Decodificação e manipulação de arquivos CSV utilizando programação a nível de tipos

Nome: Lucas Silva Amorim **RA**: 11201720968 **Video**: Apresentação Youtbe

`type-safety-csv` é uma biblioteca para leitura e manipulação de csv 's, implementando métodos que garantem a segurança da API via tipagem. Desta forma, o desenvolvedor não precisa se preocupar com erros em tempo de execução ao lidar com um arquivo csv (seja por tentar acessar uma coluna inexistente, ou receber uma registro com um tamanho inesperado).

Funcionalidades disponíveis

- Decodificação de arquivos separados por vírgulas em estruturas de dados seguras.
- Acesso **seguro** as linhas por índice.
- Acesso **seguro** as colunas por índice..
- Acesso a cabeçalho (se o arquivo possuir um).

Funcionalidades que podem ser implementadas

- Buscar colunas pelo nome.
- Filtros dinâmicos.
- Coerce do CSV para tipo de dado específico.
- Induzir o tamanho das linhas e colunas.

Dificuldades, surpresas e destaques

Certamente, a maior **dificuldade** encontrada durante o desenvolvimento foi a tentativa de converter os dados "não seguros" (inteiros, strings) nas estruturas de dados criadas. Em especial lidar com os problemas relacionados ao fato de algumas vezes, precisar manipular os dados do **vetor indexado** sem necessariamente possuir o tamanho total do vetor. Um dos problemas não resolvidos, foi a conversão valores `Int` (tamanho das linhas e colunas do csv) para o tipo `SNat n`, pois o `GHCI` não é capaz de deduzir o valor esperado para `n` somente com a entrada fornecida (algo como `forsome n`). A solução atuall obriga o desenvolvedor a informar o tamanho do `csv` na criação do arquivo.

Entre os **destaques** estão o uso dos `GADT's` para lidar com as diferenças entre os tipos possíveis de arquivos no formato `csv`. Também as soluções dadas para a criação dos `Sized Vectors` a partir dos vetores de `Strings`, e a implementação do tipo `Fin` (Finity) para limitar o tamanho dos vetores de acordo com o número de linhas e colunas esperados.

Como testar

- Para executar os testes da aplicação: `$ stack test`
- Caso queira observar o funcionamento da biblioteca, a função `main` no arquivo `Main.hs` importa algumas funções "demo", basta chamar qualquer uma delas na função `main`. Exemplo:

```
main :: IO ()
main = loadingCSVToType
```