# Rapport - Lab05

# Labo 05 : Rapport


- **Auteurs :** Khelfi Amine, Fernandez Samuel

- **Date :** 09.12.2025

## 1. Introduction et Analyse du Problème

### 1.1 Contexte

Le laboratoire consiste à modéliser un système de vélos en libre-service. Les utilisateurs prennent des vélos à des stations, se déplacent entre sites et déposent les vélos. Une équipe de maintenance répartit les vélos entre les stations pour éviter surcharges et pénuries.

L'enjeu principal réside dans la gestion concurrente de ressources limitées : chaque station a un nombre fixe de bornes, les vélos sont de différents types (VTT, Route, Gravel) et les utilisateurs ont tous un type de vélo favori.

### 1.2 Problématiques de concurrence

Le système présente plusieurs défis de synchronisation :

- **Accès concurrent aux stations** : Plusieurs personnes peuvent vouloir prendre ou déposer des vélos simultanément sur la même station.
- **Ressources limitées** : Nombre de bornes limité par station, vélos de types spécifiques potentiellement indisponibles.
- **Ordre FIFO** : Les habitants doivent être servis selon leur ordre d'arrivée lors de conflits.
- **Interaction Van-Personnes** : Le van et les personnes accèdent aux mêmes stations sans interférence mutuelle.

## 2. Choix d'Architecture et de Conception

### 2.1 Moniteur de mesa avec bikestation

La synchronisation est centralisée dans la classe `BikeStation` qui implémente un moniteur de Mesa.

Chaque station gère :

- Un stockage par type (Un tableau de deques pour chaque type de vélo)
- Un mutex pour l'exclusion mutuelle sur les opérations critiques
- Des variables de conditions :
  - `bikes_of_type_available[type]` : Une par type de vélo, pour signaler la disponibilité
  - `slots_available` : Pour signaler la libération de places

**Principe du moniteur de Mesa** : Les threads attendent sur des variables de condition et sont réveillés par `notifyOne()` ou `notifyAll()`. À leur réveil, ils doivent re-vérifier la condition.

## 2.2 Personnes

### 2.2.1 Mécanisme de synchronisation

**`getBike(type)`** : Attend sur `bikes_of_type_available[type]` tant que le type demandé n'est pas disponible. Utilise une boucle `while` pour re-vérifier après réveil.

**`putBike(bike)`** : Attend sur `slots_available` si la station est pleine. Une fois le vélo déposé, signale `bikes_of_type_available[type].notifyOne()` pour réveiller un utilisateur potentiel.

L'ordre FIFO est garanti par les variables de conditions qui maintiennent une file d'attente.

### 2.2.2 Arrêt de la simulation

Lorsque `getBike()` retourne un pointeur nul, le thread sort de sa boucle et se termine.

## 2.3 Van

### 2.3.1 Mécanisme d'équilibrage par site

Si Vi > B-2 : Retire l'excédent de vélos

Si Vi < B-2 : Dépose des vélos en priorité pour les types manquants et complète avec n'importe quel type.

Le van utilise `getBikes()` et `addBikes()` qui ne bloquent jamais. Ces méthodes retournent immédiatement avec ce qui est possible de prendre/déposer

### 2.3.2 Arrêt de la simulation

Le van vérifie `stopRequested()` dans sa boucle principale et sort proprement.

## 2.4 Arrêt de la simulation

La fonction `stopSimulation` dans le main orchestre l'arrêt propre du système en deux étapes :

- Il appel `ending()` sur toutes les BikeStations, ce qui réveille tous les threads bloqués
- Enfin on elle demande l'arrêt à tous les threads via `requestStop()` .

Les threads se terminent alors :

- Les personnes sortent de leur boucle quand `getBike()` retourne un pointeur nul
- Le van sort de sa boucle en détectant `stopRequested()`

# 2.5 Limite du système

Une situation d'équilibrage inefficace est possible si toutes les stations atteignent exactement B-2 vélos avec une répartition déséquilibrée des types :

- Station 1 que des VTT
- Station 2 que des Routes
- Tous les utilisateurs attendent un type absent

Le van ne modifie plus rien car $V_i = B-2$ partout et aucun utilisateur ne peut avancer. Le système va donc toujours tendre vers cette situation.

**Déclaration IA :** L'IA a été utilisée pour l'aide à la planification et à la décomposition des tâches et pour la structuration de ce rapport.