# Choix Techniques

## Backend:

- **Node.js et Express.js**: Ces technologies sont choisies pour leur efficacité dans la création d'APIs robustes et rapides. Express.js offre une architecture légère et flexible, idéale pour le développement d'applications web.
- **PostgreSQL**: En tant que système de gestion de base de données relationnelles, PostgreSQL est réputé pour sa fiabilité, sa conformité aux normes et ses fonctionnalités avancées, ce qui en fait un choix solide pour stocker les données de l'application.
- **bcrypt pour le hachage des mots de passe**: bcrypt est une bibliothèque de hachage de mots de passe bien établie et sécurisée, utilisée pour protéger les mots de passe utilisateur en les rendant difficiles à décrypter même en cas de violation de données.
- **JSON Web Tokens (JWT)**: JWT est un standard ouvert (RFC 7519) qui permet de créer des jetons d'authentification compacts et autonomes. Il est utilisé pour la gestion des sessions utilisateur de manière sécurisée et efficace.
- **CORS (Cross-Origin Resource Sharing):** Cela permet de sécuriser l'accès aux ressources du serveur backend en autorisant uniquement les domaines de confiance à interagir avec l'API. La configuration CORS permet uniquement les requêtes provenant du domaine spécifié, renforçant ainsi la sécurité de l'application.

## Frontend:

- **Vue.js**: Vue.js est un framework JavaScript progressif utilisé pour la construction d'interfaces utilisateur interactives et dynamiques. Il offre une approche déclarative et composante pour le développement frontend, ce qui facilite la création et la maintenance de l'interface utilisateur de l'application Wishlist.
- **Vuex**: Vuex est une bibliothèque de gestion d'état pour les applications Vue.js. Il offre un moyen centralisé de gérer l'état de l'application et facilite la communication entre les composants.
- **Axios**: Axios est un client HTTP basé sur les promesses utilisé pour effectuer des requêtes HTTP depuis le navigateur ou Node.js. Il est utilisé pour communiquer avec le backend de manière asynchrone et efficace.

## Base de Données:

- **PostgreSQL Docker Image**: L'utilisation d'une image Docker PostgreSQL simplifie le déploiement et la gestion de la base de données. En encapsulant PostgreSQL dans un conteneur, il devient facile de déployer la base de données de manière cohérente et reproductible sur différents environnements.

## Insertion Script:

- **Python**: Python est choisi pour sa simplicité et sa polyvalence. Il est largement utilisé dans le développement d'applications web et offre une variété de bibliothèques pour interagir avec les bases de données, ce qui en fait un choix naturel pour le script d'insertion de données.
- **psycopg2-binary**: psycopg2 est une bibliothèque Python populaire pour la connexion aux bases de données PostgreSQL. Son support binaire binaire (psycopg2-binary) est préférable pour les installations dans des environnements conteneurisés en raison de sa facilité d'installation et de sa légèreté.

# Sécurité

## Authentification:

- **bcrypt pour le hachage des mots de passe**: Le hachage des mots de passe avec bcrypt garantit que même en cas de violation de données, les mots de passe des utilisateurs restent sécurisés. Les fonctions de hachage lentes et les saltages aléatoires intégrés à bcrypt rendent les attaques par force brute et par dictionnaire très difficiles.
- **JSON Web Tokens (JWT)**: Les JWT sont utilisés pour la gestion des sessions utilisateur. Ils contiennent des informations d'authentification signées cryptographiquement, ce qui garantit leur intégrité. Seuls les serveurs disposant de la clé secrète peuvent décoder et vérifier la validité des JWT, ce qui les rend sécurisés pour l'authentification des utilisateurs.

## Gestion des Ports:

- **Ports Non-standard pour Frontend et Backend**: En utilisant des ports non-standard pour le frontend (8085) et le backend (5000), nous réduisons les risques de conflits avec d'autres services fonctionnant sur les ports par défaut. Cela ajoute une couche de sécurité en minimisant les attaques potentielles par rebond.

## Configuration Sécurisée:

- **Stockage des Informations Sensibles dans les Variables d'Environnement**: Les informations sensibles telles que les identifiants de base de données et les clés secrètes JWT sont stockées en tant que variables d'environnement. Cela empêche leur exposition accidentelle dans les dépôts de code source et réduit les risques de violation de données.

## Séparation des Réseaux avec Docker Compose:

- **Réseaux Docker Séparés pour Chaque Composant**: En configurant Docker Compose pour créer des réseaux séparés pour chaque composant (frontend, backend, base de données, script d'insertion), nous limitons la communication entre les composants aux seuls réseaux autorisés. Cela réduit la surface d'attaque en empêchant l'accès non autorisé entre les composants.