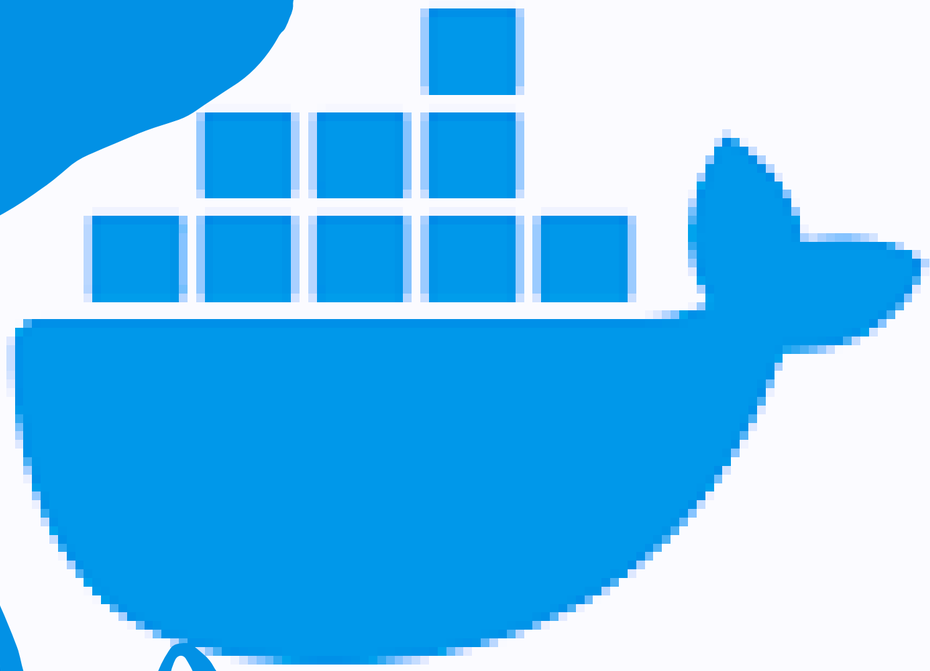
ARCHITECTURE LOGICIELLE

TP2_PRINCIPES FONDAMENTAUX DES MICROSERVICES : CONSTRUIRE UNE API



docker

GROUPE_8 :

