Documentação - Organização de Código

João Vitor Antoniassi Segantin

March 12, 2018

Abstract

Este documento visa explicar a organização de códigos fonte, e oferecer um breve resumo sobre a importância de cada arquivo de código-fonte.

Ao final da leitura desse documento, você será capaz de decidir posições novas de códigos-fonte oriundos de atualizações, implementações de funcionalidades, e/ou re-organizações possíveis. Seja bem-vindo(a) ao projeto Visu-RA!

1 Da organização

A organização dos códigos é dada por sua utilidade, seu efeito no projeto.

Cada arquivo foi separado intencionalmente, a fim de facilitar possíveis extensões, e mudanças de arquiteturas, visando um baixo custo e alto desempenho, a manutenção da organização é de vital importância a todo o projeto.

A interface do banco de dados, em sua maioria está implementada em linguagem C de programação, utilizando definições compatíveis com o gcc (GNU C Compiler), com exceção dos arquivos contidos no diretório "scripts/", que foram implementados em linguagens de programação alternativas, visando maior velocidade de desenvolvimento do projeto.

Faz-se necessário entender que, os códigos de banco de dados, e os códigos de lógica, estão separados intencionalmente, a fim de obter futura integração com um sistema de Big-Data, independende de Banco de Dados.

O projeto da interface completo, está situado num repositório privado em bitbucket, caso queira ter acesso a atualizações e coisas do tipo, peça acesso ao seu superior, informando o motido do acesso, seus dados pessoais e quando, precisará desse acesso.

Como todo repositório git, esse projeto está subversionado, com o intuito de evitar problemas com más implementações. Antes de solicitar um "merge" ou dar "commit", assegure-se de que pelo ou menos duas pessoas tenham visto e autorizado seu commit.

Lembre-se, está mechendo com milhões de dados, e operações importantes.

2 Arquivos de Bancos de Dados

Os arquivos com relação a banco de dados são divididos, em duas partes básicas, os **arquivos SQL**, que servem para a criação do banco de dados em si, alterações diretas em banco e arquivos de inserção de dados fictíceos para testes, e os **arquivos de Código-Fonte**, arquivos headers, e bibliotecas do projeto para manipulação do banco de dados.

Os arquivos de código-fonte, do banco de dados, foram separados do projeto inicial, para facilitar a implementação de um futuro Big-Data.

2.1 Arquivos SQL

Os arquivos SQL estão localizados no diretório **Banco/**, são eles:

- Banco-final.sql

  Utilizado para a geração do banco de dados MySQL.

- Banco-raw.sql

  Arquivo oriundo do "POST-FORWARD script" da ferramente "MySQL Workbench", pode também criar o banco, mas, este ficará com nome "mydb", que não é o padrão reconhecido pela interface. Sua utilização para criação de um banco de dados é recomendada, em caso de re-implementação, ou alteração da estrutura do banco, utilizando a ferramente MySQL workbench. Esse deve ser alterado e salvo como Banco-final.sql, substituindo as ocorrências de "mydb" por "teste"

- produzir-dados.sql

  Script para criação dos dados essenciais para a utilização do Banco de dados pela interface. Cria somente os dados necessários.

- produzir-dados-testes.sql

  Script para criação de dados de testes. Para seu funcionamento ideal, é necessário que "produzir-dados.sql" seja chamado anteriormente.

2.2 Arquivos de Código-Fonte

Os arquivos de código-fonte estão contidos no diretório **OperacoesBanco/**, são eles:

- OperacoesBanco.h

  Arquivo que contém funções específicas para obtenção, adição e alteração de dados ao banco de dados.

- OperacoesBanco-Visualizacao.h

  Arquivo que contém funções específicas para obtenção, adição e alteração de dados ao banco de dados, especificamente relacionados a visualizações.

- OperacoesBanco-FuncoesGenericas.h

  Arquivo que contém funções generalizadas, utilizadas por todo o restante do código, como conectarBanco().

