**Estrutura de Dados**

**Data Access Object**

Como durante as aulas da disciplina não nos foi ensinado como aceder à nossa base de dados, nós tivemos que realizar alguma pesquisa de forma a sabermos como definir os métodos para este problema.

Primeiro como nos foi sugerido, analisamos o **JDBC** do java (*Java Database Connectivity* ou **JDBC** é um conjunto de classes e interfaces (API) escritas em Java que fazem o envio de instruções SQL para qualquer banco de dados relacional).

Depois de analisar vimos quais seriam os métodos desta API que podíamos utilizar e que nos iriam oferecer uma maior performance e facilidade de acesso à Base de Dados.

Por fim, conciliamos o nosso código com o código **SQL**, para aceder à base de Dados com o objetivo de obter os resultados pretendidos em cada método através de **selects**, **inserts** e **updates**.

**Lists**

Neste projeto utilizamos a estrutura de armazenamento de Dados **Lists** em detrimento das outras estruturas porque

**Interface**

Apesar de no ano anterior na **Unidade Curricular de LI3** termos feito já uma pequena interface, neste projeto queríamos fazer uma interface mais personalizada, com melhor manuseamento por parte do utilizador e também com uma boa performance. Para isso pesquisamos sobre as utilidades do **java swing** com fim de utilizar as suas funcionalidades para realizar uma interface com as características acima referidas.

**Ícones** Como é mais fácil para um Utilizador associar um ícone a uma função do projeto, por exemplo a função pesquisar é associada a uma lupa, tentamos colocar ícones sugestivos em toda a interface. Isto vem oferecer um melhor manuseamento e uma interface mais limpa e divertida. Alguns dos ícones foram mesmo desenhados por nós de forma a representarem melhor aquilo que nós queremos transmitir.

**Janelas** Neste projeto tentamos utilizar o mais possível uma janela de forma a o utilizador não se confundir tanto. Para isso usamos um layout disponibilizado pelo java chamado **CardLayout**, este layout permite alterar os **JPanels** de acordo com o que nós queremos, não criando novas janelas pois estas iriam tornar a interface muito confusa porque teria um grande numero de janelas o que poderia induzir o próprio Utilizador a erro. Depois de alguma pesquisa e alguma prática conseguimos trabalhar com este layout de forma a deixar a interface da forma pretendida, ou seja mais simples e menos confusa.

A camada de interface, Presentation Layer, permite isolar a

interface com o utilizador por forma a que o resto da aplicação

esta não esteja dependente de uma interface concreta.

• A camada de negócio, Business Logic, implementa a lógica da

aplicação.

• A camada de dados, Data Layer, permite isolar o acesso aos

dados, por forma a que o resto da aplicação não esteja

dependente da origem ou estrutura sob a qual os dados estão

armazenados.