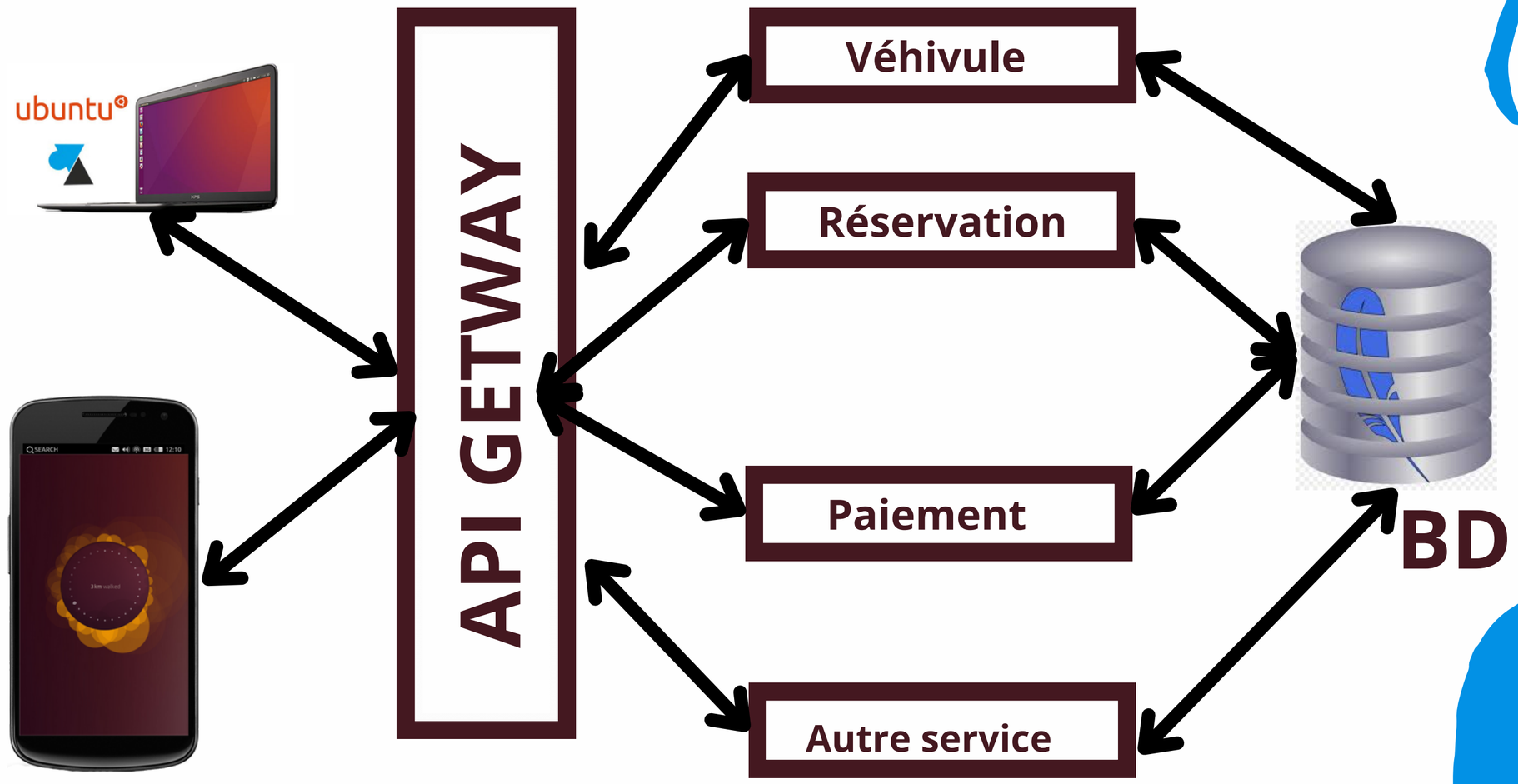
INTRODUCTION

Les architectures microservices gagnent en popularité ces dernières années, car elles offrent de nombreux avantages par rapport aux architectures monolithiques traditionnelles. En décomposant une application en une série de services autonomes et indépendants, les microservices permettent une plus grande flexibilité, une meilleure scalabilité et une maintenance simplifiée.

OBJECTIF DE PROPLIZE

Dans le cadre du projet de l'entreprise Propelize, qui se lance dans l'activité de location de véhicules, une architecture microservices a été choisie pour construire la nouvelle application. Celle-ci permettra aux utilisateurs de rechercher, de réserver et de gérer la location de voitures et de camionnettes.

REPRÉSENTATION DE L'ANCIENNE ARCHITECTURE DE L'APPLICATION DE VÉHICULE DE PROPELIZE



INCONVÉNIENTS DE L'ARCHITECTURE MONOLITHIQUE DE PROPELIZE

👉 **Complexité de la base de code qui augmente avec la croissance de l'application, rendant les modifications et les mises à jour plus difficiles.**

👉 **Cycles de construction et de déploiement plus longs, ralentissant le processus de développement, surtout pour les grandes équipes.**

👉 **Forte dépendance à une pile technologique spécifique, rendant l'intégration de nouvelles technologies ou la mise à jour des technologies existantes plus difficile.**

