



BTS SIO - Option SLAM
Documentation d'épreuve

E4 - Mission 9 : Portfolio

Par Baptiste Grimaldi



E4-Mission9-Portfolio-Baptiste-Grimaldi

Liens utiles

Le site:

(Utiliser un navigateur chromium pour une meilleure expérience)

Portfolio Baptiste Grimaldi

Check out the portfolio of Baptiste Grimaldi, a full-stack web developer specializing in creating dynamic and responsive websites. Explore his projects and contact him for your next web development project.

 <https://portfolio.baptistegrimaldi.info/>

Le repo du projet

<https://github.com/GrimalDev/Portfolio>

session storage and cookie for passport.js

<https://stackoverflow.com/questions/62894933/why-use-cookie-session-in-addition-to-passport-js>



SOMMAIRE

Liens utiles

Le site:

Le repo du projet

1. Introduction

1.1 Présentation du projet de mon portfolio

1.2 Contexte de la mission

2. Expression fonctionnelle du besoin

2.1 Objectifs du portfolio

2.2 Public cible

2.3 Fonctionnalités principales

Analyse du besoin

Liste des analyses fonctionnelles du besoin

Front office:

Back office:

4. Description des environnements

4.1 Stack de technologies utilisé

Couche présentation (front-end)

Couche logique (back-end)

Couche données

4.2 Environnement de production

4.3 Outils utilisés

IDE

Stack en détail

Gestion de la base de données

Ecriture des articles et projets

5. Méthodologie

5.1 Méthodologie de développement

5.2 Gestion de version

Bonnes pratiques

Deux options principales:

Deux exemples de stratégies de branching

5.3 Gestion de tests

5.4 Documentation

6. Mise en oeuvre

6.1 Conception de l'architecture

6.2 Développement du back-end

Fichier `app.js`

Fichiers de routes `.js`

Fichiers d'applications (modèles ou contrôleurs)

Les vues

Dossier public

Système d'authentification

Librairies

En bref

6.3 Développement du front-end

Organisation du CSS

Optimisations

Communication serveur et BDD

6.4 Les grandes fonctionnalités de ce site

7. Gestion de la maintenance corrective et évolutive

8. Bilan du projet

8.1 Validation des exigences point par point

8.2 Axes d'amélioration

9. Conclusion

9.1 Synthèse de la mission

9.2 Perspectives futures

Amélioration du système de recherche

Augmentation du nombre d'articles

Ajout d'une section certifications

1. Introduction

Le but d'un portfolio de développeur web fullstack est de présenter ses compétences, ses projets passés et ses réalisations à des employeurs potentiels, à des clients, à des partenaires et à toute personne qui souhaite en savoir plus sur ses capacités. Le portfolio doit être professionnel, facile à naviguer, accessible et esthétiquement attrayant, et doit inclure les éléments clés tels que les compétences, les projets passés, les réalisations, les expériences professionnelles, ainsi qu'une présentation de la personnalité et des passions du développeur.

```
1. <img_src="logo.svg"> HOME <span>MENU</span>
2. <br>
3. <br>
4. <br>
5. <br>
6. <center><span><h1>PORTFOLIO</h1></span></center>
7. <center><span><h1>GRIMALDI BAPTISTE</h1></span></center>
8. <br>
9. <br>
10. <br>
11. <center><span><h2>Scroll to learn more</h2></span></center>
12. <br>
13. <br>
14. <br>
15. <br>
16. <br>
```

1.1 Présentation du projet de mon portfolio

1.2 Contexte de la mission

Dans le cadre de mon BST SIO, on me demande de réaliser pour mes productions. Ce support est bien plus qu'un rendu d'épreuve. Cela e permet de me mettre en valeur au près des recruteurs.

2. Expression fonctionnelle du besoin

2.1 Objectifs du portfolio

1. **Analyse des besoins** : Il est important de comprendre les besoins du client et de l'utilisateur final afin de déterminer les fonctionnalités et les exigences du projet.
2. **Conception de l'architecture** : Il s'agit de définir l'architecture de l'application, les composants et les technologies nécessaires pour réaliser le projet.
3. **Planification du projet** : Il est important d'établir une liste de tâches, un calendrier et un budget pour s'assurer que le projet est achevé en temps voulu et dans les limites du budget.
4. **Développement du front-end** : La conception de l'interface utilisateur et la mise en œuvre du front-end de l'application peuvent commencer une fois que

l'architecture est en place. Il s'agit de créer une expérience utilisateur intuitive et attrayante.

5. **Développement du back-end** : Le développement du back-end implique la création des fonctionnalités et des services nécessaires pour que l'application fonctionne de manière efficace. Il est important de s'assurer que le back-end est extensible, évolutif et sécurisé.
6. **Tests et débogage** : Une fois le développement terminé, il est temps de tester l'application pour s'assurer qu'elle fonctionne correctement. Les tests devraient être effectués à chaque étape du développement, en utilisant des outils de débogage pour résoudre les problèmes.
7. **Déploiement** : Après avoir achevé les tests et la correction des erreurs, l'application est prête à être déployée. Cette étape implique la mise en place de l'infrastructure nécessaire pour que l'application fonctionne correctement.
8. **Maintenance et mise à jour** : Une fois l'application déployée, il est important de la maintenir régulièrement et de la mettre à jour pour répondre aux besoins changeants des utilisateurs. La maintenance peut inclure la correction de bugs, la mise à jour de la sécurité et l'ajout de nouvelles fonctionnalités.

2.2 Public cible

Le public ciblé sont les personnes cherchant à se renseigner sur mes projets ou connaître mon parcours professionnel. L'interface doit donc être une démonstration rapide de mes capacités. Il faut également prendre en compte que des personnes peuvent se retrouver sur des articles de mon sites par référencement. Il faut donc que mon portfolio puisse également remplir une fonction de simple blog.

