Documentation

## **Introduction**

### Objectif du document

### Contexte du projet

### Portée du projet

## **Description du système**

### Vue d'ensemble du système

### Diagrammes et architecture du système

### Description détaillée des composants du système

## **Spécifications du système**

### Spécifications fonctionnelles

### Spécifications non fonctionnelles (sécurité, performance, etc.)

## **Plan de mise en œuvre**

### Calendrier du projet

### Ressources nécessaires

### Risques et atténuation

## **Test et validation**

### Stratégie de test

## **Maintenance et support**

## **Conclusion**

## Introduction

### 1.1 Objectif du document

L'objectif de ce document est de fournir une description détaillée du projet de contrôle des panneaux d'affichage. Il décrit le contexte, les spécifications, le plan de mise en œuvre, les stratégies de test et de validation, et le plan de maintenance et de support pour le système. Le document servira de référence pour toutes les parties prenantes du projet.

### 1.2 Contexte du projet

La partie concernée par ce projet est le système d'affichage qui vise à informer le public. Actuellement, le système envoie automatiquement trois informations deux fois par jour : le débit entrant, le débit sortant et le niveau de l'eau. L'objectif est de reproduire ce fonctionnement avec une communication GPRS / 4G aussi sécurisée que possible, tout en permettant l'envoi d'images.

### 1.3 Portée du projet

Le projet comprend le développement de l'application web, des applications embarquées, la configuration du serveur, la mise en place de mesures de sécurité appropriées, le test du système et le déploiement de l'application sur le serveur et les contrôleurs embarqués dans le panneau. Le système sera capable de basculer entre trois modes d'affichage : affichage d'image, affichage de données et boucle d'affichage. Le système sera également capable de planifier l'envoi de données et d'envoyer des images pour des messages de communication personnalisés ou pour l'affichage de médias.

### **2. Description du système**

#### 2.1 Vue d'ensemble du système

Le système est conçu pour fournir une solution d'affichage robuste et sécurisée pour un barrage. Il se compose de plusieurs composants clés qui interagissent pour fournir des fonctionnalités d'affichage dynamique et sécurisé. Ces composants incluent des panneaux d'affichage, un serveur dédié, une application web, un réseau VPN et plusieurs mesures de sécurité.

