Universidade Federal do Rio Grande do Sul — Instituto de Informática
INF01120 – Técnicas de Construção de Programas — 2018/2
Diego Dimer Rodrigues — 00287690
Augusto Peiter — 00287685
Eduardo Paim — 00277322

Trabalho Final - Sistema de Biblioteca

O que é

Escolhemos criar um sistema que poderia ser usado em bibliotecas para controle de alugueis de livros. A motivação do assunto veio de percebermos que muitas bibliotecas ainda tem o controle dos alugueis dos seus livros por fichas de papel guardadas junto aos livros. Baseamos parte do sistema também no SABi[1], da UFRGS, onde podemos consultar livros e nossos aluguéis.

O que usamos

Persistência de dados

Para guardar os registros, usamos um banco de dados gratuito do site Heroku[2]. Se o sistema fosse ser *realmente* implementado e vendido, procuraríamos uma base de dados paga para termos mais segurança e suporte. Usamos também a linguagem SQL para acessar e manipular dados do banco de dados. A conexão com a base de dados foi pelo JDBC[3] usando as credenciais do heroku. Usamos uma classe Database que implementa uma interface DatabaseInterface, a implementação facilita podermos mudar, se um dia quisessemos usar outro método, poderíamos apenas reimplementar a interface DatabaseInterface e pouco código precisaria ser modificado.

Documentação

Para a documentação, usamos Javadoc[4], que extrai direto do código todos os métodos e seus comentários. Na documentação, encontram-se todas as descrições dos métodos e árvores de herança das classes.

[1] https://sabi.ufrgs.br
[2] "https://dashboard.heroku.com/"
[3] https://www.oracle.com/technetwork/java/javase/jdbc/index.html
[4] https://docs.oracle.com/javase/8/docs/technotes/tools/windows/javadoc.html

Interfaces Gráficas

Para interface gráfica, usamos a extensão do eclipse windowbuilder[5], onde todas interfaces são JFrames[6].

Versionamento

Usamos o GitHub para versionamento do nosso software. Criamos um repositório privado em que todos do grupo foram adicionados como contribuidores.

Singleton

Usamos na classe de database singleton, para tornar única a conexão, representada pela variavel *conec* do tipo *Connection*. Para isso, a variável começa com null, e quando o construtor da classe é chamado pela primeira vez, estabelece-se a conexão. Depois disso, nada é feito quando chama-se o construtor novamente.

Interfaces

Para possíveis alterações de implementação, foram usadas interfaces para Usuário, Funcionário e Database. Qualquer alteração na tecnologia usada nessas classes, seriam apenas implementações dessas interfaces, garantindo o funcionamento do software.

Testes

Usamos o JUnit 4 para testes. O problema de testes quando envolve database é que todo teste pode ser executado somente uma vez (a função de adicionar algo numa tabela, se aquela linha é única, só pode ser adicionada uma vez). Então, a estratégia adotada foi comentar o teste após ele passar e ser garantido que está funcionando. Usamos setup antes de todos os métodos para conectar a base de dados, e teardown para fechar a conexão após os testes. A implementação dos métodos foi baseada em TDD, escrevíamos o teste e depois implementávamos o método na classe Database.

Diagramas UML

Os diagramas de classes e sequências foram atualizados da etapa 1 para 2. O sistema mudou, excluímos algumas classes e fizemos métodos diferentes do esperado. A essencia do projeto, porém, continuou igual.

[5]https://www.eclipse.org/windowbuilder/

[6]https://docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html