Public ciblé

- Les personnes cherchant à se **renseigner** sur les projets et le **parcours professionnel** de l'auteur
- Les personnes qui découvrent le portfolio par **référencement** et qui peuvent être **intéressées par le contenu en général**

2.3 Fonctionnalités principales

Analyse du besoin

Mon besoin est de créer un portfolio personnel en ligne qui me permettra de présenter mes compétences, mes réalisations et mes projets passés. En créant un portfolio, je pourrais présenter mon travail et mes compétences à des employeurs potentiels, des clients, des partenaires ou à toute personne qui souhaite en savoir plus sur moi et mes capacités.

Un portfolio en ligne me permettra de :

- Monter en compétences : en travaillant sur différents projets personnels, j'ai appris de nouvelles compétences. En créant un portfolio, je pourrais consolider ces compétences et mettre en évidence mes réalisations.
- Valoriser mon travail : En tant que développeur, j'ai travaillé sur plusieurs projets par le passé, y compris des projets personnels. En créant un portfolio, je pourrais présenter mes projets terminés et montrer mes compétences en matière de développement web.
- Renforcer ma crédibilité : En publiant des exemples de projets que j'ai réalisés, je pourrais renforcer ma crédibilité auprès des employeurs potentiels, des clients, des partenaires et des autres intervenants.
- Mieux me positionner sur le marché : Un portfolio en ligne me permettra de mieux me positionner sur le marché en mettant en évidence mes compétences, mes réalisations et mes projets passés. Cela me permettra également de me démarquer des autres candidats.
- Faciliter la recherche d'emploi : En ayant un portfolio en ligne, je pourrais facilement partager mes réalisations avec des employeurs potentiels et les aider à mieux comprendre mes compétences et mon potentiel.



Pour répondre à ces besoins, mon portfolio doit être professionnel, facile à naviguer, accessible et esthétiquement attrayant. Les éléments clés de mon portfolio devraient inclure mes compétences, mes projets passés, mes réalisations, mes expériences professionnelles et une présentation de ma personnalité et de mes passions.

En résumé, je souhaite créer un portfolio en ligne pour présenter mes compétences, mes projets passés et mes réalisations à des employeurs potentiels, des clients, des partenaires et à toute personne qui souhaite en savoir plus sur moi et mes capacités. Mon portfolio doit être professionnel, facile à naviguer, accessible et esthétiquement attrayant. Les éléments clés de mon portfolio devraient inclure mes

compétences, mes projets passés, mes réalisations, mes expériences professionnelles et une présentation de ma personnalité et de mes passions.

Nous allons maintenant voir comment je peux refléter ses points clés dans la structure et l'architecture de mon portfolio

Liste des analyses fonctionnelles du besoin

Front office:

La liste des fonctionnalités attendues sont:



Back office:

Articles:

- Ajouter un nouvel article
- Modifier un article existant
- Supprimer un article existant
- Afficher une liste d'articles avec leur titre, leur date de publication et leur auteur
- Rechercher un article par mot-clé ou par date de publication

Projets:

- Ajouter un nouveau projet
- Modifier un projet existant
- Supprimer un projet existant
- Afficher une liste de projets avec leur titre, leur date de création et leur description
- Rechercher un projet par mot-clé ou par date de création

Utilisateur:

- Modifier les informations de l'utilisateur (nom, prénom, adresse e-mail, mot de passe)
- Afficher les informations de l'utilisateur (nom, prénom, adresse e-mail)
- Supprimer le compte de l'utilisateur
- Réinitialiser le mot de passe

Je gère les fonctionnalités de back-office en utilisant une base de données MySQL et un ensemble d'API RESTful pour communiquer avec la base de données.

J'utilise Node.js et Express.js pour créer des API RESTful rapides et évolutives. Les API RESTful sont ensuite consommées par le front-end pour afficher les données dans l'interface utilisateur.

4. Description des environnements

4.1 Stack de technologies utilisé



Le stack de technologies utilisé pour le projet Mediocs suit une architecture trois tiers pour garantir la modularité et la maintenabilité du système.

Couche présentation (front-end)

- **EJS** est utilisé comme langage de modèle pour générer des pages HTML dynamiquement.
- **CSS** est utilisé pour la mise en forme graphique des pages.
- **JavaScript** est utilisé pour la logique front-end.

Couche logique (back-end)

- **Node.js** est utilisé comme environnement d'exécution pour le code JavaScript côté serveur.
- **Express** est utilisé comme framework d'application web pour fournir un ensemble d'outils et d'utilitaires pour la construction d'applications web.
- **Passport** est utilisé comme middleware d'authentification pour Node.js qui fournit un ensemble de stratégies d'authentification pour différents types d'authentification.

Couche données

- **LightSQL** est utilisé pour la gestion de la base de données.

Le choix de cette architecture permet de séparer les différentes couches de l'application pour garantir une évolutivité et une modularité maximales. Chaque couche est responsable de sa propre fonctionnalité, ce qui facilite la maintenance et la mise à jour du système.

4.2 Environnement de production

L'environnement de production est une docker déployée sur mes serveurs.

Voir la documentation de ma mission "[Home-Lab](#)" qui explique comment sont organisés mes serveurs.

J'ai choisi Docker comme environnement de production pour plusieurs raisons. Tout d'abord, cela me permet d'avoir un contrôle total sur les conteneurs via un docker-compose, où je peux définir toutes les propriétés nécessaires.

Ensuite, Docker me permet de relancer facilement un conteneur tout frais en cas de problème, ce qui est très utile pour maintenir la disponibilité de mon application.

Enfin, la conteneurisation offre de nombreux avantages en termes d'optimisation des ressources, ce qui peut m'aider à **améliorer les performances** de mon application tout en **réduisant les coûts d'infrastructure**.

4.3 Outils utilisés

IDE



Pour la programmation de ce site, j'utilise un IDE de JetBrain © fait pour la programmation web et parfaitement optimisé pour le NodeJS:

WebStorm

WebStorm

WebStorm: The Smartest JavaScript IDE by JetBrains



<https://www.jetbrains.com/fr-fr/webstorm/>



C'est un IDE payant avec le quel j'ai obtenue un partenariat avec l'école IRIS de Strasbourg pour des licences étudiantes.

