Jessica GOURDON
Julien CHOUKROUN
MAM3 groupe 1

Université Nice Sophia Antipolis | POLYTECH NICE-SOPHIA
Membre de UNIVERSITÉ CÔTE D'AZUR

Rapport de projet : Gestion d'évènements dans une ville

CANNES CÔTE d'AZUR FRANCE

Le 27 Avril 2020

Dans le cadre de ce projet, nous avons réalisé un outil permettant de gérer les évènements d'une ville. Nous avons choisi la ville de Cannes pour cela. Mondialement connue, cette ville accueille chaque année un grand nombre d'évènements dans différents domaines.

Nous avons donc créé cinq types d'évènements :

- Evènements Musicaux
  Ce type contient 3 évènements :
  - Les plages électroniques 2020
  - NRJ Music Awards 2020
  - Soirée Anniversaire de Bach

  Ce type d'évènement contient les attributs suivants : le type (Musical), le titre, la référence, la date de début et de fin (tous deux sous la forme d'un entier entre 1 et 365), le lieu où il se déroule, un booléen indiquant si l'évènement est payant ou pas, le prix si celui-ci est payant (sous la forme d'un entier) et enfin un booléen indiquant si les personnes assistant à cet évènement auront des places attitrées ou non.

- Evènements Cinématographiques
  Ce type contient 2 évènements :
  - Le Festival International du Film
  - CannesSeries

  Ce type d'évènement contient les attributs suivants : le type (Cinématographique), le titre, la référence, la date de début et de fin (tous deux sous la forme d'un entier entre 1 et 365), le lieu où il se déroule, un booléen indiquant si l'évènement est accessible à tous ou s'il nécessite d'avoir une invitation et enfin un booléen indiquant si les personnes assistant à cet évènement auront des places attitrées ou non.

- Evènements Culturels
  Ce type contient 2 évènements :
  - Exposition - La lune et les satellites des planètes
  - Exposition Peinture - L'oeuvre du peintre Korbas

  Ce type d'évènement contient les attributs suivants : le type (Culturel), le titre, la référence, la date de début et de fin (tous deux sous la forme d'un entier entre 1 et 365), le lieu où il se déroule, un booléen indiquant si l'évènement est payant ou pas, le prix si celui-ci est payant (sous la forme d'un entier) et enfin un booléen indiquant si les personnes assistant à cet évènement auront des places attitrées ou non.

- Evènements Gastronomiques
  Ce type contient 2 évènements :
  - Salon du Vin et de la Gastronomie
  - Cuisine Cannoise en Fête

  Ce type d'évènement contient les attributs suivants : le type (Gastronomique), le titre, la référence, la date de début et de fin (tous deux sous la forme d'un entier entre 1 et 365), le lieu où il se déroule, un booléen indiquant si l'évènement est payant ou pas et le prix si celui-ci est payant (sous la forme d'un entier).

- Evènements Business
  Ce type contient 3 évènements :
  - It & It Security Meeting
  - MAPIC : Marché International de l'implantation commerciale et de la distribution
  - Finance and RH Meeting

  Ce type d'évènement contient les attributs suivants : le type (Business), le titre, la référence, la date de début et de fin (tous deux sous la forme d'un entier entre 1 et 365), et le lieu où il se déroule.

Pour la création de tous ces évènements, nous avons une classe abstraite Evenement possédant les attributs qui seront communs à tous les types d'évènements : le titre, le type, la référence, la date de début, la date de fin et enfin le lieu où il se déroule. Les classes filles EvenementBusiness, EvenementCinematographique, EvenementCulturel, EvenementMusical et EvenementGastronomique possèderont des attributs spécifiques. La classe Evenement possèdera les fonctions getReference() ainsi que getType() en abstrait. En effet, le type d'un évènement sera défini lors de la création des classes filles et la référence, contenant le type, également. La référence définie par la fonction setReference() est composée des lettres majuscules composant le titre de l'évènement, puis des dates de début et de fin (sous forme d'un entier entre 1 et 365) et enfin des deux premières lettres du type d'événement. Tous ses éléments seront séparés par un tiret. Par exemple, LPE-221-223-Mu correspond à : Les Plages Electroniques ayant comme date de début 221 et comme date de fin 223 et étant un évènement Musical. Chaque référence est unique.

- La recherche par date

La recherche par date repose sur deux classes : Intervalle et ABRIntervalle. La classe Intervalle, très simple sert à générer un objet de type Intervalle à partir de deux dates sous forme d'entier. La classe ABRIntervalle va créer des arbres d'intervalles de type ABRIntervalle. L'ajout de chaque élément se fait de la manière suivante : on prend un évènement en entrée et on récupère sa date de début et sa date de fin. On crée ensuite un objet Intervalle à partir de ces deux entiers. Les éléments de l'arbre seront triés à partir de l'élément de début de l'intervalle. Par exemple, si l'entier de gauche de l'intervalle est inférieur à celui de la racine et que l'arbre de recherche ne possède pas de sous arbre gauche, alors on ajoute notre intervalle à gauche de la racine, c'est-à-dire que l'on crée un sous arbre gauche ayant comme racine notre intervalle mais ne possédant pas de sous arbre. Dans le cas où l'entier de gauche de l'intervalle est égal à l'entier de gauche de la racine, nous avons décidé de l'ajouter à gauche de la racine. Notre classe ABRIntervalle possède un attribut tableau de type HashMap, dans lequel on stocke chaque élément de l'arbre. La clé correspond à l'intervalle et la valeur à l'évènement de celui-ci.
Pour effectuer la recherche par date, on se sert de la fonction rechercheValeurIntervalle() qui renvoie true si l'intervalle qu'on donne en entrer intersecte un autre intervalle. On parcourt donc les éléments de l'arbre en faisant les vérifications avec cette fonction. Lorsque l'on trouve un intervalle correspondant, on récupère l'évènement associé grâce au HashMap tableau. On stocke chaque évènement correspondant dans une ArrayList liste. Il ne restera donc plus qu'à afficher en sortie les éléments de cette ArrayList.

