Rafael Oliveira Prado Nascimento

Ensino de programação baseado em TDD

Belo Horizonte, Minas Gerais
2024

Rafael Oliveira Prado Nascimento

Ensino de programação baseado em TDD

Relatório Final POC 2

Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Orientador: Thiago Ferreira de Noronha
Coorientador: Natã Goulart da Silva

Belo Horizonte, Minas Gerais
2024

Sumário

1 Introdução

O ensino de linguagens de programação de computadores, em todos os seus níveis, exige dos alunos a resolução de um grande número de exercícios, a fim de que eles desenvolvam o raciocínio lógico sequencial necessário para solucionar problemas computacionais. A correção destes exercícios demanda muito tempo dos professores e dos monitores das disciplinas. Com o crescimento do número de alunos nas disciplinas de ensino de programação, a correção manual de todos exercícios se tornará impraticável.

O projeto visa resolver o desafio de modernizar e aprimorar o sistema de correção de exercícios em disciplinas de programação proposto e desenvolvido na pós-graduação do professor Natã Goulart da Silva, para que as listas já existentes passem a utilizar um novo framework e refatorar os códigos.

Nesse contexto a aplicação Easy... (2010) foi criada com o intuito de:

- Economia de Recursos: A automação da correção economiza recursos financeiros e reduz a carga de trabalho dos professores e monitores, permitindo que se concentrem em dúvidas mais complexas apresentadas pelos alunos.
- Melhorias no Aprendizado: A correção automática proporciona feedback imediato aos alunos, acelerando seu aprendizado e ajudando-os a desenvolver habilidades de programação.
- Padronização e Boas Práticas: As listas implementadas no Easytesting introduzem os alunos à boas praticas de programação desde o início de sua formação, preparando-os melhor para o mercado de trabalho.

1.1 Objetivos Gerais

O objetivo geral deste projeto é modernizar e expandir o sistema de listas de exercícios autocorrigíveis, oferecendo uma solução eficaz e escalável. Isso será alcançado por meio da migração para um framework de testes mais moderno que o utilizado no momento do desenvolvimento da primeira versão da lista, e pela criação de novas listas de exercícios.

1.2 Objetivos Específicos

Durante a etapa da POC 2, o objetivo foi desenvolver os seguintes aspectos:

- Migração para um Framework mais moderno: Refatorar o sistema existente para substituir o framework de testes anterior (Google... (2008)) para um framework mais moderno e eficiente.
- Revisão das Listas de Exercícios: Realizar uma revisão completa das listas de exercícios existentes, garantindo que os exercícios estejam atualizados e alinhados com as ementas das disciplinas.
- Criação de Novas Listas: Desenvolver novas listas de exercícios, abordando diferentes níveis de dificuldade e temas relevantes para disciplinas de programação.
- Documentação e Licenciamento: Preparar documentação completa para facilitar o uso do sistema e garantir seu licenciamento adequado, tornando-o acessível como um projeto de código aberto.

2 Referencial Teórico

Alguns sistemas e tecnologias que são usadas atualmente para ensino de linguagens de programação, que utilizam sistemas de Juízes Online:

- O Beecrowd (2021) é uma plataforma brasileira que oferece desafios de programação competitiva utilizando o método de juiz online para avaliação dos desafios.
- O Spoj (2004) é uma plataforma online que oferece uma ampla variedade de problemas de programação e algoritmos para que os programadores possam praticar e aprimorar suas habilidades que utiliza o método de juiz online para avaliar as atividades em sua plataforma.
- Exercism (2013) é uma plataforma online que oferece exercícios práticos de programação em várias linguagens que utiliza métodos de juizes online para avaliar as atividades.
- Moodle (2002) é uma plataforma de aprendizagem projetada para fornecer aos educadores, administradores e alunos um único sistema robusto, seguro e integrado para criar ambientes de aprendizagem personalizados. Essa plataforma é atualmente utilizada na Universidade Federal de Minas Gerais que utiliza a função VPL (Virtual Programming Lab) para correções de exercicios em disciplinas da UFMG.

Todas essas plataformas acima são online e utilizam sistemas de Juízes Online para avaliar exercícios propostos. A avaliação de juízes online envolve os alunos escrevendo códigos para resolver problemas e enviando esses programas pela Internet a um servidor de testes. O servidor trata o programa como uma "caixa preta", executando casos de teste e, se os resultados coincidirem com o esperado, o juiz online aprova a correção do programa do aluno.

Outro referencial é a aplicação que será modernizada e refatorada neste trabalho, que é o Easy... (2010), um projeto criado pela iniciativa do professor Natã Goulart da Silva da Universidade Federal de Lavras e do professor Thiago Noronha da Universidade Federal de Minas Gerais. Esse projeto, diferentemente dos sistemas de Juízes Online, não precisa de um servidor online para que seja executado e utiliza de testes unitários para a correção das listas de exercícios implementadas na aplicação.