Stack en détail

Le choix des outils pour la réalisation d'un portfolio en tant que développeur web fullstack est crucial pour obtenir une application Web dynamique et performante avec une interface utilisateur intuitive et attrayante. Les outils disponibles pour les développeurs comprennent NodeJs, JavaScript, ThreeJs, PassportJs, MySQL et d'autres bibliothèques.

- **NodeJs** est un environnement d'exécution JavaScript open source qui permet aux développeurs de créer des applications Web hautement évolutives et performantes. Il est basé sur le moteur JavaScript V8 de Google et permet aux

développeurs de créer des applications Web rapides et évolutives. Node.js est également facilement extensible à l'aide de modules NPM (Node Package Manager) qui facilitent le développement de fonctionnalités supplémentaires.

- **JavaScript** est un langage de programmation de premier ordre pour le développement Web. Il permet aux développeurs de créer des pages Web interactives et réactives. Les bibliothèques JavaScript telles que JQuery, AngularJS et ReactJS sont largement utilisées par les développeurs pour créer des interfaces utilisateur riches et réactives.

Pour ce projet j'utilise les dernières normes ES6 en programmation type modules.

- **ThreeJs** est une bibliothèque JavaScript open source pour la création de graphiques 3D en temps réel dans un navigateur Web. Elle est utilisée pour créer des animations 3D interactives, des visualisations de données et des jeux en ligne.
- **PassportJs** est une bibliothèque d'authentification et d'autorisation pour NodeJs. Elle permet aux développeurs de gérer l'authentification des utilisateurs et de contrôler l'accès aux ressources.
- **MySQL** est un système de gestion de base de données open source qui peut être utilisé pour stocker des données de portefeuille. Il est un choix courant pour les développeurs Web. Son désavantage est sa mise à l'échelle uniquement verticale.

Gestion de la base de données

Pour la gestion de la base de données, le très bon outil open-source **PhpMyAdmin** est utilisé.

Par la suite je développerai mon propre outil type CRM.

Ecriture des articles et projets

Pour l'écriture des articles de ce portfolio j'utilise Notion. Notion est un outil formidable avec une très bonne interface utilisateur. Le gros avantage de notion est sa modularité ainsi que son écriture "MarkDown".



Notion est un outil de productivité tout-en-un qui permet de gérer des projets, de prendre des notes, de créer des bases de données, de collaborer avec d'autres personnes et bien plus encore. Il utilise une interface utilisateur intuitive et permet de créer des pages avec différents types de contenus, tels que des listes de tâches, des tableaux, des documents, des calendriers, des galeries d'images et des bases de données.

Les articles sont ensuite intégrés à leurs propres pages de mon portfolio en passant par un moteur d'adaptation que j'ai développé.

5. Méthodologie

En tant que développeur web fullstack, l'adoption de méthodologies et de rigueur est d'une importance capitale. Il est essentiel de choisir une stratégie de versioning appropriée pour le projet en question, ainsi que pour la gestion des tests et de la documentation.

5.1 Méthodologie de développement

Quand on parle de programmation, rien ne vaut la pratique. Lorsqu'on pratique, on remarque très vite l'importance de méthodologies.

On entend beaucoup de méthodes : Agile, Scrum, Agile-Scrum... On parle de Pomodoro et de l'importance de faire des pauses. Rien ne vaut la pratique, d'essayer différentes méthodologies.

Ce qui fonctionne pour moi, c'est un kanban et des sprints. Je me fixe des petits objectifs à sprinter et je garde un suivi de mes tâches avec un kanban sur Notion ou Trello.

Gérez les projets de votre équipe, où que vous soyez | Trello

Trello est l'outil de gestion de projet n° 1. Configurez un tableau en quelques secondes, automatisez les tâches fastidieuses et collaborez où que vous soyez, même sur mobile.

 <https://trello.com/fr>

Your wiki, docs & projects. Together.

A new tool that blends your everyday work apps into one. It's the all-in-one workspace for you and your team.

 <https://www.notion.so/>

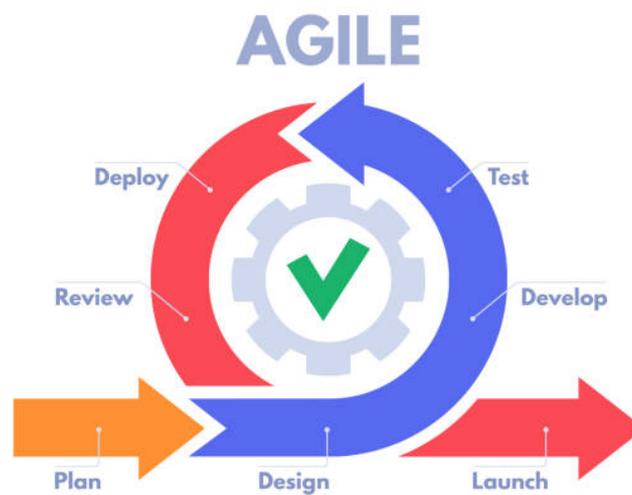


Notion

The all-in-one workspace.
Notes, tasks, wikis, & databases.



Je ne dirais pas que j'utilise une méthodologie SCRUM car je ne suis pas un expert mais je pense m'en rapprocher.

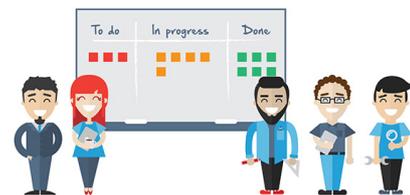


Pour plus information j'invite à lire ce très bon article:

La méthode SCRUM pour les nuls

La méthode SCRUM est l'approche AGILE la plus utilisée dans le monde. Découvrez avec notre article les étapes clés de la méthode, les rôles de chacun et comment mettre en application

 <https://ignition-program.com/tuto/la-methode-scrum-pour-les-nuls>



5.2 Gestion de version

Bonnes pratiques

- ne commitez que des choses sur les mêmes sujets (style, front, back, etc...)

