1. **Ancienne architecture monolithique de l'application de véhicules chez Propelize**:

* Dans une architecture monolithique, l'application est conçue comme une seule unité unifiée. Tous les composants et fonctions sont étroitement imbriqués dans une base de code unique.
* L'application de location de véhicules de Propelize avait probablement une application mobile ou un site web comme point d'entrée pour les utilisateurs.
* Cette partie front-end envoyait des requêtes à un serveur dorsal qui gérait tout : commandes, paiements, recherche de véhicules, etc.
* Derrière cela, il y avait probablement une base de données (peut-être un serveur SQL) ou un système d'entreprise (comme SAP ou Salesforce).
* Cependant, cette approche présente des inconvénients à mesure que l'application grandit et évolue.

1. **Inconvénients de l'architecture monolithique**:

* **Complexité de la base de code** : Plus l'application grandit, plus la base de code devient complexe. Cela rend difficile l'apport de modifications et allonge les cycles de construction et de déploiement.
* **Difficulté d'intégration des nouvelles technologies** : Les monolithes sont étroitement liés à une pile technologique spécifique. Intégrer de nouvelles technologies ou mettre à jour les existantes peut nécessiter une réécriture importante de l'application.
* **Fiabilité limitée** : Un bogue ou un problème dans une partie du système peut affecter l'ensemble de l'application, entraînant des pannes importantes.
* **Maintenance complexe** : Au fil du temps, l'application peut devenir difficile à maintenir, avec une base de code enchevêtrée et peu lisible, surtout pour les nouveaux développeurs.

1. **Architecture microservices proposée pour l'application de location de véhicules**:

* Les microservices sont une alternative à l'architecture monolithique. Ils découpent l'application en petits services indépendants, gérant chacun une fonction spécifique.
* Voici comment Propelize pourrait adopter une architecture de microservices :
* **Service de recherche de véhicules** : Un microservice gère la recherche de véhicules disponibles.
* **Service de réservation** : Un autre microservice gère les réservations et les disponibilités.
* **Service de paiement** : Un microservice gère les transactions financières.
* **Service de gestion des utilisateurs** : Gère les profils, les authentifications, etc.
* Chaque microservice peut être développé, déployé et mis à l'échelle indépendamment.
* La communication entre les microservices se fait via des API RESTful ou d'autres mécanismes.
* Cette approche facilite la maintenance, l'évolutivité et l'intégration de nouvelles fonctionnalités.