#### 2.2 Description détaillée des composants du système

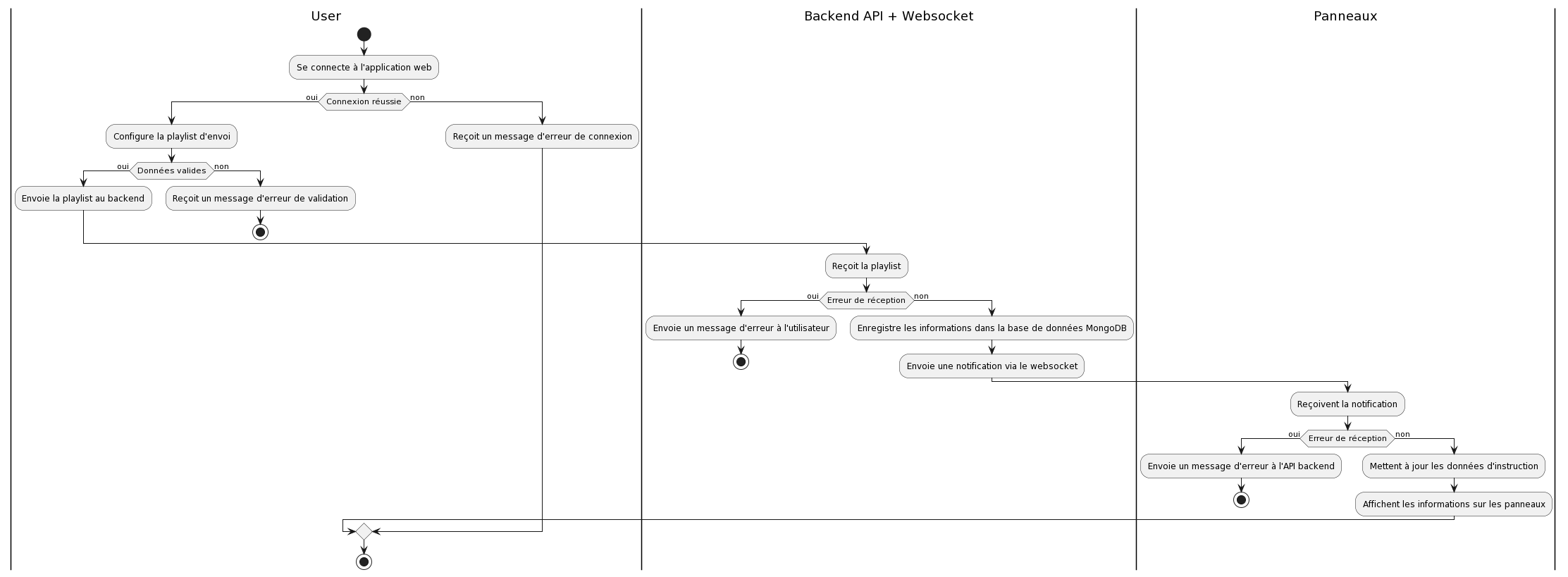
* **Application web** : L'application web est conçue en utilisant la pile technologique MERN qui se compose de MongoDB, Express.js, React.js et Node.js.
  + MongoDB : Système de gestion de base de données utilisé. Il est responsable du stockage des instructions destinées aux panneaux d'affichage ainsi que des identifiants et des mots de passe des utilisateurs.
  + Express.js : Framework de serveur web basé sur Node.js, utilisé pour construire le backend de l'application web.
  + React.js : Bibliothèque JavaScript pour la construction d'interfaces utilisateur. Il est utilisé pour développer le frontend de l'application web, offrant une expérience utilisateur interactive.
  + Node.js : Environnement d'exécution côté serveur pour JavaScript. Il est utilisé pour exécuter l'application web.
* **Serveur** : Le serveur du système tourne sur le système d'exploitation Debian. Ce système d'exploitation a été choisi en raison de sa robustesse, de sa fiabilité et de son excellente sécurité. Le serveur héberge la base de données MongoDB ainsi que l'application web.
* **VPN** : Le système utilise un réseau VPN pour sécuriser la communication entre le serveur et les panneaux d'affichage. Cela assure que les informations sensibles ne sont pas exposées et que le système est à l'abri des attaques externes.
* **Panneaux d'affichage** : Deux panneaux d'affichage qui sont équipés d'un routeur 4G et d'une Asus TinkerBoard. Ces panneaux affichent des informations sur le barrage : le débit entrant, le débit sortant et le niveau de l'eau, ainsi que des messages de communication personnalisés. Les panneaux d'affichage sont pilotés par des TinkerBoard qui exécutent des applications développées à l'aide du Framework Electron.js.
* Application embarquée : Cette application, développée sous Electronjs, est chargée de gérer l'affichage des informations sur les panneaux. Electron.js permet de construire des applications de bureau robustes en utilisant des technologies web standard comme JavaScript, HTML et CSS.
* **Communication** : La communication entre l'application web et les panneaux d'affichage est réalisée en utilisant le protocole WebSocket Secure (WSS). Le WSS offre une communication bidirectionnelle en temps réel entre le client et le serveur sur une connexion sécurisée. La sécurisation de la connexion est garantie par l'emploi d'un réseau VPN.
* **Mesures de sécurité** : Diverses mesures de sécurité sont mises en place pour assurer la protection des données et la stabilité du système :
  + Pare-feu UFW : Limiter l'accès au serveur, en permettant uniquement les communications entrantes sur le port 443.
  + Certificat SSL : Chiffrer la connexion entre le serveur et les clients, assurant ainsi la confidentialité et l'intégrité des données échangées.
  + Reverse Proxy Nginx : Configuré avec SSL/TLS pour fournir une couche supplémentaire de sécurité, notamment en bloquant l'accès par IP non autorisées.
  + Authentification par jeton d'accès : Sécuriser la communication entre les panneaux d'affichage et le serveur.

