Universidade Federal do Rio Grande do Sul

Aerosafe

Jéferson Ferreira Guimarães

Vinícius Chagas Soares

Yuri Escalianti

Professor: Leandro Krug Wives

Porto Alegre, 2016

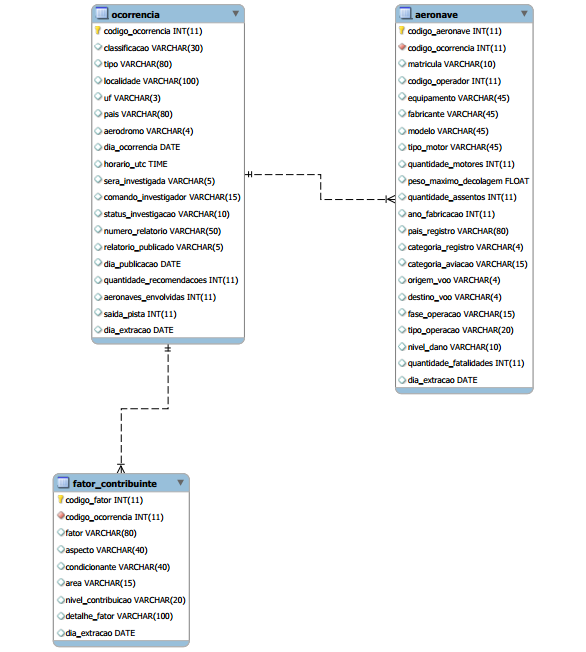
**VISÃO GERAL**

O aplicativo *“Ocorrências Aeronáuticas”*, através dos esforços conjuntos de seus criadores, veio à tona propondo ser uma maneira nova – e mais intuitiva – de ter-se acesso às informações oficiais da Aeronáutica e do Governo, a respeito dos diversos acidentes e incidentes ocorridos dentro do perímetro aéreo brasileiro.

Ao vasculhar-se o vasto acervo de dados oficiais, disponibilizados pelo Governo Federal de forma pública, percebemos que o conjunto de dados referentes a incidentes envolvendo aeronaves, especificamente, representaria uma excelente oportunidade de se desenvolver um novo aplicativo, com a finalidade de melhor representa-los. Da forma como são divulgados ao público, atualmente, os dados são extensos e difíceis de analisar a olho nu; simplesmente carrega-los em uma tabela, em algum programa do tipo, também não nos parecia ser a melhor solução.

Diante destes fatores, propusemo-nos e dispusemo-nos a desenvolver, da estaca zero, um programa utilizando a linguagem **C#** que implementasse uma interface intuitiva e visual, para melhor visualizar e estudar os dados em questão. Utilizando dados do Google Maps™, conseguimos mostrar ao usuário o local onde fora registrada a ocorrência selecionada – a qual, logicamente, também aparece em detalhes (e de forma clara e organizada), tornando seu entendimento muito mais fácil.

A respeito do conjunto de dados selecionado, acompanha-o uma documentação oficial, explicitando os tipos de dados de cada campo lido, e como as 3 diferentes classes se relacionam. O diagrama que representa os dados e suas relações é o que está a seguir, na próxima página:



*Diagrama de Entidades e Relações do conjunto de dados escolhidos.*

Esta especificação original do conjunto de dados foi, naturalmente, utilizada como base para as classes, tipos, métodos e campos que implementamos no programa, com a seguinte ressalva: além de adaptar os tipos sugeridos para os tipos utilizados pelo C# (i.e. “string” ao invés de “VARCHAR”), foi necessário prestar atenção ao que estava escrito nos arquivos em si, dado que *em diversas situações os campos* ***não*** *possuíam dados condizentes com os tipos especificados.* Sendo assim, fez-se necessária a implementação de alguns testes básicos de consistência no programa.