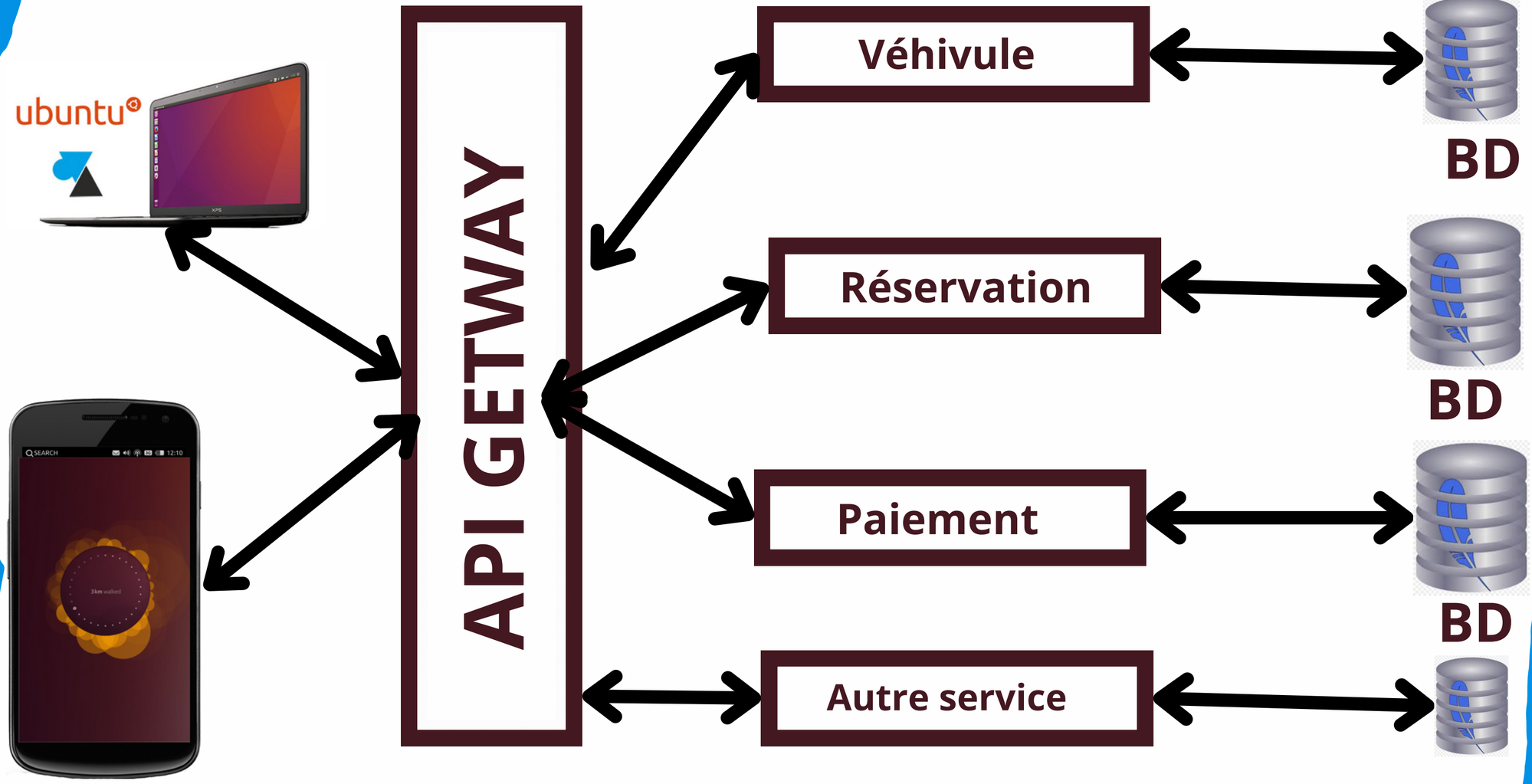
INCONVÉNIENTS DE L'ARCHITECTURE MONOLITHIQUE DE PROPELIZE

👉**Fiabilité réduite, car un bogue ou un problème dans une partie du système peut potentiellement faire tomber l'ensemble de l'application.**

👉 **Maintenabilité difficile à cause de la complexité de la base de code et du manque de modularité.**

👉 **Compréhension du code plus difficile, surtout pour les nouveaux développeurs rejoignant le projet.**

PROPOSITION D'UNE ARCHITECTURE MICROSERVICE POUR L'APPLICATION DE LOCATION DE VÉHICULES



PROPOSITION D'UNE ARCHITECTURE MICROSERVICE POUR L'APPLICATION DE LOCATION DE VÉHICULES

Pour l'application de location de véhicules de Propelize, une architecture microservices serait plus adaptée. Cette architecture impliquerait de décomposer l'application monolithique en services autonomes et indépendants, chacun responsable d'une fonctionnalité spécifique, tels que la gestion des véhicules, la gestion des réservations, la gestion des paiements, etc. Ces services communiqueraient entre eux via des API bien définies, permettant une plus grande flexibilité, une meilleure maintenabilité et une fiabilité accrue.

CHOIX DU STACK TECHNOLOGIQUE POUR L'IMPLÉMENTATION DU SERVICE DE GESTION DES VÉHICULES

Pour le service de gestion des véhicules, un stack technologique basé sur Django, Django REST Framework et SQLite3 serait un bon choix.

AVANTAGE DE CE STACK

Django est un framework Python robuste et complet pour le développement web

Django REST Framework facilite la construction d'APIs RESTful

SQLite3 est une base de données légère et simple à mettre en place, bien adaptée aux besoins de l'application

Conception de l'API

Compréhension des exigences métier

L'application doit permettre aux utilisateurs de rechercher, de réserver et de gérer la location de véhicules (voitures et camionnettes).

Les fonctionnalités principales sont : la gestion des véhicules, la gestion des réservations, la gestion des paiements.

Conception de l'API

Identification de la cible de l'application

L'application cible les particuliers et les entreprises souhaitant louer des véhicules.

Conception de l'API

Établissement des objectifs de cette application

Fournir une expérience utilisateur fluide et intuitive pour la location de véhicules.

Offrir une gestion efficace et sécurisée des véhicules, des réservations et des paiements.

Permettre une évolution et une maintenance simplifiées de l'application grâce à une architecture microservices.

Exploration du processus de conception

Identification des ressources et des actions

Ressources principales : véhicules, réservations, paiements

Actions : créer, lire, mettre à jour, supprimer les véhicules, les réservations et les paiements

Exploration du processus de conception

Choix des principes de conception (REST)

Conception d'une API RESTful pour permettre une intégration fluide avec les applications client (web, mobile).

Exploration du processus de conception

Définition des endpoints

GET /vehicles : Récupérer la liste de tous les véhicules

GET /vehicles/{id} : Récupérer les détails d'un véhicule spécifique

POST /vehicles : Créer un nouveau véhicule

PUT /vehicles/{id} : Mettre à jour les informations d'un véhicule

DELETE /vehicles/{id} : Supprimer un véhicule