSectionsData(),
  ];
  $this->view('admin/projects', $data);
}
```

Figure 10 - Route de la page projet du BackOffice

L'architecture de mon site se compose de deux dossiers, 'app' et 'public'. Le dossier 'app' comprend tous les fichiers destinés au fonctionnement de notre application. Quant au dossier 'public', il contient les images, les fichiers css et js.

En somme, notre portfolio intègre le modèle MVC et un système de classe. De plus, les requêtes sont gérées par un routeur qui dirige les utilisateurs vers l'URL demandé.

4. Développement du backoffice

Comme nous l'avons vu précédemment, notre 'BackOffice' comporte plusieurs routes menant vers nos pages. Elles sont toutes accessibles par le menu principal du 'BackOffice'.

C'est depuis cette interface que nous pouvons afficher, modifier ou supprimer les éléments de notre base de données.

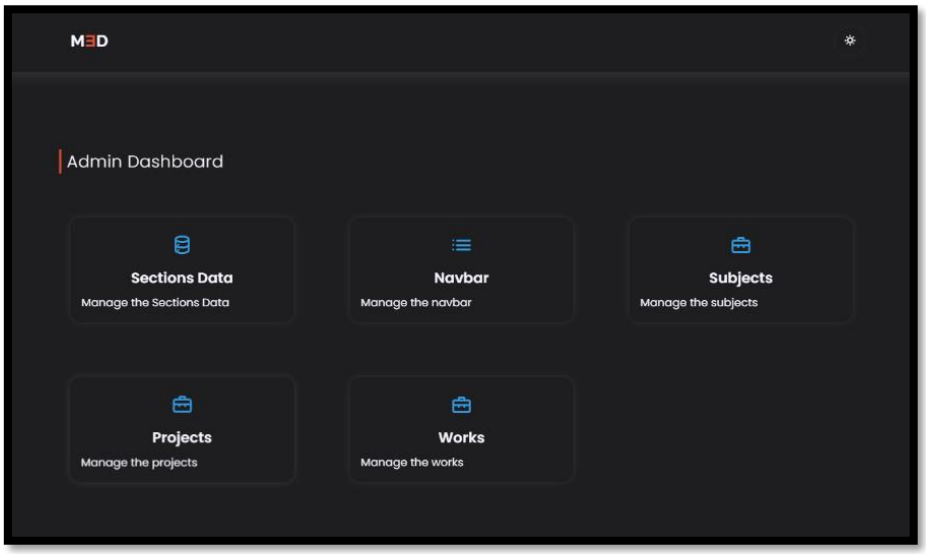


Figure 11 - Menu principal du BackOffice

Prenons l'exemple de la page 'Projects'. Les autres pages se composent de la même façon.

Premièrement, on retrouve un formulaire permettant l'inscription d'une nouvelle donnée. En second, un tableau regroupant toutes les données de la table concernée. La dernière colonne du tableau sert de menu d'action, nous avons le choix entre la modification ou la suppression de la ligne courante.

