Solution Building Blocks (SBBs) Document

Introduction

Le document des Solution Building Blocks (SBBs) vise à décomposer l'architecture cible en composants distincts et réutilisables, chacun répondant à une description détaillée des besoins en spécifications fonctionnelles et la démarche pour l'implémenter.

---

1. Front-End Block

1. **Building block name :** Front-End Block
2. **Functionality provided :** Développer une interface utilisateur réactive avec une navigation fluide, intégrant des mécanismes de sécurité pour l'accès aux services backend, permettant la gestion des comptes utilisateurs et supportant les plateformes desktop et mobile.
3. **Link to example implementation or interfaces :**
    - Framework : React.js
    - Outil de gestion de projet : Jira
    - Outil de contrôle de version : GitHub
    - Outil de conception d'interface : Figma
4. **Outstanding work to complete this building block :**
    - Implémenter les composants réutilisables avec React Hooks et Context API.
    - Configurer et exécuter les tests avec Jest, Enzyme et Cypress.
    - Conteneuriser l'application frontend avec Docker et déployer avec Netlify ou Vercel.
5. **Architectural alignment :**
    - **Objective 1 :** Assurer une interface utilisateur intuitive et réactive.
        - **Principle 1** : Utilisation de frameworks modernes et de composants réutilisables selon l'approche micro-frontend.
    - **Objective 2 :** Garantir la sécurité des accès aux services backend.
        - **Principle 2 :** Implémentation de mécanismes de sécurité robustes comme le 2FA en respectant les principes OAUTH2.0.

---

2. Back-End Block

1. **Building block name** : Back-End Block
2. **Functionality provided :** Gérer les requêtes des utilisateurs, assurer la gestion des données, implémenter des mécanismes d'authentification et d'autorisation, et maintenir l'état des sessions de manière sécurisée.
3. **Link to example implementation or interfaces :**
    - Langage de programmation : Java
    - Framework : Spring Boot
    - Documentation API : Swagger
4. **Outstanding work to complete this building block :**
    - Implémenter les microservices avec Spring Boot.
    - Configurer la base de données PostgreSQL.
    - Documenter les APIs RESTful avec Swagger.
    - Implémenter l'authentification et l'autorisation avec Spring Security et JWT.
    - Configurer Docker et Kubernetes pour le déploiement.
5. **Architectural alignment :**
    - **Objective 1** : Assurer une gestion efficace des requêtes et des données.
        - **Principle 1 :** Utilisation de microservices SpringBoot et de bases de données robustes PostgreSQL.
    - O**bjective 2 :** Garantir la sécurité des sessions et des données.
        - **Principle 2 :** Implémentation de mécanismes de sécurité avancés comme le 2FA.

---

3. Data Management Block

1. **Building block name : Data Management Block**
2. **Functionality provided :** Assurer le stockage sécurisé des données, supporter les transactions ACID, mettre en place des mécanismes de sauvegarde et de restauration, et contrôler l'accès aux données.
3. **Link to example implementation or interfaces :**
    - Base de données relationnelle : PostgreSQL

- Outil de gestion : pgAdmin
- Sauvegarde : AWS S3, AWS Backup

4. **Outstanding work to complete this building block :**
   - Configurer PostgreSQL pour les transactions ACID.
   - Configurer MongoDB pour les données semi-structurées.
   - Implémenter le chiffrement des données au repos et en transit.
   - Planifier et tester les sauvegardes automatiques avec AWS S3 et AWS Backup et PGBackRest.
   - Implémenter les rôles et permissions avec PostgreSQL RLS.
5. **Architectural alignment :**
   - **Objective 1 :** Assurer la sécurité et la fiabilité des données.
     - **Principle 1 :** Utilisation de bases de données sécurisées et fiables.
   - **Objective 2 :** Garantir la disponibilité et la récupération des données.
     - **Principle 2 :** Implémentation de mécanismes de sauvegarde et de restauration efficaces.

---

4. Security Block

1. **Building block name** : Security Block
2. **Functionality provided :** Implémenter des systèmes robustes pour l'authentification et l'autorisation, assurer le chiffrement des données, détecter et prévenir les intrusions, et garantir la conformité aux normes de sécurité.
3. **Link to example implementation or interfaces :**
   - Chiffrement : SSL/TLS, AES
   - Authentification : OAuth2, OpenID Connect
   - Détection d'intrusions : Snort, Suricata
   - Conformité : TrustArc, OneTrust
4. **Outstanding work to complete this building block :**
   - Configurer SSL/TLS pour le chiffrement des données en transit.
   - Implémenter OAuth2 et OpenID Connect pour la gestion des identités et des accès.
   - Mettre en place des solutions de sécurité comme Snort ou Suricata.
   - Utiliser TrustArc ou OneTrust pour gérer la conformité au RGPD.
   - Réaliser des tests de pénétration réguliers avec OWASP ZAP.

5. **Architectural alignment :**
    - **Objective 1 :** Assurer une authentification et une autorisation robustes.
        - **Principle 1 :** Utilisation de standards éprouvés pour l'authentification et l'autorisation.
    - **Objective 2 :** Garantir la sécurité et la conformité des données.
        - **Principle 2** : Implémentation de mécanismes de chiffrement et de détection des intrusions.

---

5. API Gateway Block

1. **Building block name** : API Gateway Block
2. **Functionality provided** : Faciliter les appels API entre les services, sécuriser les APIs, et gérer le routage et le formatage des messages.
3. **Link to example implementation or interfaces :**
    - API Gateway : Kong Gateway
    - Sécurité : OAuth2, JWT
    - Tests : Postman, SoapUI
4. **Outstanding work to complete this building block :**
    - Configurer Kong Gateway pour la gestion des appels API.
    - Implémenter des politiques de sécurité avec OAuth2 et JWT.
    - Configurer des règles de routage et de transformation des messages avec Kong.
    - Tester les APIs avec Postman et SoapUI.
    - Déployer Kong avec des configurations automatisées.
5. **Architectural alignment :**
    - **Objective 1 :** Assurer une gestion efficace des appels API.
        - **Principle 1 :** Utilisation d'une API Gateway robuste et scalable.
    - **Objective 2 :** Garantir la sécurité des APIs.
        - **Principle 2 :** Implémentation de politiques de sécurité avancées.