Foi escolhida e utilizada a linguagem **C#** - como dito anteriormente – para o desenvolvimento do projeto. Tomamos esta decisão levando em conta diversos fatores cruciais, tais como tempo hábil de desenvolvimento, facilidade e praticidade de uso, e flexibilidade. A linguagem (e, em especial, a IDE Visual Studio) mostra-se uma escolha extremamente inteligente para se desenvolver um programa, em tempo relativamente curto, que ainda assim tenha um bom nível de funcionalidade, estabilidade, uso moderado de recursos e, principalmente, uma interface gráfica fácil de ser usada e desenvolvida.

**IMPLEMENTAÇÃO**

Para implementação do projeto planejado, foram criadas classes para cada arquivo de dados da base escolhida, sendo essas as classes: “**Aeronave**”, “**FatorContribuinte**” e “**Ocorrencia**”. Essas classes contêm somente os dados lidos do arquivo, e são unidas em uma nova classe, “**DadosOcorrencia**”, através do campo “**código\_ocorrencia**”, comum a todas as classes. O diagrama ER abaixo mostra a relação dos dados utilizados:



Os arquivos de dados foram obtidos em formato CSV através do Portal Brasileiro de Dados Abertos (2016), disponibilizados pelo CENIPA. Para a leitura dos dados foi utilizada a biblioteca **CsvHelper** (2016), que provê uma API para acesso aos arquivos, permitindo que todos os dados do arquivo selecionado sejam lidos de forma rápida e prática, em alguns casos requerindo apenas e tão somente uma única linha de código.

Depois de lidos, os dados são adicionados a um dicionário, que utiliza o campo “**código\_ocorrencia**” como chave e o objeto “**DadosOcorrencia**”, contendo todos os dados em questão, como valor. O dicionário foi escolhido por permitir listar todos os dados separando-os pelo código de ocorrência, que é único para cada ocorrência lida e faz-se presente em todos os arquivos, facilitando o agrupamento das entradas após a leitura do arquivo.

Os dados lidos são armazenados em uma árvore B+ diretamente no disco rígido, utilizando a biblioteca **BPlusTree** (2016), que provê a implementação da estrutura. Esta biblioteca permite, dentre outras coisas, a definição de opções para ajuste de tamanho em disco e política de criação de um novo arquivo para a árvore; desta forma, torna-se extremamente mais prática e rápida a implementação funcional deste tipo de estrutura de dados, que normalmente, implementando-se de forma manual, seria uma tarefa árdua e trabalhosa.

Utilizando o componente foi possível alcançar o funcionamento desejado: leitura da árvore a partir do disco rígido para todas as operações realizadas no programa, utilizando um código testado e otimizado, permitindo uma boa performance no aplicativo. Entretanto, foi necessário implementar um protocolo de serialização dos dados a serem armazenados no disco, bem como uma classe responsável pela serialização e “desserialização” dos dados.

Para o protocolo foi utilizada a biblioteca **Protobuf-Net** (2016), que permite a serialização e desserialização de dados ao adicionar códigos de contrato às classes. A classe de dados criada para efetivamente conduzir todos estes processos referentes à serialização dos dados é a classe “**ProtoNetSerializer**”, obtida na própria documentação da biblioteca da árvore B+.

Por fim, optou-se pela utilização de uma árvore B+ para a representação dos dados lidos no disco devido à sua organização diferenciada: dado que todos os nodos-folha ocupam o mesmo nível, e que esta árvore diferencia-se por trata-los como uma lista duplamente encadeada, isso nos proporciona ganhos satisfatórios de performance ao efetuar-se operações de busca de dados, uma vez que requer menos operações de acesso ao disco.

**GUIA DE USO**

Presente no arquivo “Guia de Usuário.pdf” incluído no projeto.