3 Arquivos responsáveis pela lógica

Os arquivos a seguir, são responsáveis pela lógica da interface. Responsáveis como, por exemplo, checar se os parâmetros de entrada são viáveis, interpretá-los, montar querys para serem executadas em banco de dados. Também cuidam de permissões de usuário, fazem limitações de utilização de recursos simultâneamente, checam conexão com o banco e coisas do tipo.

Mas, como os arquivos de organização de lógica são, de fato, a maioria do código existente, eles precisam também, de ter subdivisões, que serão explicadas nesse documento.

De modo geral, os arquivos para controle(a lógica) estão contidos dentro dos diretórios **"Comandos/"**, **"Criptografia/"**, **"Fila/"** e **"scripts/"** (esta ultima, merece uma atenção especial) salvo algumas exceções que não se encaixaram nessa definição, que são os arquivos: **AdaptadorDeString.h**, **Server.h**, **Server.c**, **Usuario.h** e **Makefile**.

Ainda, no diretório **Comandos/**, existe mais uma subdivisão, menos aparente. São os arquivos responsáveis pela lógica de Adição de dados, Obtenção de dados, Remoção de dados, Comandos Administrativos, e arquivos de dissipação (aqueles que decide qual a operação deve ser invocada).

3.1 Arquivos da Lógica de Usuário

O arquivo que cuida da estrutura e funções de Usuario, intuitivamente é o **Usuario.h**.

Esse arquivo está localizado no diretório raiz do projeto, e possui funções de controle de usuario, tais como checagem de login e senha, verificador de permissões, construtores e destrutores.

É parte essencial do projeto, e está em frequente atualização.

3.2 Arquivos de Lógica de Adição/Alteração de dados

O arquivo principal de controle de adição e/ou alteração de dados é o arquivo **"Comandos/Comando-Adicao.h"**.

Esse arquivo cuida da lógica responsável por adicionar quaisquer tipos de dados, como informações a usuário, usuários, visualizações, produtos, empresas...

Esse arquivo também limita permissões antes de qualquer alteração.

3.3 Arquivos de Lógica de Obtenção de dados

Existem dois arquivos para controle de obtenção dos dados. São eles:

- "Comandos/Comando-Obter.h"

  Cuida da lógica de obtenção de dados gerais, como "obterTop10NovosProdutos()", ou funções para obter informações de produto, id de localização e coisas do tipo.

- "Comandos/Comando-Obter_Visualizacoes.h"

  Cuida da lógica da obtenção de informações relacionadas a visualizações, como funções para obter a quantidade de visualizações de usuários anônimos ao produto X.

3.4 Arquivos de Lógica de Remoção de dados

O arquivo responsável pela lógica de remoção de dados, é o **"Comandos/Comando-Remover.h"**.

Atualmente, nenhuma função foi implementada ainda, mas em breve, será o arquivo que conterá funções para remoção de dados do banco de dados. Com o exemplo de uma função para remover o registro de um cliente do banco de dados, ou remover um produto da lista de produtos.

3.5 Arquivos de Lógica de comandos Administrativos

Esse arquivo é um pouco delicado. Segurança deve vir em primeiro lugar, pois é o comando responsável pela execução de comandos que podem ser executados diretamente no SHELL do computador que estiver executando a

interface, de fato. Lembre-se que, a interface está sendo executada em modo núcleo, portanto, tenha sensatez ao alterar esse arquivo.

Esse tão temido arquivo é o **"Comandos/Comando-Root.h"**. Suas funções foram brevemente explicadas acima, para mais informações veja o arquivo de documentação referente ao arquivo.

3.6 Arquivos de Lógica de dissipação.

Esse arquivo, interpretará a base do comando recebido, e então decidirá se está apto ou não para seguir com sua interpretação.

Também é responsável por checar se o usuário está logado, e algumas outras informações de controle.

Após a identificação de um comando viável, o controle sobre o que fazer com o comando recebido é passado, de acordo com as definições descritas no arquivo de Macros, por esse arquivo, ao arquivo correspondente, responsável pela operação identificada.

Esse arquivo é identificado por **"Comandos/interpretadorDeComandos.h"**.