- La recherche par référence

Pour effectuer la recherche par référence, on se sert de la classe ArbreReference. On va donc créer un arbre de référence. L'ajout de chaque élément se fait de la manière suivante : On prend un évènement en entrée et on récupère sa référence (type String et de la forme : LPE-221-223-Mu). Les éléments de l'arbre (les chaines de caractères correspondant aux références) seront donc triés en fonction de l'ordre alphabétique du 1^{e} caractère). Par exemple, si le 1^{e} caractère de l'élément est alphabétiquement inférieur à la racine et que l'arbre de recherche ne possède pas de sous arbre gauche, alors on crée un sous arbre gauche ayant comme racine notre référence mais ne possédant pas de sous arbre. Notre classe ArbreReference possède un attribut tableau de type HashMap, dans lequel on stocke chaque élément de l'arbre. La clé correspond à la référence et la valeur à l'évènement de celui-ci.
Pour effectuer la recherche par référence, on parcourt les éléments de l'arbre en vérifiant s'ils correspondent à la référence que l'on cherche. Lorsque l'on trouve un élément correspondant, on récupère l'évènement associé grâce au HashMap tableau. Comme chaque référence est unique, une seule peut correspondre. Nous n'avons donc pas besoin d'utiliser une ArrayList ici. Il nous suffira seulement d'afficher l'évènement correspondant.

- La recherche par type

Pour effectuer la recherche par type, nous avons choisi de travailler sur des arbres de référence, de type ArbreReference. En effet, dans la référence d'un évènement on retrouve le type de celui-ci (les 2 premières lettres du type). La recherche par type se situe donc dans la classe ArbreReference. Pour effectuer cette recherche, on extrait les deux premières lettres du type que l'on recherche, en mettant la première en majuscule, et on les stocke dans une variable. C'est à partir de celle-ci que l'on va réaliser la recherche. On parcourt les éléments en comparant cette chaine à la chaine composée des deux derniers caractères de l'élément qu'on veut parcourir. En effet, les deux derniers éléments de la référence correspondent aux deux premières lettres du type de l'évènement. Lorsque l'on trouve un élément correspondant, on récupère l'évènement associé grâce au HashMap tableau. On stocke chaque évènement correspondant dans une ArrayList liste. Il ne restera donc plus qu'à afficher en sortie les éléments de cette ArrayList.

- La recherche combinée

La recherche combinée consiste à chercher des évènements en filtrant en fonction à la fois du type mais également de la date. La recherche combinée se servira donc du travail que l'on a réalisé pour la recherche par type ainsi que pour la recherche par date. On travaillera d'abord sur un arbre de référence. La recherche combinée se situe donc dans la classe ArbreReference. Deux arguments seront nécessaires : le type et la date. On commence par effectuer la recherche par type sur l'arbre de référence. L'ArrayList liste contient les éléments trouvés par cette recherche. Si celle-ci est vide, on affiche directement qu'aucun évènement n'a été trouvé. Dans le cas contraire, on crée un arbre de type ABRIntervalle et on y stocke les évènements contenus dans l'ArrayList liste. Enfin, on effectue la recherche par date sur cet arbre.

- Options

Nous avons utilisé la classe Date de java pour présenter nos intervalles de façon lisible. Nous avons alors dû faire le lien entre une Date et un jour dans l'année. Pour ce faire, nous avons crée deux méthodes dans la classe Evenement. La méthode nombreEnDate convertit un nombre entier en date au format JJ/MM/AAAA et la méthode dateEnNombre qui convertit une date au format JJ/MM/AAAA en nombre entier.

- Répartition des tâches

Pour la réalisation de ce projet, nous nous sommes réparti les tâches. Jessica s'est donc chargée de la création des évènements, ainsi que de la recherche par date, par type et de la recherche combinée. Julien quant à lui a effectué la recherche par référence, la mise en forme du code, l'organisation de la classe Utilisateur avec les Scanner afin de coordonner toutes les recherches.

- Difficultés

Nous avons d'abord pu rencontrer quelques difficultés pour réaliser la recherche par date. En effet, il nous a fallu se familiariser avec les arbres d'intervalles, ce qui ne fut pas chose aisée au début. Nous avons donc réalisé le code de la manière la plus simple possible pour plus de clarté. Puis, il a également été difficile d'appréhender le fait de manipuler des arbres avec des chaines de caractères et non des entiers. Cela concernait la classe ArbreRecherche. En effet, il a fallu créer un arbre par référence et donc trié des chaines de caractères. Nous les avons donc triés par ordre alphabétique. Nous n'avions jamais fait cela. Nous avons donc appris de nouvelles choses lors de ce projet.

- Notre diagramme