**CONTRIBUIÇÃO**

Acreditamos que, primeira- e primariamente, a grande contribuição deste trabalho seja direcionada a todos que, seja pelo motivo que for, se interessam por ou precisam trabalhar com os dados disponibilizados pelo CENIPA, a respeito de cada incidente aéreo ocorrido dentro do perímetro aéreo brasileiro (o que, inclusive, em ao menos uma ocasião, incluiu o território marítimo sobre o qual o país mantém soberania), dado que tornou-se possível visualizá-los, analisa-los e estuda-los de forma muito mais prática, intuitiva e visual.

Dito isso, durante todo o período de concepção e desenvolvimento do projeto, não tivemos conhecimento de nenhum outro programa ou aplicativo – seja nativo, web, mobile ou desktop – de funcionalidade similar ao que nos propusemos a criar.

Por outro lado (e como já dito anteriormente), observando os aspectos internos do programa, é possível perceber que fizemos contribuições especialmente ao conjunto de dados disponibilizado publicamente pelo CENIPA: encontramos inconsistências entre o diagrama oficial e o que realmente encontramos nos arquivos .csv, tais como dados não-condizentes com o tipo de dados esperados e, em uma ocasião, um campo que constava na especificação e que nunca aparecia no .csv onde deveria constar.

Também contribuímos com os projetos *third-party* que utilizamos, a fim de simplificar o desenvolvimento do programa, tais como os projetos **CsvHelper** e **BPlusTree** – respectivamente utilizados para leitura dos arquivos .csv, e leitura e escrita de árvores B/B+ no disco com os dados utilizados. Foram feitas adaptações a fim de abstrair alguns métodos e funções destas bibliotecas, diminuindo-se, assim, a complexidade do nosso código final.

Ademais, também foram utilizados algoritmos de ordenação encontrados na internet, adaptando-se os tipos de dados esperados para os com os quais trabalhávamos, por vezes requerendo mudanças mais aprofundadas nos próprios algoritmos.

**CONSIDERAÇÕES FINAIS**

Analisando os registros de ocorrências aeronáuticas vemos que muita informação útil pode ser retirada desses dados. Há cidades com muitas ocorrências, e há ocorrências bem interessantes, que nem sempre são citadas em notícias de jornais e televisão.

Este projeto nos permitiu ver que utilizando as técnicas de classificação e pesquisa de dados adequadas, podemos utilizar os dados de ocorrências de forma eficiente, e gerar estatísticas úteis a partir disso.

De igual sorte, o trabalho neste projeto nos permitiu ter um contato mais profundo e requintado com um projeto de aplicação mais próxima ao mundo real, dado que fomos solicitados a trabalhar diretamente com dados extraídos de entidades do Governo Federal.

Ademais, o uso de ferramentas de mais alto nível - tais como a IDE Visual Studio - nos possibilitou adquirir maior familiaridade com a filosofia de desenvolvimento rápido de aplicações dotadas de interface visual, utilizando-se ferramentas de RAD (Rapid Application Development), dos quais a IDE em questão faz parte. No outro extremo, sob a ótica de mais baixo nível dos algoritmos em si, pudemos implementar, testar e observar o funcionamento dos algoritmos vistos em aula - tanto de ordenação quanto de armazenamento, busca e leitura de dados - em campos e estruturas obtidos de fontes alheias às do professor, comprovando-se, assim, sua correta funcionalidade.

**REFERÊNCIAS**

PORTAL BRASILEIRO DE DADOS ABERTOS. ***Ocorrências Aeronáuticas na Aviação Civil Brasileira.*** Disponível em: <http://dados.gov.br/dataset/ocorrencias-aeronauticas-da-aviacao-civil-brasileira>. Acessado em 01 set. 2016.

CSV HELPER. Disponível em: < https://joshclose.github.io/CsvHelper/>. Acessado em 18 nov. 2016.

CSHARPTEST.NET. ***BPlusTree.*** Disponível em: <http://csharptest.net/projects/bplustree/index.html>. Acessado em 18 nov. 2016.

PROTOBUF-NET.Disponível em: < https://github.com/mgravell/protobuf-net>. Acessado em 18 nov. 2016.