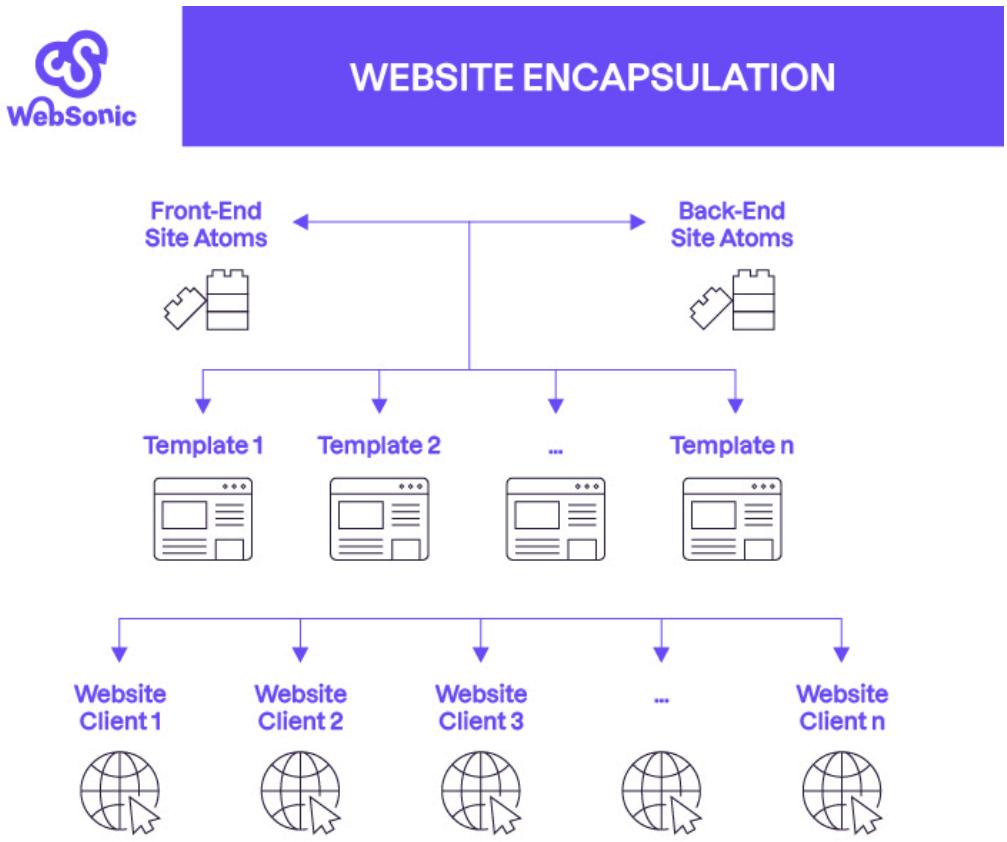
Nouvelle architecture

1. Contexte :

1. Architecture existante :



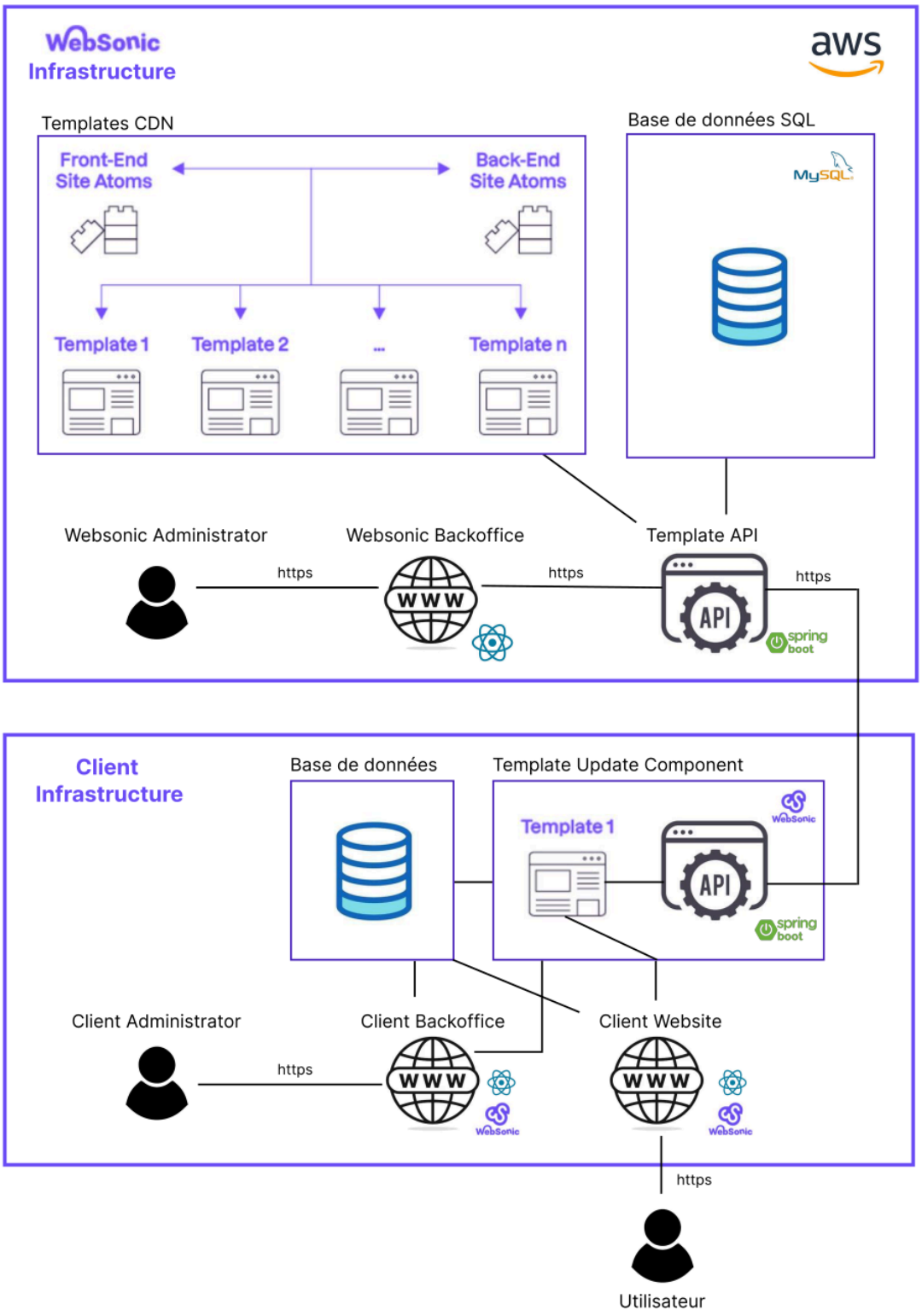
2. Analyse de l’écart entre l'architecture existante et l’architecture cible :