- Si vous ne pouvez pas écrire de messages de commit concis, cela indique trop de sujets dans le même commit.
- Utilisez un titre et un corps avec seulement la commande commit et en ajoutant une ligne vide entre le titre et le corps.

```
git commit
```

```
::
```

```
TITRE
```

```
CORPS
```

```
\# rest comments
```

Stratégies de branching

- Une convention écrite pour organiser l'équipe.

Deux options principales:

1. Développement **Mainline**:

- **quelques** branches
- commits relativement petits
- normes de test de haute qualité
-

2. Branches **State**, **Release** et **Feature**

- **Deux** types de branches différents qui remplissent des types de travail différents

1. **LongRunning**

- existe tout au long de la vie du projet
- souvent, ils reflètent les "étapes" de votre cycle de vie de développement

2. **Short Running**

- pour les nouvelles fonctionnalités, les corrections de bogues, le refactoring, les expériences
- sera supprimé après l'intégration (fusion/rebase)

Deux exemples de stratégies de branching

1. GitHub Flow

- très simple, très léger : seulement long-running
- branche ("main") + branches de fonctionnalités

2. GitFlow

- plus de structure, plus de règles
- long-running : "main" + "develop"
- de courte durée : fonctionnalités, versions, correctifs

Pour la gestion de version de ce projet, j'ai opté pour une stratégie de branching avec un main, un dev et des branches pour les fonctionnalités. Cette méthode de versioning permet de travailler sur plusieurs fonctionnalités en parallèle tout en minimisant les conflits de code.

En résumé, la stratégie de versioning avec un main, un dev et des branches pour les fonctionnalités est efficace pour travailler sur plusieurs fonctionnalités en parallèle tout en minimisant les conflits de code. Les fonctionnalités sont testées avant d'être fusionnées dans la branche principale, ce qui permet de garantir la qualité des fonctionnalités intégrées.



5.3 Gestion de tests



Le test est une phase importante dans le développement web car il permet de s'assurer que le site web ou l'application web fonctionne correctement et répond aux exigences et spécifications. Pendant la phase de test, les développeurs et les testeurs identifient et corrigent les défauts ou bugs dans le code, et vérifient que le site web ou l'application web fonctionne comme prévu.

Etant seul développeur de ce site web, les phases de tests sont relativement rapides. et sont en trois phases.

1. Test avant implémentation du “bug fix”
2. Test après implémentation du “bug fix”
3. Rapport dans la “pull request” de correction de bug

5.4 Documentation

Je tiens à aborder ce point uniquement pour parler des commentaires et autres documentations rédigées autour du projet.



Mis à part ce document, tout est en anglais. La programmation est une langue universelle, alors autant ne pas restreindre la compréhension du code par une langue. Pour moi, cela fait partie du clean code. Je tiens à mettre un point d'honneur là dessus.

6. Mise en oeuvre

6.1 Conception de l'architecture

Quand je commence un projet, j'aime bien mettre les bases de mon projet:

- Organisation de mes dossiers
 - Pour l'organisation de mes dossiers que me base sur du MVC avec un dossier public pour les ressources

- Premier Readme
- organisation des fichiers de projet
 - .gitignore
 - dockerfile
 - .env variables d'environnement avec `dotenv`
- Premier commit

Le modèle MVC (Model-View-Controller) est une architecture de conception qui permet de diviser une application en trois parties principales: le modèle (Model), la vue (View) et le contrôleur (Controller). Le modèle représente la structure de données de l'application, la vue est responsable de l'interface utilisateur et le contrôleur gère les interactions entre le modèle et la vue.

Dans mon projet, j'ai organisé mes fichiers en utilisant le modèle MVC. Le dossier `app` contient les fichiers de configuration, de contrôleurs et de modèles. Le dossier `public` contient les ressources statiques, telles que les images, les fichiers CSS et JavaScript. Le dossier `routes` contient les fichiers de routage qui déterminent la logique de l'application. Enfin, le dossier `views` contient les fichiers de vue au format EJS.

Lorsqu'un utilisateur effectue une demande, le contrôleur correspondant est appelé. Le contrôleur effectue les opérations nécessaires sur le modèle, puis renvoie les résultats à la vue correspondante. La vue utilise les données renvoyées pour afficher les informations à l'utilisateur.

Voici l'architecture de mon projet:

```
├─ app
│  ├── config
│  ├── controllers
│  └── models
├─ public
│  ├── fonts
│  │   ├── Inter
│  │   ├── Roboto
│  │   └── Source_Code_Pro
│  ├── images
│  ├── javascripts
│  ├── media
│  │   └── curriculum
│  ├── model
│  └── stylesheets
```

```
├─ routes
├─ tmp
└─ views
    ├─ imported
    └─ partials
```

Le dossier `app` contient les fichiers de configuration, de contrôleurs et de modèles. Le dossier `public` contient les ressources statiques, telles que les images, les fichiers CSS et JavaScript. Le dossier `routes` contient les fichiers de routage qui déterminent la logique de l'application. Le dossier `views` contient les fichiers de vue au format EJS.

6.2 Développement du back-end

Mon backend de serveur Node.js pour mon projet est organisé de la manière suivante:

Fichier `app.js`

Le fichier `app.js` est le serveur qui se charge d'écouter les requêtes HTTP et de les rediriger en fonction. Il possède toutes les bibliothèques nécessaires pour faire un check des dépendances avant l'exécution du serveur.

Le fichier `app.js` est organisé de la manière suivante :

- Importation des bibliothèques nécessaires
- Définition de la configuration du serveur
- Définition des routes
- Lancement du serveur

Fichiers de routes `.js`

Les fichiers de routes `.js` sont appelés dans le fichier serveur. Ces routes appellent des modèles ou des contrôleurs.

Les fichiers de routes sont organisés de la manière suivante :

- Importation des bibliothèques nécessaires
- Définition des routes
- Exportation des routes

Fichiers d'applications (modèles ou contrôleurs)

Les modèles sont des fichiers cadres avec la logique de certains programmes. Les contrôleurs se chargent quant à eux de la communication avec la base de données.

