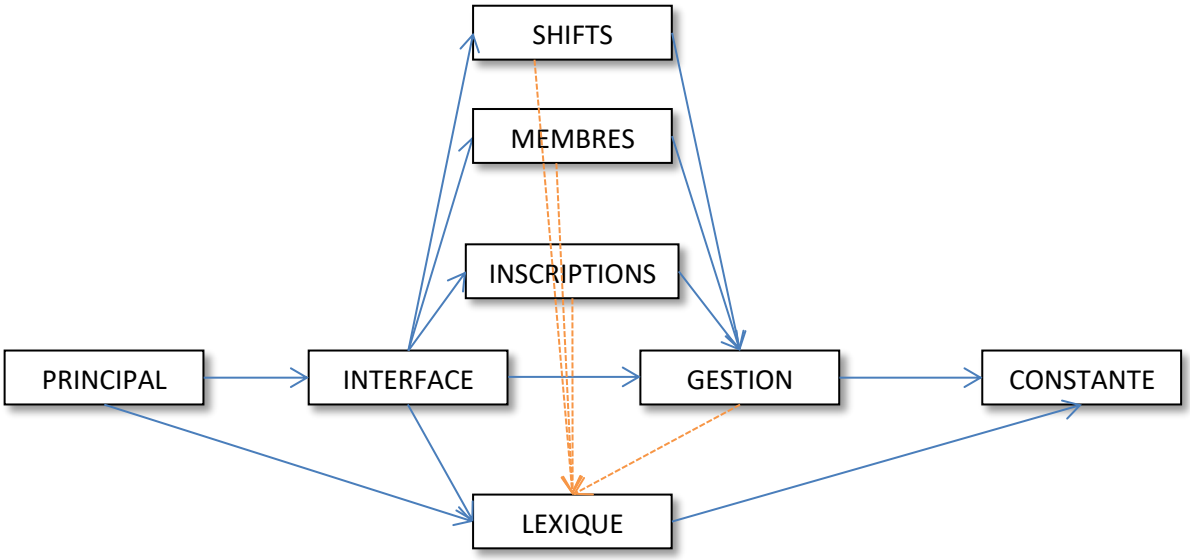
Structure pour le dossier

1. Vue d'ensemble des modules



Note. Le document suppose une découpe en plusieurs fichiers .c accompagnés de plusieurs fichiers .h. Si vous le désirez et si cela vous semble plus simple, dans un premier temps, vous pouvez n'utiliser qu'un seul fichier .h.

2. Remarque sur le PP et l'organisation du code

Portez une attention toute particulière au **principe du point de modification unique**.

Concrètement, cela signifie que votre programme final ne doit pas comporter de code répété. Si un bout de code est utilisé à deux endroits, il faut en faire un module !

Par exemple : *ajouter une nouvelle doublette dans la liste chaînée*. On doit réaliser cette opération non seulement lorsque l'utilisateur inscrit une nouvelle doublette, mais aussi quand, au lancement du programme, on charge les informations à partir du fichier sauvé à la fin de la dernière utilisation du programme (fichier *inscriptions.dat)*. Il devrait donc y avoir un module (au sens de PP) qui s'occupe uniquement de l'ajout et ce module devrait être appelé à la fois par le module visant à ajouter un joueur entré par l'utilisateur et par le module visant à ajouter les informations stockées dans le fichier.

3. Constantes

Placez dans le fichier *constantes.h* toutes les définitions de constantes (constantes symboliques, énumération…) utilisées globalement par tous les modules.

4. Lexique

Ce module contiendra toutes les fonctions liées aux messages ; par exemple :

- une fonction permettant d'afficher le message correspondant à un code donné (cette fonction sera très utilisée par les autres modules) ;
- [éventuellement] une fonction permettant d'afficher un menu et de demander le choix de l'utilisateur.

Note. Dans cette description, on ne cite que les fonctions principales de chaque module, à savoir les fonctions qui seront utilisées par les autres modules. Vous pouvez bien sûr ajouter des fonctions auxiliaires pour vous faciliter le travail ! Cette remarque vaut pour tous les modules !

5. Gestion

Ce module contiendra toutes les fonctions liées aux listes chaînées retenues en mémoire et stockant les shifts et les inscriptions … :

- une fonction pour ajouter un shift dont la date est donnée ;
- une fonction pour ajouter une inscription à un shift ;
- une fonction pour afficher des listes des inscriptions pour un shift ;
- une fonction pour réaliser la suppression d'une inscription.

Note. Dans l'idéal, ces fonctions ne devraient pas communiquer directement avec l'utilisateur (et donc ne devraient pas utiliser le lexique) ; elles devraient se contenter de travailler sur la mécanique des listes chaînées. Concrètement, ce n'est pas toujours facile d'avoir une découpe nette ; certains recoupements sont donc autorisés (------>).

6. Shifts, Membres et Inscriptions

Ces modules comporteront toutes les fonctions liées à l'utilisation des fichiers de données, respectivement, le fichier *datesOrgShifts.dat*, m*embresFSBB.dat* et *inscriptions.dat* ; entre autres la sauvegarde et le chargement des données.

Note. Ces fonctions ne devraient s'occuper que de la lecture et/ou de l'écriture dans les fichiers ; pour tout ce qui est manipulation de liste chaînée ou tableau, elles devraient faire appel aux fonctions du module Gestion.

7. Interface

Ce module contiendra toutes les fonctions relatives au dialogue avec l'utilisateur, comme par exemple les divers menus, les obtentions d'informations (date du shift choisi, numéro de matricule

des joueurs …) et l'affichage « en clair » des messages d'erreur renvoyés par les fonctions plus mécaniques (de Gestion et de Fichiers par exemple).

Grosso modo, les fonctions du module Interface seront des fonctions de haut niveau qui feront appel (1) aux fonctions de Lexique pour afficher les choix qui s'offrent à l'utilisateur ; (2) aux fonctions de Gestion pour tout ce qui concerne la manipulation de la liste chaînée en mémoire ; (3) aux fonctions de Shifts, Membres et Sauvegarde pour la validation des membres, le chargement et la sauvegarde.

8. Principal

Finalement, le module principal ne devrait plus avoir qu'à (1) déclarer les variables utilisées dans tout le programme, (2) faire appel à la fonction correspondant au menu principal.

9. Remarque finale mais super importante

Pensez **clean code** !