Élément	Architecture existante	Nouvelle Architecture
Modularité des atoms	Les *Front-End Site Atoms* et *Back-End Site Atoms* sont bien structurés, favorisant la réutilisabilité et la modularité.	Conserver la modularité et l’organisation actuelle des *Front-End Site Atoms* et *Back-End Site Atoms*

Templates	Les templates permettent de démarrer rapidement avec des solutions prêtes à l'emploi sans fonctionnalité majeure pour la personnalisation.	Permettre une personnalisation avancée des templates via une interface utilisateur dans le backoffice Websonic/Client.
Base de données SQL	Aucune informations concernant une base de données	Ajouter une base de données SQL ou réutiliser la base de données existante.
Infrastructure et CDN	Un CDN est mis en place pour distribuer les ressources partagées comme les templates et atoms.	Conserver l'utilisation du CDN pour distribuer les templates et atoms.
Gestion des sites clients	Les sites clients et l'ensemble des données associées sont centralisés chez WebSonic.	Intégrer une API pour communiquer entre l'infrastructure client/Websonic et synchroniser les templates . Les données propres à chaque site seront gérées par le client
Gestion de mise à jour	L'architecture actuelle ne semble pas gérer les mises à jour des sites web clients de façon automatique	Intégrer un composant de mise à jour automatique qui se connectera à l'infrastructure de Websonic pour gérer la mises à jours des templates et atoms

1. Architecture cible

1. Schéma :



2. Explications des éléments de l’architecture cible

1. **Infrastructure** Websonic:

Templates CDN :
Stocke et distribue des modèles (templates, atoms) via un réseau optimisé pour fournir les ressources aux clients rapidement et efficacement.

Base de données SQL :
Une base relationnelle utilisée pour stocker les informations structurées, comme les utilisateurs, les droits, les offres et les données spécifiques à Websonic.

Template API :
Fournit une interface pour gérer les templates et atoms. Les applications clientes peuvent interagir avec cette API pour récupérer ou mettre à jour des templates.

Websonic Backoffice :
Les administrateurs de Websonic ont accès au backoffice pour gérer la création, la suppression ou la mise à jour des templates et atoms.

2. **Infrastructure** Client:

Template Update Component :
Une fonctionnalité installée sur le serveur du client permettant de mettre à jour ou personnaliser les templates existants. C’est l’élément qui fait la connexion entre l’infrastructure de Websonic et celle du client.

Base de données :
Une base de données utilisée pour stocker les informations du site web du client.

Client Backoffice :
Les administrateurs clients peuvent configurer, tester, customiser les templates et régler les options de mise à jour.

Client Website :
Le site web public utilisé par les utilisateurs finaux pour accéder aux services du client de Websonic.

3. Justifications des technologies

1. **AWS (Infrastructure)** :

- ➢ **Fiabilité et scalabilité** : AWS offre une infrastructure flexible capable de s'adapter aux variations de charge. Les services comme CloudFront garantissent une disponibilité continue.
- ➢ **Documentation** : AWS offre une documentation riche et de nombreuses ressources sont disponibles pour garantir la mise en place de l'infrastructure de façon optimale.
- ➢ **Coût ajustable** : Les prix à l'usage permettent une gestion optimale des coûts pour des besoins variables.
- ➢ **Sécurité** : AWS est certifié pour plusieurs normes de sécurité, assurant la protection des données.

2. **MySQL (Base de données)** :

- ➢ **Simplicité et robustesse** : MySQL est une solution mature, adaptée pour des applications nécessitant des transactions fiables et des relations complexes entre données.
- ➢ **Large communauté** : L'écosystème étendu facilite la maintenance et l'accès à des ressources externes.
- ➢ **Compatibilité** : Intégration aisée avec des frameworks tels que Spring Boot.

3. **React (Front-end)** :

- ➢ **Interactivité** : Permet de construire des interfaces utilisateur dynamiques et réactives, essentielles pour un site moderne.
- ➢ **Écosystème riche** : La bibliothèque offre des outils performants comme Redux pour la gestion d'état, et une large gamme de composants.
- ➢ **Réutilisabilité** : Les composants React peuvent être partagés entre différents projets ou sections (backoffice et site public).

4. **Java Spring Boot (Back-end)** :

- ➢ **Modularité** : Le framework est idéal pour construire des services API robustes et évolutifs.
- ➢ **Sécurité intégrée** : Avec des modules comme Spring Security, il est facile de sécuriser les endpoints et d'implémenter des rôles utilisateur.
- ➢ **Support de bases relationnelles** : Spring Data JPA facilite les interactions avec MySQL, réduisant le temps de développement.