Les fichiers d'applications (modèles ou contrôleurs) sont organisés de la manière suivante :

- Importation des librairies nécessaires
- Définition des fonctions
- Exportation des fonctions

Les vues

Les affichages sont générés côté client avec une `render`. Ici j'utilise Express avec l'extension de fichiers `.ejs`.

Les vues sont organisées dans le dossier `views`. Les fichiers de vues utilisent le langage de modèle EJS pour générer des pages HTML dynamiques. Les fichiers de vue incluent également des fichiers de style et des fichiers JavaScript pour la mise en forme graphique des pages et la logique front-end.

Dossier public

Le dossier public est défini par le serveur et toutes les ressources sont accessibles depuis ce dossier. Il contient tous les assets.

Le dossier public contient tous les fichiers statiques tels que les images, les fichiers CSS et JavaScript. J'ai organisé ce dossier en sous-dossiers pour faciliter la recherche et la gestion des fichiers.

Système d'authentification

Pour l'authentification, j'ai utilisé une combinaison de cookies et de sessions pour stocker les informations d'identification de l'utilisateur.



Un cookie est un petit fichier texte stocké sur l'ordinateur de l'utilisateur. Les cookies sont envoyés par le serveur au client et stockés sur le disque dur de l'utilisateur. Les cookies sont souvent utilisés pour stocker des informations sur les préférences de l'utilisateur, telles que la langue préférée ou les informations de connexion.

Name	Value	D...	P...	Expires / Max-A...	Size	HttpOnly	Secure	SameSite	Partition ...	Priority
user_sid	s%3AidmEqqr2IoaDmfvNjk6o6UI399SNhs.xk1%2FHLBF12FOHBak9zo2AW2...	p...	/	2023-05-10T11...	90	✓				Medium

Un cookie sur mon site



Une session, quant à elle, est un moyen de stocker des informations sur un utilisateur spécifique de manière persistante. Les sessions sont stockées côté serveur et sont indexées par un identifiant de session unique qui est envoyé au client sous forme de cookie. Ainsi, chaque fois que l'utilisateur effectue une demande, le serveur peut récupérer les informations de session associées à cet identifiant de session pour savoir qui est l'utilisateur.

La combinaison de cookies et de sessions est souvent utilisée pour stocker des informations d'identification de l'utilisateur, car cela permet de stocker les informations de manière persistante et sécurisée, tout en minimisant le temps de latence entre les demandes. Lorsqu'un utilisateur se connecte, le serveur stocke des informations sur l'utilisateur dans une session, qui est stockée dans la base de données.

Lorsque l'utilisateur effectue une demande ultérieure, le serveur récupère les informations de session associées à l'identifiant de session stocké dans le cookie, et utilise ces informations pour déterminer qui est l'utilisateur et s'il est autorisé à accéder à la ressource demandée.

Dans le fichier de configuration de Passport, j'ai défini une fonction de sérialisation qui stocke l'identifiant de l'utilisateur dans la session, et une fonction de désérialisation qui récupère l'identifiant de l'utilisateur à partir de la session. J'ai également créé un objet cookie avec une durée de vie de deux heures, qui est envoyé au client lorsqu'il est authentifié.

Enfin, pour stocker les sessions dans une base de données SQL, j'ai utilisé la bibliothèque `express-mysql-session`, qui fournit une classe pour stocker les sessions dans une table de base de données MySQL. J'ai créé un nouvel objet `sessionStore` en utilisant cette classe, et je l'ai passé à la fonction `session()` d'Express pour stocker les sessions dans la base de données.

Cela permet de stocker les informations d'identification de l'utilisateur de manière persistante et sécurisée, tout en minimisant le temps de latence entre les demandes.

session_id	expires	data	created_at	modified_at
-07ZkxQwoEFNCVdo2TtSF0ZcxckmSUQ	1683712755	{"cookie":{"originalMaxAge":7200000,"expires":"2023-05-10T09:59:14.5..."}}	2023-05-10 07:59:14	0000-00-00 00:00:00
09HyLY51pkn6HWO-pReDtdPEbwJEinjY	1683718755	{"cookie":{"originalMaxAge":7200000,"expires":"2023-05-10T11:39:14.5..."}}	2023-05-10 09:39:14	0000-00-00 00:00:00

une persistance temporaire des sessions

Librairies

Je peux préciser que j'utilise le framework Express.js pour gérer les routes et la logique de l'application. J'ai également choisi d'utiliser la bibliothèque `mysql` (notons que `sequelize` est une option très adéquate aussi) pour la gestion de la base de données MySQL.

Voici des explications supplémentaires sur certaines des librairies utilisées dans ce projet :

- **@sendgrid/mail@7.7.0** : une librairie JavaScript pour envoyer des e-mails avec SendGrid.
- **bcrypt@5.1.0** : une librairie pour le hashage des mots de passe.
- **dotenv@16.0.3** : une librairie pour la gestion des variables d'environnement.
- **ejs@3.1.8** : un moteur de template JavaScript pour les pages web.
- **express-fileupload@1.4.0** : une librairie pour les téléchargements de fichiers dans Express.
- **express-mysql-session@2.1.8** : une librairie pour la gestion des sessions avec MySQL et Express.
- **express-session@1.17.3** : une librairie pour la gestion des sessions avec Express.
- **express@4.18.2** : un framework Web minimaliste pour Node.js.
- **highlight.js@11.7.0** : une librairie pour la coloration syntaxique du code.
- **html-pdf-node@1.0.8** : une librairie pour la création de fichiers PDF à partir de pages HTML.
- **markdown-it@13.0.1** : un analyseur de Markdown pour Node.js.
- **moment@2.29.4** : une librairie pour la manipulation des dates et des heures en JavaScript.
- **morgan@1.9.1** : un middleware pour la gestion des logs dans Express.
- **mysql@2.18.1** : un pilote MySQL pour Node.js.

