CLASSES

Event

Begin : DateTime

end : DateTime

Title : string

Description : string

Catégorie : Categorie

idDatabase : unique

Catégorie

title : string

color : RGB

Database

Bool createDatabase(string fullPath) // crée physiquement le fichier

Database\* openDatabase(string fullPath) // ouvre la base. Retourne un pointeur vers l’instance.

Bool closeDatabasedatabase\*) // ferme la base.

Class DAL

Iterator getAllEvent() // remplit un container d’Events, local à DAL mais persistant avec la classe DAL.

Retourne un iterateur vers le conteneur. Ainsi les resulttats de la requete sont accessibles jusqu’à la prochaine requete, et le owner des Events est la DAL.

Les clients de l’iterateur doivent exploiter immediatement les objets, et ne pas garder de pointeurs sur ceux-ci, car ils seront deétruits à la prochaine requete.

**Void getAllEvent(container &)** // bien que les objets Event soient instanciés dans cette methode, c’est le client proprietaire du container qui aura la charge de les detruires.

Un set de Events est stocké par l’application dans un container.

**Void getEvents(container &, dateTime debut, dateTime fin).**

**Void insertEvent(newEvent : Event)**

Son Id est vide avant l’insert. Apres l’insert, on recupere cette valeur, et complete l’objet Event.

**bool deleteEvent(Event &)** // Le client est en charge de detruire l’instance de cet Event quand le delete reussit.

**void updateEvent(Event &)**

**bool recreateDatabase()** // Recré la base. Suprresion et Creation du fichier de la base. Requete une grosse string SQL de creation de la base et des tables, et de valeurs initiales(exemples de Categorie, d’Event, …). REQUETE A TENIR UP-TO-DATE !

AFFICHAGE

On affiche tout, ou un ensemble des Events.