

Projeto de AEDA

Praias Fluviais

**Projeto de AEDA2018 -- Engenharia Informática e Computação**

**Regente: Ana Paula Cunha da Rocha**

**Turma 4**

**Grupo 1**

* Tomás Nuno Fernandes Novo --- up201604503 --- up201604503@fe.up.pt
* Mariana Catarina Pereira Soares --- up201605775 --- up201605775@fe.up.pt
* João Pedro Viveiros Franco --- up201604828 --- [up201604828@fe.up.pt](mailto:up201604828@fe.up.pt)

**Supervisor:** Liliana da Silva Ferreira

Novembro, 2017

Índice

[Introdução 3](#_Toc498449254)

[Desenvolvimento do projeto 4](#_Toc498449255)

[Casos de utilização 6](#_Toc498449257)

[Obstáculos 7](#_Toc498449258)

[Dedicação 8](#_Toc498449258)

[Conclusão 9](#_Toc498449258)

Introdução

No âmbito da unidade curricular Algoritmos e Estrutura de Dados, foi-nos proposta como 1ª parte do trabalho prático, a realização de um projeto cujo tema é as Praias Fluviais portuguesas.

No nosso belo país encontramos variadas praias fluviais, todas elas com idiossincrasias diferentes que as tornam únicas. De um modo geral, obviamente que possuem caraterísticas comuns. No nosso projeto, podemos distingui-las não só pelo seu nome como também pelas suas coordenadas GPS (latitude e longitude). Cada uma pertence ao seu respetivo concelho e pode (ou não) ter bandeira azul, bem como serviços de café e restauração, desporto e presença de nadador-salvador. Também podem ter serviços fora da praia, como alojamento e pontos de interesse turístico.

Apesar de todas as caraterísticas comuns, como dito anteriormente, as praias diferem umas das outras, podendo elas ser praias fluviais de rio ou de albufeira. Ao longo do nosso projeto, relativamente às primeiras abordamos o seu caudal, largura e profundidade. Quanto às segundas estudamos apenas a sua área.

Desenvolvimento do Projeto

Com o trabalho desenvolvido na primeira parte do projeto, a solução na sua 2ª parte tornou-se muito mais simples de implementar, visto que já possuíamos as classes feitas e só foi necessário realizar algumas alterações e criar as estruturas de dados pedidas. A alteração mais relevante da 1ª para a 2ª parte foi a mudança do tipo da classe alusiva aos serviços fora da praia para uma classe de apontadores.

Começámos então por guardar as praias fluviais numa árvore binária de pesquisa que se encontra na classe de gestão *GestorPraias*. Fomos forçados a transformar as funções *loadPraias* e *savePraias* do ficheiro *praias.cpp* visto que mudámos de estrutura de dados. Como era pretendido que as praias estivessem ordenadas por concelho e dentro de cada concelho por classificação com bandeira azul, foi necessária a implementação do operador < que se encontra na classe *praiaFluvial*. Criámos uma *struct* designada de *comparePraia* que chama o operador e compara as praias dentro da BST. A utilização de iteradores facilitou imenso a implementação desta estrutura de armazenamento de dados, não só na pesquisa devido à sua função-membro *find,* como também na listagem das praias pelos critérios anteriormente referidos, bem como a remoção ou inserção de uma praia na árvore e a atribuição ou extração da bandeira azul a uma praia fluvial.

Após a conclusão da parte relativa à BST, criámos uma nova classe de nome *data* com o fim de saber o dia, mês e ano em que sucedeu um determinado evento.

Procedemos à criação de um vetor de *priority queues* na classe gestora em que cada elemento é uma *priority queue* de um único tipo de serviço, podendo ser então café, restaurante, nadador salvador ou campo desportivo. Visto que era pretendido a ordenação destas filas de prioridade pela data da última inspeção, criámos um operador < desta vez na classe *servicos* e uma *struct* que nomeámos de *compareServico* que compara os serviços dentro da fila de prioridade. As *priority queues* criadas estão ordenadas de maneira a que o topo seja um serviço com a data de inspeção menos recente. Nestas é possível registar a realização de um serviço de inspeção bem como a atualização da data do respetivo serviço.

Quanto aos serviços, como estes podem ser sujeitos a períodos de inatividade temporários ou definitivos, guardámo-los numa *hashtable* com nome *servicosnaoativos*. Criámos a *struct* *servicohash* que possui os operadores que definem a posição do serviço na tabela de dispersão, bem como o operador de igualdade entre dois elementos da tabela. Adicionámos também uma variável *bool* de nome *aberto* na classe *servicos* para verificar se o serviço está ativo ou não. Na classe *servicoForaDaPraia* criámos outra variável *bool* de nome *permanente* que informa se o serviço está inativo temporariamente ou permanentemente.

No menu de interface é possível realizar todas as operações que nos foram propostas no enunciado da 2ª parte do projeto, estando este preparado não só para *inputs* incorretos por parte do utilizador como também para lhe permitir um fácil acesso ao objetivo que o *user* pretende.

Casos de Utilização

Com a nossa aplicação, o utilizador pode:

* Adicionar ou remover uma praia fluvial numa BST.
* Pesquisar uma praia na BST.
* Editar a bandeira azul de uma praia.
* Registar a inspeção de uma praia fluvial.
* Abrir/Fechar um serviço de uma praia.
* Listar todas as praias fluviais.
* Listar todas as praias fluviais por concelho.
* Pesquisar uma praia.
* Pesquisar a praia fluvial mais perto.
* Pesquisar os serviços perto de uma praia fluvial.

Obstáculos

Ao longo da realização deste projeto, deparámo-nos com certas dificuldades que retardaram o seu término, apesar de todas elas terem sido superadas com sucesso.

Aprofundando um pouco, os nossos maiores entraves residiram na elaboração do menu de interface e da aplicação do polimorfismo do operador << no *cout* das classes *servico* e *praiaFluvial*.

Utilizámos o *GitHub* para podermos trabalhar de forma sincronizada e em conjunto conseguimos distribuir tarefas, repartindo assim a carga de trabalho por todos e aproveitando o tempo que dispúnhamos.

Mesmo havendo alguma complexidade na elaboração do projeto, aplicámo-nos com concentração e determinação na sua realização, o que facilitou o seu entendimento e tornou mais acessível a sua conclusão.

Dedicação

A elaboração deste projeto foi realizada com empenho e dedicação elevada por parte dos estudantes. A distribuição de tarefas em função dos conhecimentos de programação que cada aluno possui foi essencial para ultrapassar as dificuldades.

Podemos concluir assim que todos aprendemos com a realização deste projeto, todos os membros se empenharam de forma semelhante naquilo que sabiam.

Conclusão

Com este projeto foi então possível aumentar os nossos conhecimentos sobre programação em C++, bem como melhorar o nosso código, visto que demos uso ao que aprendemos tanto nas aulas teóricas como nas práticas.

Concluímos assim que este trabalho foi benéfico para todos os elementos pois melhorámos a nossa capacidade programadora aprendendo trabalhar com árvores binárias de pesquisa, filas de prioridade e tabelas de dispersão.