- **nodemon@2.0.20** : un outil pour le redémarrage automatique de l'application Node.js lors des modifications du code.
- **openai@3.2.1** : une librairie pour l'accès à l'API OpenAI.
- **passport-local@1.0.0** : une stratégie d'authentification locale pour Passport.js.
- **passport@0.6.0** : une librairie pour l'authentification dans Node.js.
- **rss-parser@3.13.0** : une librairie pour l'analyse des flux RSS.
- **three@0.147.0** : une librairie pour la création de graphiques en 3D dans le navigateur.
- **turndown@7.1.2** : une librairie pour la conversion de HTML en Markdown.
- **webpack-cli@5.0.1** : une interface en ligne de commande pour Webpack.
- **webpack@5.78.0** : un module bundler pour JavaScript et ses dépendances.

En bref

Le dossier public contient tous les fichiers statiques tels que les images, les fichiers CSS et JavaScript. J'ai organisé ce dossier en sous-dossiers pour faciliter la recherche et la gestion des fichiers.

J'utilise le framework Express.js pour gérer les routes et la logique de l'application. J'ai également choisi d'utiliser la bibliothèque mysql pour la gestion de la base de données MySQL.

Enfin, j'utilise plusieurs bibliothèques pour des tâches spécifiques tels que l'envoi d'e-mails avec SendGrid, la création de fichiers PDF à partir de pages HTML, la coloration syntaxique du code, la manipulation des dates et des heures en JavaScript, l'analyse des flux RSS, la création de graphiques en 3D dans le navigateur, etc.

6.3 Développement du front-end

Pour mon front-end, j'ai opté pour une approche basique mais efficace. J'utilise le modèle BEM (Block Element Modifier) pour organiser mon code HTML et CSS. Le CSS est écrit en SCSS pour une meilleure lisibilité du code, puis converti en CSS pour une meilleure compatibilité des navigateurs.

Organisation du CSS

Le CSS est organisé avec les fonctionnalités du SCSS.



SCSS (ou SASS) est un préprocesseur CSS qui permet d'écrire du CSS de manière plus structurée et modulaire. Il ajoute des fonctionnalités comme les variables, les mixins, les fonctions, les boucles et les conditions, qui permettent de créer des styles réutilisables et plus facilement maintenables. Le SCSS est ensuite compilé en CSS pour être utilisé par les navigateurs web.

L'organisation du CSS est divisée en trois fichiers principaux:

1. `colors.scss` : fichier contenant les couleurs utilisées dans l'ensemble du site web.
2. `main-style.scss` : fichier principal appelé sur toutes les pages, contenant les styles communs à toutes les pages.
3. Fichiers de styles de page : chaque page a son propre fichier de style, nommé d'après la page correspondante (par exemple, `home-style.scss` pour la page d'accueil).

En outre, il existe des fichiers de style supplémentaires pour certains des éléments récurrents du site web, tels que la barre de navigation (`header-style.scss`) et les styles de formulaire (`login-style.scss`).

Dans `main-style.scss` , les styles sont organisés par blocs, en utilisant la méthode BEM (Block Element Modifier). Les styles sont également organisés en utilisant les fonctionnalités de SCSS, telles que les variables, les mixins et les fonctions.

Optimisations

J'ai également pris soin de mettre en place un listener qui minimise tous les scripts pour optimiser les temps de chargement de la page. J'ai également veillé à ce que toutes les images soient en format web pour une meilleure optimisation.



La minimisation de fichiers JavaScript consiste à réduire la taille des fichiers en supprimant tout code inutile, tel que les commentaires, les espaces blancs et les noms de variables non utilisés. Cela permet de réduire le temps de chargement des pages web et d'améliorer les performances globales du site. **J'utilise UglifyJS.**

Le choix d'utiliser des images en format web est principalement motivé par la taille de ces fichiers. Les images en format web sont généralement plus légères que les images en format PNG ou JPEG, ce qui permet de réduire considérablement le temps de chargement de la page.

Communication serveur et BDD

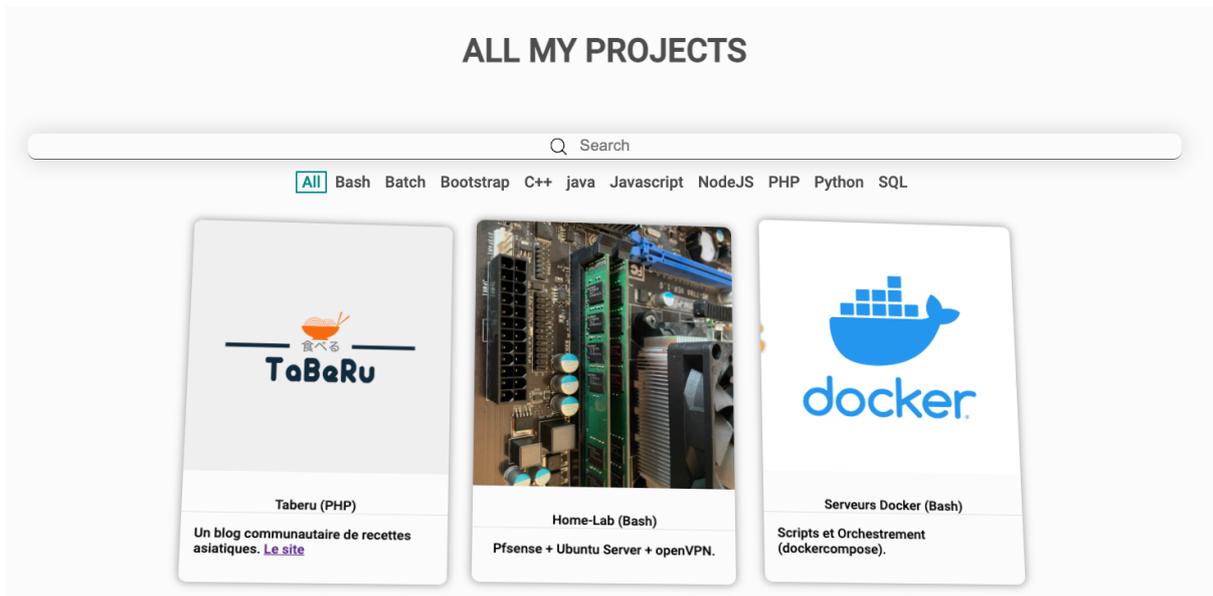
Il est à noter que, pour des raisons évidentes de contrôle du flux de données, toutes les requêtes vers la base de données sont gérées par le serveur via les routes. Cela permet de garantir l'intégrité des données et d'éviter tout risque de compromission de la sécurité en provenance du front-end. En outre, cette méthode permet une gestion plus efficace de la base de données, car elle centralise toutes les requêtes et les traitements de données en un seul endroit.

