Relatório do Sistema de Gerenciamento de Biblioteca em Haskell

1. Introdução

Este relatório descreve o desenvolvimento de um sistema de gerenciamento de biblioteca implementado em Haskell, conforme os requisitos do segundo trabalho de Programação Funcional. O sistema permite o cadastro de livros e usuários, gestão de empréstimos e devoluções, além de gerar diversos relatórios. O projeto foi desenvolvido seguindo os princípios da programação funcional, com ênfase em tipos algébricos, funções puras e imutabilidade.

2. Principais Decisões de Projeto

2.1 Estrutura de Módulos

- Tipos.hs: Define os tipos algébricos fundamentais do sistema (Livro, Usuario, Emprestimo, etc.)
- Livros.hs: Implementa as operações relacionadas ao cadastro e gestão de livros
- Usuarios.hs: Contém as funções para gerenciamento de usuários
- Emprestimos.hs: Gerencia empréstimos, devoluções e listas de espera
- Relatorios.hs: Implementa as funcionalidades de geração de relatórios
- Persistencia.hs: Responsável pela leitura e escrita dos dados em arquivos
- Main.hs: Coordena o fluxo principal do programa e a interface de usuário

2.2 Modelagem de Dados

Os tipos algébricos foram cuidadosamente projetados para representar os conceitos do domínio:

```
data Livro = Livro { titulo :: String, autor :: String, ano :: Integer, idLivro :: Int }
data Usuario = Usuario { nome :: String, matricula :: String, email :: String }
data Emprestimo = Emprestimo { dataEmprestimo :: DiaMesAno, livroE :: Livro, usuarioE :: Usuario, devolvido :: Bool }
data Devolucao = Devolucao{ dataDevolucao   :: DiaMesAno, livroD :: Livro, usuarioD :: Usuario} deriving (Show, Eq)
data Espera = Espera{ usuarioEsp   :: Usuario, livroEsp :: Livro} deriving (Show, Eq)
```

Esta modelagem permite:

- Clareza na representação dos dados
- Facilidade de manipulação através de pattern matching
- Garantia de integridade através do sistema de tipos

2.3 Princípios Funcionais

- Imutabilidade: Todas as estruturas de dados são imutáveis
- Funções puras: A maioria das funções não possui efeitos colaterais
- Recursão e funções de alta ordem: Uso intensivo de map, filter, foldl, etc.
- Separação clara entre IO e lógica: As operações de I/O ficam isoladas em poucas funções

3. Divisão de Tarefas

3.1 Pablo Rodrigues - Persistência em Arquivos

- Implementou as funções para serialização e desserialização dos dados
- Desenvolveu as rotinas para salvar e carregar o estado do sistema
- Criou funções para converter tipos em strings e vice-versa

3.2 Ruan Pablo - Módulo Main e Interface

- Implementou o menu principal e submenus
- Desenvolveu o fluxo de controle do programa
- Criou funções auxiliares para interação com o usuário

3.3 Gabriel Valentim - Tipos, Empréstimos e Devoluções

- Definido os tipos algébricos fundamentais
- Implementado a lógica de empréstimos e devoluções
- Desenvolvido o sistema de lista de espera
- Criado funções para validação de operações

3.4 Pedro De Colla - Livros, Usuários e Relatórios

- Implementado o CRUD de livros e usuários
- Desenvolvido as funções de filtragem e ordenação
- Criado os relatórios do sistema
- Implementado validações de unicidade (matrícula, email, ID)

4. Dificuldades Encontradas

4.1 Gerenciamento de Estado em um Paradigma Funcional

- Desafio em manter o estado consistente sem variáveis mutáveis
- Solução: Uso de recursão para passar o estado atualizado entre chamadas

4.2 Persistência de Dados

- Complexidade em serializar estruturas de dados complexas para arquivos texto
- Solução: Desenvolvimento de funções específicas de conversão para cada tipo

4.3 Listas de Espera

- Lógica complexa para evitar duplicação e manter a ordem
- Solução: Implementação de funções de verificação e filtragem específicas

4.4 Interface de Usuário

- Dificuldade em criar menus hierárquicos de forma puramente funcional

- Solução: Uso de funções recursivas e composição de funções IO

5. Funcionalidades Implementadas

5.1 Cadastro de Livros

- Adição, remoção e listagem de livros
- Filtros por título, autor, ano e ID
- Validação de ID único

5.2 Cadastro de Usuários

- Adição, remoção e listagem de usuários
- Validação de matrícula e email únicos
- Ordenação por nome e matrícula

5.3 Empréstimos e Devoluções

- Registro de empréstimos com verificação de disponibilidade
- Sistema de lista de espera para livros emprestados
- Marcação de devoluções e atualização de status

5.4 Relatórios

- Listagem de empréstimos ativos
- Histórico por usuário
- Livros disponíveis e emprestados
- Livros com lista de espera

6. Conclusão

O desenvolvimento deste projeto proporcionou uma valiosa experiência prática na aplicação dos conceitos de programação funcional. As principais lições aprendidas incluem:

1. A importância de uma boa modelagem de tipos para garantir a correção do sistema
2. Os benefícios da separação clara entre lógica e I/O
3. As vantagens da imutabilidade para evitar efeitos colaterais indesejados
4. O poder das funções de alta ordem para manipulação de coleções

O sistema cumpre todos os requisitos funcionais e técnicos especificados, demonstrando a eficácia da programação funcional para o desenvolvimento de aplicações do mundo real. A modularização do código facilita a manutenção e futuras extensões.

7. Instruções de Compilação e Execução

1. Certifique-se de ter o GHC (Glasgow Haskell Compiler) instalado
2. Navegue até o diretório do projeto no terminal
3. Compile o programa com o comando: ghc -o biblioteca Main.hs
4. Execute o programa com: ./biblioteca

O sistema criará ou carregará automaticamente o arquivo "biblioteca.txt" para persistência dos dados.