DEADLINES

Alexandre Zeferino Lima[1], *Tiago Henrique Simionato Machado*[2]

[1] *11201720542, alexandre.lima@aluno.ufabc.edu.br*
[2] *11201810899, tiago.simionato@aluno.ufabc.edu.br*

Resumo: Este relatório apresenta o projeto "DEADLINES", criado em Haskell, para disciplina de Paradigmas de Programação. A proposta do projeto é criar um calendário em que seja possível visualizar e organizar compromissos.

1 COMO UTILIZAR

Assume-se que o leitor possua um computador com uma distribuição Linux e a linguagem Haskell instalada.

```
$ sudo apt-get install libgirepository1.0-dev
libwebkit2gtk-4.0-dev libgtksourceview-3.0-dev

$ sudo apt-get install gtk2.0
```

Uma vez instaladas as bibliotecas, basta que se execute o comando "stack run" no diretório do projeto. O vídeo explicativo, com detalhes adicionais sobre as funções presentes no programa pode ser encontrado em: https://youtu.be/7ULOamWIEjc

2 DIFICULDADES, SURPRESAS E DESTAQUES

Muitos desafios surgiram durante o desenvolvimento do projeto. A primeira dificuldade foi encontrada antes mesmo de iniciá-lo, na fase de pesquisa: como criar uma interface gráfica em uma linguagem funcional como Haskell? Foi sugerido um guia realizado por alunos desta disciplina em quadrimestres anteriores[1], em que é possível ter uma ideia de como tais interfaces são criadas: usando o GTK, uma biblioteca voltada para criação de interfaces gráficas com uma vasta documentação[2].

Sabendo como tais interfaces são criadas, foi preciso também pesquisar por métodos para acelerar o processo de criação das mesmas. Tal pesquisa indicou[3] uma grande surpresa: uma ferramenta chamada "Glade"[4], criada para facilitar o desenvolvimento de interfaces para o GTK, em que o usuário tem o controle visual da interface que está criando. Ao final do desenvolvimento, a ferramenta gera um arquivo XML, o qual é lido e convertido para o GTK por meio da biblioteca GTK Builder.

Outra dificuldade foi aprender a manipular arquivos CSV com Haskell. Para isso, foi encontrada uma biblioteca chamada Cassava[5], que pode realizar a conversão dos ADT's do Haskell para os campos que são guardados no CSV e vice-versa. Tutoriais online auxiliaram na aprendizagem do uso dessa biblioteca. Entretanto, houve um problema na hora de usar a função **insertToFile**. Quando ela era chamada, seu retorno sempre indicava que não era possível escrever no arquivo, pois este se encontrava bloqueado. A solução encontrada para isso foi usar um **print** antes de chamar a função, e ela acabava funcionando.

Uma vez definido o ADT dos Eventos, na hora de realizar a ordenação por data acabou surgindo uma pequena dificuldade inesperada. Era possível usar a função **sortOn** do Haskell para organizar uma lista de *Event* de acordo com apenas uma das informações da data, pois o ano, mes e dia eram campos diferentes do ADT, então ao se organizar por mês, havia o risco de perder a organização por ano. Para se contornar isso, uma vez ordenada a lista por ano, ela foi agrupada em sub listas entre os eventos que aconteciam no mesmo ano e cada sub lista foi organizada por mês. O processo foi repetido para os dias.

Outro problema identificado depois de feita a ordenação por data, foi que no caso de eventos recorrentes, a data que ele guardava na verdade nem sempre representava a data real de sua próxima ocorrência (caso a data fosse anterior ao dia de hoje), o que causaria problemas na hora de exibir a data de sua próxima ocorrência na GUI e também faria a o evento estar na posição errada no caso de ordenação por data. Para solucionar isso foi feita a função **updateEvents** que foi usada durante a chamada de **getEvents**, assim os eventos retornados por essa última função sempre estariam atualizados e corretos. Para fazer a atualização foi utilizada a função **daysleft**, que ja calculava a quantidade de dias corretamente para eventos recorrentes que já haviam passado.

Um destaque interessante do programa é a forma como cores são atribuidas automaticamente aos eventos. Para isso foi feita uma função que calcula um ponto de um gradiente que vai do vermelho ao verde. O ponto é calculado em função dos dias restantes até a data chegar, e da urgência que o evento tem, que é definida pelo usuário.

No projeto também foi necessário saber em que dia da semana em que uma data vai cair, para exibir na GUI. O calendário gregoriano possui uma fórmula para isso, que foi usada no programa.

3 REFERÊNCIAS

[1] Dionello, C. C.; Mendonça, R. R. de. Tutorial de implementação da Biblioteca GTK2HS. Arquivo disponibilizado pelo professor da disciplina.

[2] 'gtk3: Binding to the Gtk+ graphical user interface library'. Disponível em: https://hackage.haskell.org/ package/gtk3-0.13.6. Acessado em 26 de Agosto de 2022.

[3] Ahmed, E. Developing Gnome Apps with Glade. Disponível em: https://www.muitovar.com/EddyAhmed/ GladeGtk2Hs.html. Acessado em 26 de Agosto de 2022.

[4] Glade - A User Interface Designer. Disponível em: https://glade.gnome.org/. Acessado em 26 de Agosto de 2022.

[5] CSV encoding and decoding in Haskell with Cassava. Disponível em: https://www.stackbuilders.com/blog/csv-encoding-decoding/ Acessado em 08 de Agosto de 2022.