Cela permet aussi d'avoir un code mieux organiser et plus facile à maintenir.

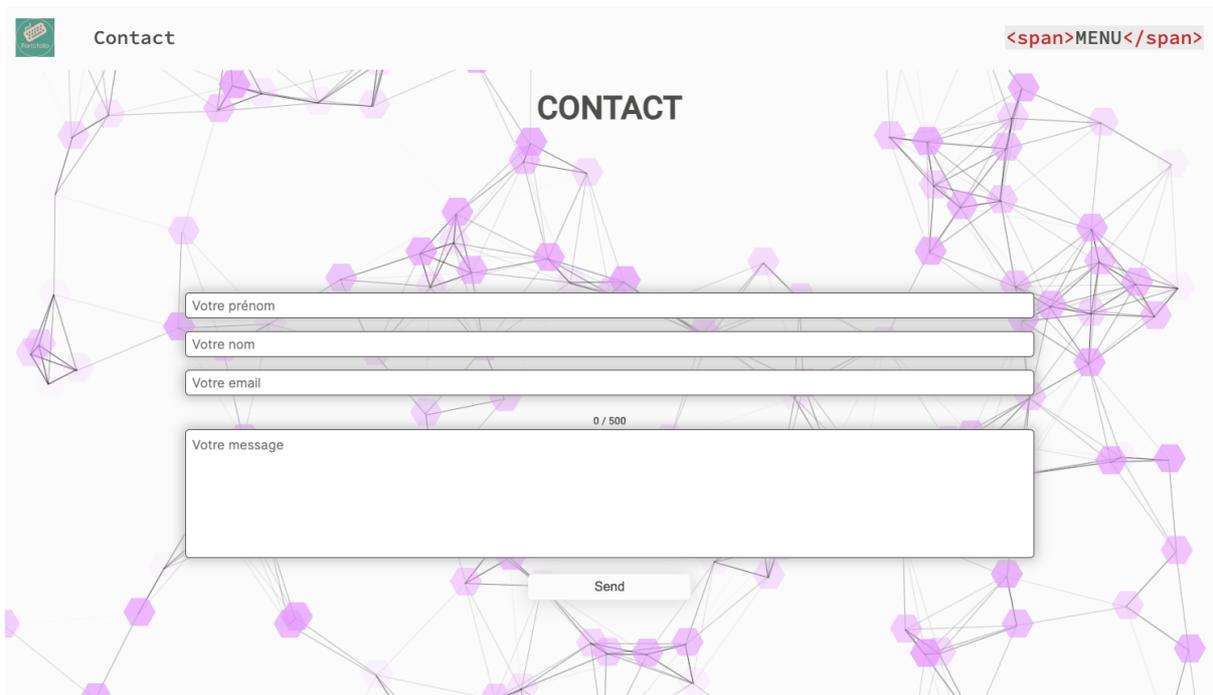
6.4 Les grandes fonctionnalités de ce site

- **Un thème sombre** : (Encore en bêta)
- **La veille automatisé** : J'automatise une partie de mon processus de veille. Je récupère des informations autour de ma veille via des flux RSS et les résumés avec un LLM de OpenAI.
- **Une interface de recherche** : à travers tout mes projets avec les propriétés suivantes
 - Recherche par catégories (une ou plusieurs)
 - Recherche par texte dans le titre ou la description (*Pour l'instant "case sensitive"*)
 - Recherche mixtes (catégories plus texte)
 - Système de pagination
- **Une page de contact** : Relier à un système de mails vers la boîte ma contact. contact@baptistegrimaldi.info
- **Une interface admin** : (très basique)
 - Permet de visualiser les logs
 - Permet d'ajouter un article avec un corp en html ou non et ajouter un pdf à télécharger.

- **L'authentification** est gérée avec `PassportJS` et une combinaison de session et de cookie.



Recherche d'articles



Page de contact

7. Gestion de la maintenance corrective et évolutive

Pour assurer la maintenance corrective de mon projet, j'ai mis en place un processus de qualité (QA) pour garantir que toutes les erreurs et les bugs sont corrigés rapidement et efficacement.

Tout d'abord, j'ai inclus un **formulaire de contact** sur mon site pour permettre aux utilisateurs de signaler tout problème qu'ils rencontrent. Ce formulaire est relié à ma boîte mail de contact, ce qui me permet de recevoir rapidement toutes les plaintes et de les traiter en conséquence.

Ensuite, j'utilise **Github** pour gérer toutes les mises à jour et les modifications de mon projet. Je peux facilement suivre toutes les modifications apportées à mon code et les tester avant de les déployer sur mon site en production. En utilisant des branches et des pull requests, je suis en mesure de séparer les fonctionnalités en cours de développement de celles qui sont prêtes à être déployées.

Enfin, j'ai mis en place des **tests automatiques** pour m'assurer que toutes les fonctionnalités de mon projet fonctionnent correctement avant de les déployer en production. Les tests couvrent tous les aspects de mon projet, de la logique de l'application aux interfaces utilisateur.



En utilisant ces méthodes, je suis en mesure de garantir que mon projet est toujours à jour et que tous les problèmes sont corrigés rapidement et efficacement.

8. Bilan du projet

8.1 Validation des exigences point par point

- Gérer le patrimoine informatique: Le document décrit la manière dont le projet a été organisé et structuré, ainsi que les bibliothèques et les outils utilisés pour le développement du site web. Il explique également comment les problèmes ont été abordés et résolus.
- Répondre aux incidents et aux demandes d'assistance et d'évolution: Le document inclut une section sur la gestion de la maintenance corrective et évolutive, qui décrit comment les erreurs et les bugs sont corrigés et comment les mises à jour et les modifications sont gérées.
- Développer la présence en ligne de l'organisation: Le document décrit comment le site web a été conçu et développé pour améliorer la présence en ligne de

l'organisation. Il détaille les fonctionnalités du site web, ainsi que les outils et les bibliothèques utilisés pour développer ces fonctionnalités.

