Software Architecture Document (SAD)

1. Introduction

Objectif du SAD : Décrire l'architecture logicielle du générateur dynamique de site web. Ce document servira de guide pour comprendre la structure technique du projet, couvrant le frontend, backend, base de données, et les spécifications API.

Contexte : Ce générateur dynamique permet aux utilisateurs de personnaliser leur site web en choisissant des templates, en configurant une identité visuelle, et en ajoutant des composants. L'architecture logicielle garantit une intégration fluide entre les différentes couches de l'application.

2. Description Générale de l'Architecture

Frontend :

- Technologies : HTML, CSS, JavaScript, AJAX.
- Rôle : Le frontend est chargé d'afficher l'interface utilisateur et d'offrir une expérience fluide et réactive. AJAX est utilisé pour les communications asynchrones avec le backend, afin de récupérer les données sans recharger la page.

Backend :

- Technologie : Python (Flask).
- Rôle : Le backend gère la logique métier et les communications entre le frontend et la base de données. Flask est utilisé pour créer et gérer les API, gérer les sessions utilisateur et interagir avec MongoDB.

Base de Données :

- Technologie : MongoDB.
- Rôle : MongoDB stocke les informations de l'utilisateur, les configurations de site, les templates et les composants ajoutés. Sa flexibilité permet de gérer facilement des données semi-structurées.

Authentification :

- Gestion des sessions : Flask et MongoDB assurent la connexion des utilisateurs et la persistance des sessions.

3. Spécifications Techniques et Structure de l'API

Structure de l'API : Le backend expose plusieurs endpoints API pour gérer les utilisateurs, les configurations de site et les composants. Voici les principaux endpoints :

- - POST /api/auth/login : Authentification des utilisateurs.
- - GET /api/templates : Récupère les templates disponibles pour la personnalisation.
- - POST /api/configure : Sauvegarde les configurations utilisateur, incluant l'identité visuelle, le choix de template, et les composants ajoutés.

Base de données MongoDB :

- - Collection `users` : Stocke les informations d'authentification, profils utilisateur, et préférences de configuration.
- - Collection `templates` : Contient les différents templates de mise en page, y compris les métadonnées.
- - Collection `components` : Enregistre les composants additionnels ajoutés par les utilisateurs.

4. Flux de Données et Interactions

Connexion et Authentification : L'utilisateur se connecte ou crée un compte via l'API /api/auth/login. Après vérification, une session est ouverte pour l'utilisateur.

Choix de l'Identité Visuelle et des Templates : Lors de la personnalisation, le frontend envoie une requête AJAX au backend pour récupérer les templates disponibles. L'utilisateur choisit ensuite les couleurs et les logos, et les informations sont envoyées au backend pour sauvegarde.

Ajout et Gestion de Composants : Les utilisateurs peuvent ajouter des composants comme des sections de texte, des images, ou des boutons via une interface interactive. Chaque composant ajouté est sauvegardé dans MongoDB via l'API /api/configure.

Sauvegarde et Publication : Une fois la configuration terminée, l'utilisateur peut enregistrer le site. MongoDB stocke l'ensemble des préférences et de la structure du site.

5. Environnement de Développement et Déploiement

Outils de Gestion : Git pour le contrôle de version, avec intégration continue via GitHub. Trello est utilisé pour le suivi des tâches, conformément aux livrables.

Déploiement : Le projet peut être hébergé sur une plateforme compatible avec Flask et MongoDB, avec des options d'hébergement comme Heroku ou d'autres services cloud.