# Rapport Projet

## Membres du Groupe :

- BENMAOUCHE Mohamed  
- HADJOU Tayeb

## Présentation :

## Structure et modélisation :

Afin de mieux simplifier la manipulation de notre code, ce dernier est réparti et organisé au sein de deux classes principales qui sont néanmoins fortement liées. Ces classes sont implémentées de la manière suivante :

### Classe Node :

Cette classe permet de créer et de manipuler un nœud, en prenant en paramètres le parent de ce nœud (si besoin) ainsi qu’un booléen leaf qui permet de définir s’il s’agit d’une feuille.

Elle est caractérisée par un ensemble d’attributs avec leurs getters et setters nécessaires à la manipulation du nœud :

- size : Nombre de clés du nœud initialement défini à 0.  
- keys : Tableau permettant de stocker l’ensemble des clés du nœud (initialement vide).  
- leaf : (booléen) Permet de vérifier si le nœud est une feuille.  
- child : Tableau permettant de stocker les fils du nœud (initialement vide).  
- parent : Permet de stocker le nœud parent.

### Classe Btree :

Cette classe permet de définir et initialiser un arbre-B. Elle est définie par les attributs suivants :

- root : Racine de l’arbre.  
- U : Degré maximum de l’arbre (nombre de clés maximum par nœud).  
- treekeys : Tableau répertoriant l’ensemble des clés dans l’arbre.  
- allNode : Tableau contenant l’ensemble des nœuds.

Elle inclut des fonctions pour manipuler l’arbre, comme linear-tree pour un parcours infixe et d’autres fonctions que nous détaillerons ci-après.

## 1. Recherche :

Pour rechercher une valeur dans l’arbre, le processus est le suivant :

1. Appeler la fonction search avec un nœud (la racine dans ce cas) et la valeur recherchée. Cette fonction appelle searchLeaf qui retourne le nœud contenant potentiellement la valeur recherchée. La fonction compare ensuite la valeur recherchée aux clés du nœud et retourne un booléen (True si trouvée, False sinon).

2. searchLeaf appelle searchInNode, qui renvoie un nœud. Si ce nœud correspond à celui contenant la valeur, il est retourné ; sinon, searchLeaf s’exécute de manière récursive sur les fils jusqu’à satisfaction.

3. searchInNode effectue des comparaisons rigoureuses pour renvoyer le bon nœud. Son algorithme suit ces étapes :

- Si le nœud est une feuille, il est retourné.  
- Incrémenter un indice i jusqu’à ce que :  
 - i dépasse le nombre de clés du nœud.  
 - La i-ème clé dépasse la valeur recherchée.  
- Comparer les clés :  
 - Si la valeur recherchée correspond à une clé, retourner le nœud.  
 - Sinon, retourner le fils correspondant.

## 2. Insertion :

Pour insérer une valeur dans l’arbre, voici les étapes :

1. Identifier la feuille où la valeur sera insérée. Si la feuille dépasse la limite U-1, vérifier si c’est la racine. En cas de dépassement :

- Appeler insertInRoot si le nœud est une racine.  
- Sinon, appeler insertNotRoot.

### InsertInRoot :

Cette fonction insère une clé dans une racine ne respectant pas la limite maximale. Le processus :

- Ajouter temporairement la valeur au nœud pour calculer la médiane.  
- Créer deux nœuds (gauche et droit) avec les clés inférieures et supérieures à la médiane.  
- Définir une nouvelle racine contenant uniquement la médiane.

### InsertNotRoot :

Cette fonction insère une clé dans un nœud non racine ayant atteint la limite maximale :

- Ajouter temporairement la valeur, calculer la médiane et créer un nouveau nœud droit.  
- Mettre à jour les fils et rappeler l’insertion sur le parent jusqu’à stabilisation.