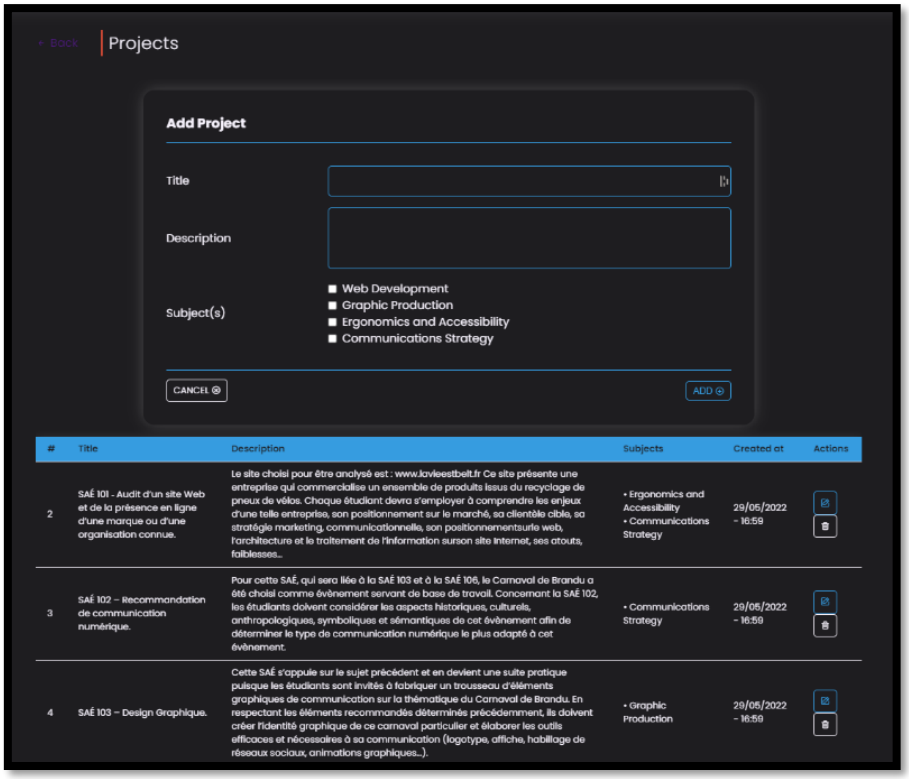


Figure 12 - Page des projets

5. Système de gestion d'utilisateurs

Afin de sécuriser l'accès au 'BackOffice', nous avons besoin d'un système d'utilisateurs et un gestionnaire de sessions. Le gestionnaire de sessions permet de définir un identifiant unique pour chaque utilisateur et est stocké dans un cookie. Il est utilisé pour déterminer si l'utilisateur est connecté ou non. C'est lors de la connexion au 'BackOffice' que l'on définit la session de l'utilisateur. L'accès est uniquement réservé aux utilisateurs ayant les droits d'administrateur.

6. Éléments multimédia

Dans un portfolio, la présence d'éléments multimédia est peut-être un plus pour attirer l'attention au lecteur. Ils offrent un confort de lecture et en disent plus sur la personnalité de l'auteur.

Dans mon cas, j'ai le choix d'inserer des photos d'illustration pour les projets et les travaux depuis le 'BackOffice'.

Les travaux contiennent qu'une seul photo qui s'affiche à coté de la description.

Figure 13 - Sections des travaux personnels

A contrario, les projets peuvent avoir autant de photos que l'on veux. En effet, si nous revenons plus en detail dans la structure de ma base de données, nous pouvons voir dans la figure 14, que j'ai crée une table 'projects_image' ayant un lien direct avec la table des projets.

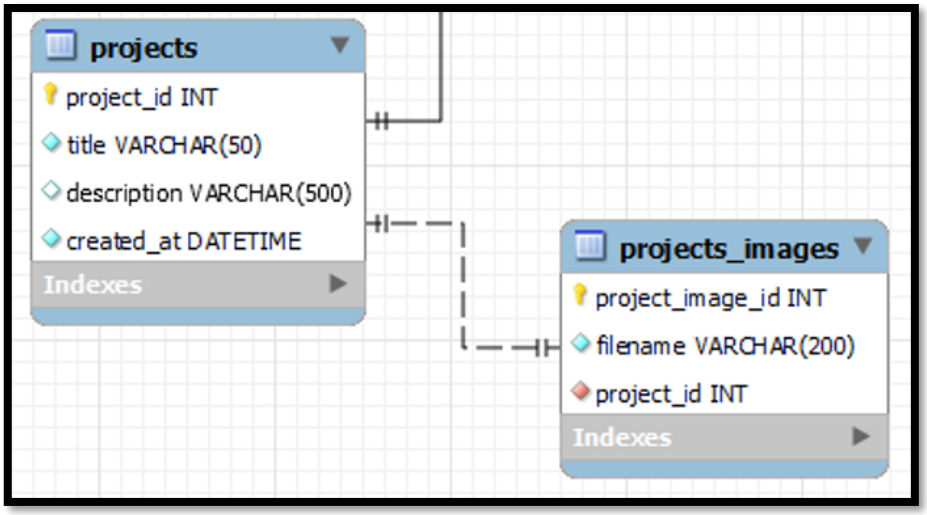


Figure 14 - Lien entre les tables 'projects' et 'projects_images'

Chaque ligne correspond à une image d'un projet identifiable avec son numéro unique. Et donc, si nous voulons plusieurs photo pour un seul projet, il nous suffit d'insérer autant de ligne que l'on souhaite avec le même identifiant de projet.

