Relatório: Sistema de Gerenciamento de Biblioteca em Haskell

Victor Martins de Sousa, Pedro Henrique Noronha,
João Pedro Rodrigues de Faria

Maio de 2025

1 Introdução.

Este relatório apresenta o desenvolvimento de um sistema de gerenciamento de biblioteca implementado em Haskell, como parte do segundo trabalho da disciplina de Programação Funcional. O sistema, construído por Victor Martins, Pedro Henrique e João Pedro, permite o cadastro e gerenciamento de livros, usuários e empréstimos por meio de uma interface de linha de comando, com dados persistidos em arquivos de texto. Nosso objetivo foi atender aos requisitos especificados, aplicando conceitos de programação funcional, como funções puras e tipos algébricos. Neste documento, detalhamos a estrutura do projeto, a divisão de tarefas, os desafios enfrentados e as decisões tomadas durante o desenvolvimento.

2 Descrição do Projeto.

O sistema gerencia uma biblioteca, oferecendo funcionalidades como cadastro de livros (com título, autor, ano e código único), cadastro de usuários (com nome, matrícula e e-mail), registro de empréstimos e devoluções, e geração de relatórios. Os relatórios incluem a listagem de livros emprestados e disponíveis, o histórico de empréstimos de um usuário e a exibição de livros com listas de espera. A interface é baseada em um menu principal com opções numeradas, como **Cadastrar livros**, **Empréstimo e Devolução** e **Salvar e Sair.** Ao salvar, o sistema verifica a existência do arquivo **biblioteca.txt**, perguntando ao usuário se deseja criar um novo arquivo ou sobrescrever o existente, e exibe a mensagem **"Até a próxima!"** ao encerrar.

O programa foi implementado em Haskell puro, utilizando funções puras e tipos de dados algébricos para representar livros, usuários e empréstimos. O código foi organizado em módulos separados, como gerenciamento de livros, usuários e empréstimos, para facilitar a manutenção e o desenvolvimento colaborativo. O código-fonte está disponível no repositório https://github.com/VictorMS-200/bibliotecaHaskell.

3 Divisão de Tarefas.

A equipe dividiu as responsabilidades com base nas competências de cada membro, com Victor assumindo a maior parte do desenvolvimento. A divisão foi a seguinte:

- **Victor**: Responsável pela arquitetura geral do sistema, incluindo a implementação do menu principal, a lógica de manipulação de arquivos (leitura e escrita), a definição dos tipos de dados (Livro, Usuário, Empréstimo) e a maioria das funcionalidades, como cadastro de livros e usuários, gerenciamento de empréstimos e listas de espera. Também coordenou a integração dos componentes.
- **Pedro Henrique Noronha**: Implementou as funções listarEmprestadosEDisponiveis (exibe livros emprestados e disponíveis), historicoEmprestimoUsuario (mostra o histórico de empréstimos de um usuário) e listarListaEsperaUsuarios (lista usuários em espera por um livro). Contribuiu com testes dessas funções.
- **João Pedro**: Desenvolveu as funções de edição editarLivroTitulo, editarLivroAutor, editarLivroAno (para alterar informações de livros) e editarUsuarioNome, editarUsuarioMatricula, editarUsuarioEmail (para alterar informações de usuários). Auxiliou na integração e validação dessas funções.
- **Equipe**: Todos participaram dos testes finais, assegurando que o sistema estivesse livre de erros, como tentativas de operar com livros ou usuários inexistentes, e colaboraram na documentação do projeto.

Victor desempenhou um papel central na liderança técnica, enquanto Pedro Henrique e João Pedro focaram em funcionalidades específicas, contribuindo para a completude do sistema.

4 Desafios Enfrentados.

O desenvolvimento apresentou desafios significativos que exigiram esforço coletivo para serem superados. Abaixo, listamos os principais obstáculos enfrentados pela equipe:

1. **Planejamento do Projeto**: Estruturar o sistema em funções modulares foi um processo complexo. Definir as responsabilidades de cada módulo e garantir que interagissem corretamente demandou várias discussões e revisões até alcançarmos uma organização clara, com módulos dedicados a livros, usuários e empréstimos.
2. **Manipulação de Arquivos**: Implementar a lógica de leitura e escrita em arquivos de texto foi desafiador, especialmente para verificar a existência do arquivo e gerenciar as opções de criação ou sobrescrição. A manipulação de entrada e saída em Haskell requereu estudo aprofundado para alinhar-se aos princípios da programação funcional.
3. **Gerenciamento de Listas de Espera**: Desenvolver a funcionalidade de listas de espera foi particularmente difícil, devido à necessidade de evitar duplicatas de usuários e exibir a lista de forma clara. A lógica para atualizar a lista após devoluções também apresentou problemas iniciais, resolvidos com testes extensivos.
4. **Definição de Tipos de Dados**: Estruturar os tipos algébricos (Livro, Usuário, Empréstimo) de maneira eficiente, garantindo que todas as funções os utilizassem adequadamente, foi um processo iterativo que exigiu ajustes para atender a todos os casos de uso.
5. **Implementação e Integração de Funções**: A criação de funções, como as de listagem de relatórios e edição de dados, foi complexa devido à necessidade de tratar

casos especiais, como listas vazias ou entradas inválidas. Além disso, integrar essas funções ao sistema principal, assegurando compatibilidade com o restante do código, demandou coordenação e revisões.

Esses desafios foram superados por meio de colaboração, estudo contínuo e testes rigorosos, o que fortaleceu nosso domínio de Haskell e programação funcional.

5 Decisões de Projeto.

Durante o desenvolvimento, tomamos decisões estratégicas para atender aos requisitos e otimizar a implementação:

- **Modularização**: O código foi organizado em módulos separados (por exemplo, Funcoes.hs, Persistencia.hs, Tipos.hs e Menu.hs) para facilitar o desenvolvimento colaborativo e a manutenção.
- **Interface Intuitiva**: A interface de linha de comando foi projetada com um menu principal claro e submenus que permitem retornar ao menu anterior, garantindo facilidade de uso.
- **Validação e Testes**: Implementamos verificações para casos de erro, como tentativas de operar com livros ou usuários inexistentes, e realizamos testes extensivos para assegurar a robustez do sistema.
- **Implementação de funções avançadas:** Para algumas funções mais complexas como a de verificação de existência de arquivo, foram utilizadas funções importadas que vão além das que foram apresentadas nas aulas. Como por exemplo a função doesFileExist, que verifica se um arquivo existe ou não.

Essas decisões contribuíram para a organização do projeto e o cumprimento dos requisitos estabelecidos.

6 Conclusão.

O desenvolvimento do sistema de gerenciamento de biblioteca foi uma experiência desafiadora e enriquecedora. A implementação em Haskell nos permitiu aprofundar nosso entendimento de conceitos de programação funcional, como funções puras, tipos algébricos e modularização. Apesar dos obstáculos, a colaboração entre os membros da equipe e o empenho em superar as dificuldades resultaram em um sistema funcional e bem estruturado. O código-fonte está disponível no repositório https://github.com/VictorMS-200/bibliotecaHaskell, pronto para apresentação e avaliação. Estamos confiantes de que o projeto atende às expectativas da disciplina.