- **Travailler en mode projet:** Le document décrit comment le projet a été organisé et structuré, ainsi que les outils et les méthodes utilisés pour le développement du site web. Il décrit également comment les problèmes ont été abordés et résolus.
- **Mettre à disposition des utilisateurs un service informatique:** Le document décrit comment le site web a été conçu et développé pour fournir un service aux utilisateurs. Il décrit les fonctionnalités du site web, ainsi que les outils et les bibliothèques utilisés pour développer ces fonctionnalités.
- **Organiser son développement professionnel:** Le document décrit comment le projet a été organisé et structuré, ainsi que les outils et les méthodes utilisés pour le développement du site web. Il décrit également comment les problèmes ont été abordés et résolus, et comment le processus de qualité a été mis en place pour assurer la maintenance corrective et évolutive du site web.

8.2 Axes d'amélioration

Lorsqu'on développe une application seul, on est souvent confronté à un manque d'organisation.

Avec le temps, je compte devenir de plus en plus à l'aise avec l'organisation lors de la mise en œuvre de mes projets, mais c'est définitivement un axe d'amélioration.

9. Conclusion

9.1 Synthèse de la mission

J'ai réalisé une mission de développement web pour le projet E4-Mission9-Portfolio-Baptiste-Grimaldi. Dans le cadre de cette mission, j'ai conçu et développé un site web permettant d'améliorer la présence en ligne de l'organisation en fournissant un service aux utilisateurs.

Pour ce faire, j'ai utilisé le framework Express.js pour gérer les routes et la logique de l'application, ainsi que la bibliothèque mysql pour la gestion de la base de données MySQL. J'ai également utilisé plusieurs autres bibliothèques pour des tâches spécifiques telles que l'envoi d'e-mails avec SendGrid, la création de fichiers PDF à partir de pages HTML, la coloration syntaxique du code, la manipulation des

dates et des heures en JavaScript, l'analyse des flux RSS, la création de graphiques en 3D dans le navigateur, etc.

Le site web comprend plusieurs fonctionnalités telles qu'un thème sombre, la veille automatisée, une interface de recherche, une page de contact, une interface admin et l'authentification gérée avec PassportJS et une combinaison de session et de cookie.

J'ai mis en place un processus de qualité (QA) pour garantir que toutes les erreurs et les bugs sont corrigés rapidement et efficacement. J'ai inclus un formulaire de contact sur mon site pour permettre aux utilisateurs de signaler tout problème qu'ils rencontrent, j'utilise Github pour gérer toutes les mises à jour et les modifications de mon projet, et j'ai mis en place des tests automatiques pour m'assurer que toutes les fonctionnalités de mon projet fonctionnent correctement avant de les déployer en production.

L'organisation du CSS est divisée en trois fichiers principaux : `colors.scss` , `main-style.scss` et les fichiers de styles de page. Les styles sont organisés par blocs, en utilisant la méthode BEM (Block Element Modifier), et en utilisant les fonctionnalités de SCSS, telles que les variables, les mixins et les fonctions.

Enfin, pour la gestion de l'authentification, j'ai utilisé une combinaison de cookies et de sessions pour stocker les informations d'identification de l'utilisateur, stockées dans une base de données SQL avec la bibliothèque `express-mysql-session` . Cette méthode permet de stocker les informations de manière persistante et sécurisée, tout en minimisant le temps de latence entre les demandes.

L'ensemble de ces fonctionnalités et méthodes ont permis de développer un site web performant, sécurisé et facilement maintenable. Cependant, j'ai identifié l'organisation comme un axe d'amélioration pour mes projets futurs.

9.2 Perspectives futures

Je vois plusieurs axes d'amélioration pour mon projet Portfolio sachant que je compte sur lui tout au long de ma carrière.

Amélioration du système de recherche

Je prévois d'améliorer le système de recherche en ajoutant une barre de recherche disponible à tout moment sur le site pour chercher à travers tous les articles et projets. Cela permettra aux utilisateurs de trouver rapidement et facilement les informations qu'ils recherchent.

Augmentation du nombre d'articles

Je compte également augmenter le nombre d'articles sur mon site pour offrir plus de contenu aux utilisateurs. Je prévois de publier régulièrement de nouveaux articles pour continuer à améliorer la présence en ligne de mon organisation.

Ajout d'une section certifications

Je prévois également d'ajouter une section certifications pour mettre en avant mes compétences et mes réalisations. Cette section permettra aux utilisateurs de mieux comprendre mes qualifications et mes réalisations professionnelles.

En travaillant sur ces améliorations, je suis convaincu que mon site web continuera à s'améliorer et à offrir une expérience utilisateur de qualité aux utilisateurs.



Cette documentation est la propriété intellectuelle de Grimaldi Baptiste.
portfolio.baptistegrimaldi.info/legal

Avis de droits d'auteur

Informations légales sur les droits d'auteur de cette documentation

Cette documentation a été créée par moi, Grimaldi Baptiste, en tant qu'étudiant. Tous les droits d'auteur sur cette documentation sont réservés. Aucune partie de cette documentation ne peut être reproduite, stockée dans un système de récupération ou transmise sous quelque forme ou par quelque moyen que ce soit, électronique, mécanique, photocopie, enregistrement ou autre, sans l'autorisation préalable écrite de l'auteur.

Toute utilisation non autorisée de cette documentation peut constituer une violation des lois sur les droits d'auteur et entraîner des poursuites judiciaires.

Si vous souhaitez utiliser cette documentation à des fins éducatives ou autres, veuillez contacter l'auteur pour obtenir l'autorisation écrite nécessaire.

Copyright Grimaldi Baptiste, 2023.

Par Baptiste Grimaldi