Ainsi, si nous voulons savoir qu'elles sont les images qui sont liées au projet n°5, nous avons juste à questionner notre base de données et lui demander de nous retourner toutes les lignes pour lesquelles le numéro de projets est égal à 5.

7. Hébergement

Pour le développement et l'hébergement de mon portfolio, j'ai choisi de le déployer sur un serveur linux distant à l'aide de l'hébergeur DigitalOcean afin qu'il soit accessible pour tous. Je me suis chargé de l'installation et la configuration du serveur apache, du serveur MySQL et des certificats SSL permettant la sécurisation et l'encryptions des données. L'acquisition d'un nom de domaine facilite l'identification de mon serveur sur le web.

Vous pouvez retrouver mon portfolio à l'adresse suivante : https://portfolio.gelk.studio.

De plus, pour produire notre site web efficacement, il faut mettre en place un environnement de développement simple et productif.

J'ai choisi d'utiliser l'éditeur Visual Studio Code, qui est un logiciel libre et gratuit. Le développement du site s'effectue directement sur le serveur. A l'aide d'une connexion sécurisée au serveur, je peux développer et visionner mon travail directement depuis le serveur sans devoir le téléverser à chaque modification. De plus, cela me permet de travailler à

distance ou que je sois, sans devoir avoir une machine avec un serveur web et ma base de données.

Je gère mon code source avec Git, un logiciel libre et open source permettant le contrôle de version et la gestion de code source.

3. Conclusion

Pour conclure, nous avons construit et mis en place un site web de type portfolio dynamique utilisant le modèle MVC. De l'élaboration de la base de données jusqu'au déploiement en passant par le développement d'un BackOffice, j'ai fait l'usage de toutes les connaissances acquises durant ces deux semestres d'éducation.

Le gain d'expérience et de professionnalisme m'ont permis de m'améliorer et de proposer un site web de qualité qui me plait vraiment.

De plus, je tiens à remercier les enseignants, qui nous ont permis de rendre accessible, autant que possible, cette première année du BUT MMI.

4. Annexe

Vous retrouverez en annexe les liens de mon portfolio et des sites web utilisés.

- Mon portfolio : https://portfolio.gelk.studio
- BackOffice Portfolio : https://portfolio.gelk.studio/admin

  Compte test pour les enseignants :
  User : **enseignant**
  Pass : **iut20250**

- Dépot Github : https://github.com/gmed2b/mvc-portfolio
- DigitalOcean : https://www.digitalocean.com/
- Validation des normes W3C : https://validator.w3.org/

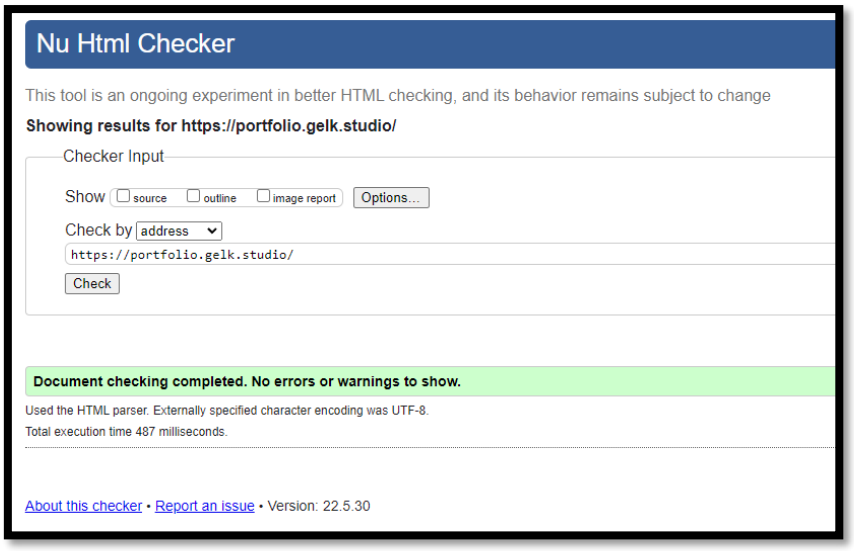


Figure 15 - Validation des normes W3C