3 Metodologia

Como a ferramenta citada neste projeto já existe, a idéia é evoluir e aprimorar a aplicação, pois após 10 anos da idealização e implementação o Easy... (2010) se tornou defasado em tecnologia e conteúdo de atividades, sendo necessária a refatoração dos códigos existentes e modernização do framework de testes utilizado.

Para a execução do projeto, os seguintes passos serão seguidos:

- *M*igração dos códigos: Inicialmente, se faz necessária a migração dos códigos do antigo repositório para o GitHub (2008), a fim de facilitar a manutenção e versionamento dos códigos.
- *E*studo de tecnologias: Nesta etapa, será desenvolvido um estudo para entender quais os melhores frameworks de teste atuais para C++, ex: (Doctest... (2014)).
- *R*efatoraçõs e reescritas de código: Nesta etapa, após o entendimento do ambiente e das tecnologias utilizadas, serão iniciadas as refatorações nas listas des exercícios já implementadas a fim de trocar o framework de teste.
- *T*estes: Nesta etapa, serão realizados testes para garantir que o sistema esteja funcional e de fácil utilização. Esses testes serão feitos resolvendo os exercicios propostos.

4 Atividades conduzidas

Desde o inicio do projeto foi definido um cronograma para o POC1, onde deveria desenvolver uma nova lista e refatorar 30% das listas já existentes. O primeiro passo antes do desenvolvimento e manutenção das listas foi estudar o novo framework de testes Doctest... (2014), para que na etapa de desenvolvimento não houvesse problemas durante a codificação. Durante a POC2 foram refatoradas as listas restantes, representando 70% das listas, trocando o framework utilizado para o Doctest... (2014), foram testadas todas as listas, o github atualizado, contendo um novo texto explicando o funcionamento do projeto e mostrando exemplos de execução das listas.

4.1 Etapas e Cronograma

O cronograma seguido durante a POC2 foi:

Data início	Atividade
05/04/24	Continuar a migração das listas
13/05/24	Preparação do video de apresentação parcial
14/05/24	Continuação da migração das listas
10/07/24	Finalização da migração
15/07/24	Testes de todas as listas
25/07/24	Preparação do video de apresentação final
30/07/24	Preparação do relatório final

5 Conclusão

Ao fim do POC2, foi notório perceber a diferença que o novo framework traz às listas antigas, tornando mais facil e rápida a execução das listas. Podendo-se notar que com o Doctest... (2014) fica mais facil de desenvolver novas listas pela sintaxe simplificada das funções do framework. A migração para o github facilita a manutenção e uso do projeto, o tornando mais atrativo a ser utilizado em disciplinas da graduação.

Referências

BEECROWD. 2021. <https://www.beecrowd.com.br/judge/pt/login>. Acesso em 11 de setembro de 2023. 5

DOCTEST C++. 2014. <https://github.com/doctest/doctest>. Acesso em 11 de setembro de 2023. 6, 7, 8

EASY Testing. 2010. <https://code.google.com/archive/p/easytesting/>. Acesso em 11 de setembro de 2023. 3, 5, 6

EXERCISM. 2013. <https://exercism.org/>. Acesso em 11 de setembro de 2023. 5

GITHUB. 2008. <https://github.com/>. Acesso em 11 de setembro de 2023. 6

GOOGLE Test. 2008. <https://google.github.io/googletest/>. Acesso em 11 de setembro de 2023. 4

MOODLE. 2002. <https://moodle.org/plugins/mod_vpl>. Acesso em 11 de setembro de 2023. 5