#### Diagramme d'architecture du système

**A diagram of a computer

Description automatically generated with low confidence**

* 1. Diagramme d’activité



1. **Début** : Le processus commence par l'authentification de l'utilisateur dans l'application web.
2. **Entrée des identifiants** : L'utilisateur entre son nom d'utilisateur et son mot de passe.
3. **Vérification des identifiants** : Le système vérifie si les identifiants entrés par l'utilisateur correspondent à un compte enregistré.
4. **Échec de l'authentification** : Si les identifiants sont incorrects, l'utilisateur est invité à réessayer.
5. **Succès de l'authentification** : Si les identifiants sont corrects, l'utilisateur accède à la page de configuration de l'affichage.
6. **Configuration de l'affichage** : L'utilisateur configure l'affichage des deux panneaux. Les options de configuration incluent les données fournies par le barrage (débits et cote) ainsi que la possibilité d'ajouter des images pour des communications ou des médias.
7. **Validation de la configuration** : L'utilisateur confirme la configuration et l'envoie.
8. **Envoi de données** : L'application envoie les informations configurées aux deux panneaux via une connexion sécurisée.
9. **Vérification de la réception des données** : Les panneaux vérifient la réception des données.
10. **Échec de la réception des données** : En cas d'échec de la réception des données, une alerte est envoyée à l'utilisateur.
11. **Succès de la réception des données** : Si les données sont reçues avec succès, elles sont affichées sur les panneaux.
12. **Fin** : Le processus se termine après l'affichage des données.
    1. Matrice de Flux

A screenshot of a computer

Description automatically generated with medium confidence

* 1. Maquette fonctionnelle de l’application non définitive (WIP)A screenshot of a computer

     Description automatically generated with medium confidence
  2. Diagramme de séquenceA diagram of a server

     Description automatically generated with low confidence

**Utilisateur** : L'utilisateur démarre le processus en s'authentifiant dans l'application web.

**Système** : Le système vérifie les identifiants de l'utilisateur.

**Utilisateur** : Si l'authentification réussit, l'utilisateur accède à la page de configuration de l'affichage et configure l'affichage des deux panneaux.

**Système** : Le système reçoit la configuration de l'utilisateur et la valide.

**Utilisateur** : L'utilisateur confirme la configuration et l'envoie.

**Système** : Le système envoie les informations configurées aux deux panneaux via une connexion sécurisée.

**Panneau 1 et Panneau 2** : Les panneaux reçoivent les données et affichent les informations.

* 1. Diagramme de classe

A picture containing text, diagram, screenshot, font

Description automatically generated

**3. Spécifications du système**3.1 Spécifications fonctionnelles

* Authentification : L'application doit permettre à un utilisateur de se connecter en utilisant des identifiants sécurisés.
* Configuration de l’affichage : Une fois connecté, l'utilisateur doit pouvoir configurer l'affichage des deux panneaux. Les options de configuration devraient inclure les données fournies par le barrage (débits et cote) ainsi que la possibilité d'ajouter des images pour des communications ou des médias.
* Envoi de données : L'application doit être capable d'envoyer les informations configurées aux deux panneaux via une connexion sécurisée.

3.2 Spécifications non fonctionnelles

* Performance : Bien que l'application ne soit destinée qu'à un utilisateur simultané maximum, elle doit être optimisée pour minimiser l'utilisation des données et éviter des temps de réponse excessifs. Cela est particulièrement important compte tenu du fait que la communication se fait via la 4G.
* Sécurité : L'application doit être hautement sécurisée pour prévenir toute attaque potentielle. Les mesures de sécurité devraient inclure l'utilisation d'un VPN, l'obtention d'un certificat SSL d'une autorité de certification, l'utilisation d'un pare-feu, et d'autres mesures appropriées.
* Maintenabilité : L'application doit être conçue de manière à faciliter la maintenance à long terme.

4. Plan de mise en œuvre

* 1. Calendrier du projet
* Définition des spécifications détaillées, Conception et Développement de l’API **(20 mars - 21 mai : 2 mois)**
  + Définition des spécifications détaillées (20 mars - 31 mars : 2 semaines)
  + Conception de l'architecture du système et modélisation des bases de données (1 avril - 30 avril : 1 mois)
  + Développement de l’API (1 mai - 21 mai : 3 semaines)
* Développement de l'application web **(22 mai - 30 juin : 1 mois et 1 semaine)**
  + Développement des fonctionnalités de base de l'application web (22 mai - 31 mai : 1 semaine)
  + Développement des fonctionnalités avancées et finalisation de l'application web (1 juin - 30 juin : 1 mois)
* Développement de l'application embarquée pour les Tinkerboards (1 juillet - 23 juillet : 3 semaines)
  + Développement de l'application embarquée (1 juillet - 15 juillet : 2 semaines)
  + Tests et corrections de l'application embarquée (16 juillet - 23 juillet : 1 semaine)
* Intégration, Tests et Ajustements (24 juillet - 31 juillet : 1 semaine)
  + Intégration de toutes les composantes du système (24 juillet - 26 juillet : 3 jours)
  + Tests du système intégré et ajustements (27 juillet - 31 juillet : 5 jours)
* Tests de sécurité, Optimisations et Déploiement du VPN (1er août - 13 août : 2 semaines)
  + Tests de sécurité et optimisations (1er août - 6 août : 1 semaine)
  + Déploiement du VPN, pare-feu et autres mesures de sécurité (7 août - 13 août : 1 semaine)
* Tests finaux, Ajustements et Déploiement (À partir du 14 août)
  + Tests finaux et ajustements (14 août - 20 août : 1 semaine)
  + Déploiement du système (À partir du 21 août)
  1. Organisation