3.7 Arquivos Principais (onde tudo acontece)

Esses arquivos são de grande importância, e são brevemente explicados a seguir:

- Server.c

  Arquivo que contém a função "main()" do projeto, e controla a ordem de inicialização da interface, e despacha threads a cada nova conexão identificada.

- Server.h

  Arquivo que contém a lógica de interpretação principal, e a função "Servidor", responsável por toda a interpretação, leitura de dados do cliente, e retorno ao mesmo.

  Também contém algumas poucas macros, necessárias para a organização de algumas variáveis específicas. É nesse arquivo que a estrutura de Usuario é de fato, armazenada, em tempo de execução. Esse código chama o **"Comandos/interpretadorDeComandos.h"** caso julgue necessário.

- Makefile

  Arquivo responsável pela lógica dos arquivos de compilação, ele contém os scripts de ordenação de arquivos e dependências de código que são resolvidas em tempo de compilação.

  Esse arquivo é responsável pela compilação do projeto, de modo geral, incluindo sua documentação.

- AdaptadorDeString.h

  Esse arquivo cuida de adaptar strings que venham mal configuradas, com acentuação e coisas do tipo, a fim de aumentar a segurança do projeto. Ele é usado em quase todos os códigos do projeto.

  Também é responsável pela conversão de inteiros para strings, comparações otimizadas de tamanho de strings e detecção de mensagens de escape. Esse arquivo também inclui o "Fila/Fila.h", necessário para uma adaptação de strings de forma otimizada e simplificada utilizando filas.

4 Arquivos de Criptografia

Esses arquivos, são responsáveis pela criptografia de mensagens recebidas e/ou enviadas, utilizando o algorítimo RSA para isso.

Também é responsável pela criação de novas chaves RSA para cada usuário, e aceitação de conexão segura. Estão localizados em **"Criptografia/"**. O arquivo principal é **"Criptografia/Criptografia.h"**

5 Arquivos responsáveis por Macros

Como, a maioria das macros utilizadas pelo projeto estão contidas dentro dos arquivos de Comandos, foi então gerado um arquivo de definição de macros de compilação. Utilizadas a fim de diminuir a quantidade de memoria utilizada em tempo de execução, e otimizar algumas operações.

O arquivo principal para definições de Macros:

- "Comandos/Comandos.h"

  Esse arquivo, futuramente será separado em alguns outros arquivos de definição de macros de ambiente, de erro, de retorno, e de controle.

Esse arquivo contém a maioria das macros necessárias, como string para conexão com banco de dados, números de códigos definidos, mapeamento de erros e de retornos, e coisas do tipo.

6 Arquivos de Script

São os arquivos localizados no diretório **"scripts/"**, eles contém funções que podem ser executadas num processo distinto, como ao exemplo, confirmar e-mails recebidos como login, ou monitorar o banco de dados. Os arquivos atualmente são:

- ConfirmadorDeEmail/confirmador.rb

  Script responsável por enviar e-mails e confirmar se eles são, de fato, viáveis e existentes. É invocado pela interface. Escrito em RUBY

- DatabaseMonitor/monitor.rb

  Script responsável por monitorar o banco de dados, gerando LOGs no mesmo, para obter a "quantidade de visualizações naquele momento", é chamado assim que a interface é iniciada. Escrito em RUBY.

- SimuladorDeCriacaoDeProdutos/simulador.rb

  Script que simula uma série de produtos que seriam criados em tempos diferentes, diretamente em banco de dados. Útil para testes que envolvem ordenação por data. Escrito em RUBY.

- ArmazenamentoDePacotes-Teste/server.rb

  Script que atua como uma espécie de servidor de arquivos, de forma extremamente simplificada, utilizando de socket TCP puro para a transferencia de arquivos. Escrito em RUBY.

- ArmazenamentoDePacotes-Teste/cliente.rb

  Script que atua como uma espécie de cliente para o servidor de arquivos, de forma extremamente simplificada, utilizando de socket TCP puro para a transferencia de arquivos. Utilizado para testar o "ArmazenamentoDePacotes-Teste/server.rb". Escrito em RUBY.

Desejo-lhe uma excelente experiência com o Visu-RA.