SPOJ. 2004. <https://br.spoj.com/>. Acesso em 11 de setembro de 2023. 5

6 Apêndice

6.1 Arquivo agenda-test.h

Arquivo com a codifição dos testes da lista agenda após a modificação para o doctest

```
// Copyright 2014 Universidade Federal de Minas Gerais (UFMG)

#include <../doctest/doctest.h>
#include "./src/agenda.h"

const int MAX = 50;

TEST_CASE("Testa_LerNomeDoTeclado") {
  // Print dos resultados do teste
  INFO("-------------------------------------");
  INFO("Teste: Testa_LerNomeDoTeclado");
  INFO("-------------------------------------");
  INFO("Valores de entrada: ");
  INFO("esperado: Joao da Silva");
  INFO("obtido: " << LerNomeDoTeclado());
  INFO("-------------------------------------");

  string esperado = "Joao da Silva";
  string obtido = LerNomeDoTeclado();
  CHECK_EQ(obtido, esperado);
}

TEST_CASE("Testa_LerAniversarioDoTeclado") {
  // Print dos resultados do teste
  INFO("-------------------------------------");
  INFO("Teste: Testa_LerAniversarioDoTeclado");
  INFO("-------------------------------------");
  INFO("Valores de entrada: ");
  INFO("esperado: 20/07/1990");
  INFO("obtido: " << LerAniversarioDoTeclado());
```

```
  INFO("----------------------------------------");

  string esperado = "20/07/1990";
  string obtido = LerAniversarioDoTeclado();
  CHECK_EQ(obtido, esperado);
}

TEST_CASE("Testa_LerTelefoneDoTeclado") {
  // Print dos resultados do teste
  INFO("----------------------------------------");
  INFO("Teste: Testa_LerTelefoneDoTeclado");
  INFO("----------------------------------------");
  INFO("Valores de entrada: ");
  INFO("esperado: (31) 99999-9999");
  INFO("obtido: " << LerTelefoneDoTeclado());
  INFO("----------------------------------------");

  string esperado = "(31) 99999-9999";
  string obtido = LerTelefoneDoTeclado();
  CHECK_EQ(obtido, esperado);
}

TEST_CASE("Testa_LerContatoDoTeclado") {
  // Print dos resultados do teste
  INFO("----------------------------------------");
  INFO("Teste: Testa_LerContatoDoTeclado");
  INFO("----------------------------------------");
  INFO("Valores de entrada: ");
  INFO("esperado:");
  INFO("nome: Joao da Silva");
  INFO("telefone: (31) 99999-9999");
  INFO("aniversario: 20/07/1990");
  INFO("obtido:");
  INFO("nome: " << LerContatoDoTeclado().nome);
  INFO("telefone: " << LerContatoDoTeclado().telefone);
  INFO("aniversario: " << LerContatoDoTeclado().aniversario);
  INFO("----------------------------------------");

  Contato contatoEsperado;
```

```
  contatoEsperado.nome = "Joao da Silva";
  contatoEsperado.telefone = "(31) 99999-9999";
  contatoEsperado.aniversario = "20/07/1990";

  Contato contatoObtido = LerContatoDoTeclado();
  CHECK_EQ(contatoEsperado, contatoObtido);
}

TEST_CASE("Testa_ConsultarContatoPorNome") {
  // Print dos resultados do teste

  Contato contato;
  contato.nome = "Joao da Silva";
  contato.telefone = "(31) 99999-9999";
  contato.aniversario = "20/07/1990";
  INFO("-----------------------------------");
  INFO("Teste: Testa_ConsultarContatoPorNome");
  INFO("-----------------------------------");
  INFO("Valores de entrada: ");
  INFO("esperado:");
  INFO("nome: Joao da Silva");
  INFO("telefone: (31) 99999-9999");
  INFO("aniversario: 20/07/1990");
  INFO("obtido:");
  INFO("nome:"<<
  ConsultarContatoPorNome("Joao da Silva", 1, &contato).nome);

}

TEST_CASE("Testa_InserirContatoNaAgenda") {
  // Print dos resultados do teste
  Contato contato;
  contato.nome = "Joao da Silva";
  contato.telefone = "(31) 99999-9999";
  contato.aniversario = "20/07/1990";

  int n = 0;
  Contato agenda[MAX];
```

```
  InserirContatoNaAgenda(contato, &n, agenda);

  INFO("-------------------------------------");
  INFO("Teste: Testa_InserirContatoNaAgenda");
  INFO("-------------------------------------");
  INFO("Valores de entrada: ");
  INFO("esperado:");
  INFO("nome: Joao da Silva");
  INFO("telefone: (31) 99999-9999");
  INFO("aniversario: 20/07/1990");
  INFO("obtido:");
  INFO("nome: " << agenda[n - 1].nome);
  INFO("telefone: " << agenda[n - 1].telefone);
  INFO("aniversario: " << agenda[n - 1].aniversario);
  INFO("-------------------------------------");



  Contato contatoEsperado = contato;
  Contato contatoObtido = agenda[n - 1];
  CHECK_EQ(contatoEsperado, contatoObtido);
}

TEST_CASE("Testa_ApagarContato") {
  // Print dos resultados do teste
  Contato contato;
  contato.nome = "Joao da Silva";
  contato.telefone = "(31) 99999-9999";
  contato.aniversario = "20/07/1990";

  int n = 1;
  Contato agenda[MAX];
  agenda[0] = contato;

  ApagarContato("Joao da Silva", &n, agenda);

  INFO("-------------------------------------");
  INFO("Teste: Testa_ApagarContato");
  INFO("-------------------------------------");
```

```
  INFO("Valores de entrada: ");
  INFO("esperado:");
  INFO("nome: Joao da Silva");
  INFO("telefone: (31) 99999-9999");
  INFO("aniversario: 20/07/1990");
  INFO("obtido:");
  INFO("nome: " << agenda[0].nome);
  INFO("telefone: " << agenda[0].telefone);
  INFO("aniversario: " << agenda[0].aniversario);
  INFO("-----------------------------------");



  Contato contatoEsperado;
  contatoEsperado.nome = "";
  Contato contatoObtido = agenda[0];
  CHECK_EQ(contatoEsperado, contatoObtido);
}
```