**Epic 1: Préparation du projet**

* Feature 1.1 : Définition des spécifications détaillées
  + User Story 1.1.1 : Comprendre les besoins du client
  + User Story 1.1.2 : Rédiger les spécifications
  + User Story 1.1.3 : Valider les spécifications avec le client
* Feature 1.2 : Conception de l'architecture du système
  + User Story 1.2.1 : Définir l'architecture logicielle
  + User Story 1.2.2 : Définir l'architecture matérielle
  + User Story 1.2.3 : Documenter l'architecture
* Feature 1.3 : Modélisation des bases de données
  + User Story 1.3.1 : Identifier les entités et relations
  + User Story 1.3.2 : Créer le schéma de base de données
* Feature 1.4 : Développement de l'API
  + User Story 1.4.1 : Identifier les endpoints nécessaires
  + User Story 1.4.2 : Créer les endpoints
  + User Story 1.4.3 : Tester les endpoints

**Epic 2: Développement de l'application**

* Feature 2.1 : Développement de l'application web
  + User Story 2.1.1 : Créer la maquette de l'interface utilisateur
  + User Story 2.1.2 : Implémenter l'interface utilisateur
  + User Story 2.1.3 : Intégrer l'API
* Feature 2.2 : Développement de l'application embarquée pour les Tinkerboards
  + User Story 2.2.1 : Écrire le code pour interagir avec le matériel
  + User Story 2.2.2 : Intégrer l'API

**Epic 3: Intégration et tests**

* Feature 3.1 : Intégration de toutes les composantes du système
  + User Story 3.1.1 : Déployer les différentes parties du système ensemble
  + User Story 3.1.2 : Résoudre les problèmes d'intégration
* Feature 3.2 : Tests et ajustements
  + User Story 3.2.1 : Tester le système intégré
  + User Story 3.2.2 : Apporter les ajustements nécessaires

**Epic 4: Sécurité et déploiement**

* Feature 4.1 : Tests de sécurité, optimisations et déploiement du VPN, pare-feu et autres mesures de sécurité
  + User Story 4.1.1 : Effectuer les tests de sécurité
  + User Story 4.1.2 : Optimiser le système
  + User Story 4.1.3 : Déployer les mesures de sécurité
* Feature 4.2 : Tests finaux, ajustements et déploiement
  + User Story 4.2.1 : Effectuer les tests finaux
  + User Story 4.2.2 : Apporter les ajustements finaux
  1. Ressources nécessaires
* Une machine serveur pour héberger l'application web et la base de données MongoDB.
* Deux Tinkerboards pour gérer les affichages.
* Deux routeurs 4G pour la communication avec les panneaux d'affichage.
  1. Risques et atténuation
* Problèmes de sécurité : Tests de sécurité réguliers, l'utilisation de meilleures pratiques de sécurité et la mise en place de mesures de sécurité robustes (VPN, pare-feu, etc.).
* Problèmes de performance : Tests de performance réguliers et une optimisation continue peuvent aider à prévenir les problèmes de performance.

5. Test et validation

* 1. Stratégie de test
* Des tests unitaires seront effectués sur toutes les fonctions critiques du code pour s'assurer qu'elles fonctionnent comme prévu.
* Les tests d'intégration seront effectués pour s'assurer que toutes les parties du système fonctionnent correctement ensemble.
* Des tests de performance seront effectués pour s'assurer que le système peut gérer les charges attendues.
* Des tests de sécurité seront effectués pour s'assurer que le système est suffisamment sécurisé.

6. Maintenance et support

Une maintenance régulière du système pourra être planifiée pour assurer la mise à jour des composants du système, l'optimisation des performances et le renforcement de la sécurité.

7. Conclusion

Le projet vise à développer un système d'affichage numérique robuste et sécurisé pour un barrage. Ce système permettra à un utilisateur de contrôler l'affichage de deux panneaux d'information via une application web. Les données à afficher seront stockées de manière éphémère dans une base de données et transmises via un réseau sécurisé.