UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Alexandre Vieira dos Santos

Sistema *Online* de Distribuição de Disciplinas: Manutenção e Implementação de Novos Requisitos

Uberlândia, Brasil

2018

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Alexandre Vieira dos Santos

Sistema *Online* de Distribuição de Disciplinas: Manutenção e Implementação de Novos Requisitos

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como requisito exigido parcial à obtenção do grau de Bacharel em Sistemas de Informação.

Orientador: Prof. Dr. Bruno Augusto Nassif Travençolo

Universidade Federal de Uberlândia – UFU

Faculdade de Computação

Bacharelado em Sistemas de Informação

Uberlândia, Brasil

2018

Alexandre Vieira dos Santos

Sistema *Online* de Distribuição de Disciplinas: Manutenção e Implementação de Novos Requisitos

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como requisito exigido parcial à obtenção do grau de Bacharel em Sistemas de Informação.

Trabalho aprovado.

Uberlândia, Brasil, 08 de agosto de 2018:

Prof. Dr. Bruno Augusto Nassif Travençolo
Orientador

Profª. Dra. Leiliane Pereira de Rezende

Profª. Dra. Roberta Barbosa Oliveira

Uberlândia, Brasil
2018

A todos os meus professores, que contribuíram para minha formação durante toda minha vida acadêmica.

Aos meus amigos, que sempre compartilham os momentos felizes e me apoiam nas horas difíceis.

Agradecimentos

Agradeço à Deus, por me conceder sabedoria diante das decisões e pela força para continuar o meu caminho.

Agradeço aos meus pais pela vida e pelo apoio incondicional durante todos esses anos que estive na universidade.

À minha irmã Suélen, pelo grande incentivo ao meu ingresso neste curso de Sistemas de Informação.

Agradeço ao meu orientador Prof. Dr. Bruno Travençolo pela oportunidade a mim concedida e pelas lições ensinadas durante toda a minha graduação e especialmente no decorrer deste trabalho.

Enfim, agradeço a todas as pessoas que participaram dessa etapa tão importante em minha vida.

Resumo

A Comissão de Distribuição de Disciplinas (CDD) da Faculdade de Computação da Universidade Federal de Uberlândia utiliza o Sistema *Online* de Distribuição de Disciplinas (SODD) para realizar a distribuição das disciplinas ofertadas a cada semestre entre seus docentes. O presente trabalho propõe realizar manutenção e a implementação de novos requisitos no SODD. A proposta inicial tem como objetivos a melhoria das funcionalidades já existentes, a correção de falhas eventualmente encontradas e a implementação de novas funcionalidades demandadas pela CDD ou pelos docentes. Ao final do trabalho, utilizando tecnologias e métodos bastante difundidos no desenvolvimento de Aplicações *Web*, as melhorias e correções foram disponibilizadas no sistema, alcançando os objetivos satisfatoriamente e cumprindo o trabalho proposto.

Palavras-chave: distribuição de disciplinas, sistema *online*, atualização de *software*, manutenção de *software*.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

CDD	Comissão de Distribuição de Disciplinas
SODD	Sistema Online de Distribuição de Disciplinas
FACOM	Faculdade de Computação
TCC	Trabalho de Conclusão de Curso
SQL	*Structured Query Language*
HTML	*HyperText Markup Language*
CSS	*Cascading Style Sheets*
JS	JavaScript
JSON	JavaScript *Object Notation*
POO	Programação Orientada a Objetos
MVC	*Model-View-Controller*
REST	*Representational State Transfer*
CPU	*Central Processing Unit*
RAM	*Random-Access Memory*
API	*Application Programming Interface*
HTTP	*Hypertext Transfer Protocol*
XML	*Extensible Markup Language*

Sumário

1 Introdução

A informatização de sistemas de informação permite que o crescente volume de informação que as instituições possuem e geram atualmente seja melhor administrado, além de tornar seus processos mais rápidos, produtivos e seguros, melhorando a eficiência de seus recursos humanos e financeiros.

A Faculdade de Computação (FACOM) da Universidade Federal de Uberlândia realiza, semestralmente, a distribuição das disciplinas que ofertará entre seus docentes. Esse processo fica sob a responsabilidade da Comissão de Distribuição de Disciplinas (CDD), que deve distribuir as disciplinas de acordo com as regras definidas pelo Conselho da FACOM.

De forma simplificada, pode-se dizer que esse processo ocorre em três etapas. Primeiro a FACOM define quais disciplinas serão ofertadas para o semestre, podendo acrescentar novas disciplinas ou retirar disciplinas ofertadas anteriormente. Cada disciplina ofertada possui uma fila de professores que já haviam manifestado interesse em ministrá-la. Em seguida, os professores podem entrar em novas filas ou sair de filas em quais já estejam. Além disso, atribuem uma ordem de prioridade entre essas disciplinas. Por último, a CDD realiza a distribuição das disciplinas, com base nas filas das mesmas, nas prioridades definidas pelos docentes e nas regras de distribuição.

Segundo Locatelli (2015), até o ano de 2015 a distribuição das disciplinas entre os docentes era realizada de forma manual, demandava uma grande quantidade de tempo e era relativamente complexa. A fim de otimizar esse processo, a CDD decidiu criar um sistema informatizado que, além de armazenar as informações sobre as distribuições, também pudesse executar a maior parte da tarefa de forma automatizada. Dessa forma, em 2015, por meio do Trabalho de Conclusão de Curso (TCC) de **??**)locatelli:2015), iniciou-se um projeto para a informatização do sistema existente. Ao fim desse TCC, o Sistema Online de Distribuição de Disciplinas (SODD) foi concebido para que parte do processo de distribuição de disciplina fosse realizado de forma automatizada.

Após a implantação do SODD surgiram novos requisitos para adição de funcionalidades ao sistema. Apesar de parte do processo ter sido automatizada, outras ainda eram feitas pelos integrantes da CDD por meio de consultas e *scripts* em Linguagem de Consulta Estruturada (*Structured Query Language* - SQL), tarefa que não é produtiva e está sujeita a erros durante a execução desses *scripts* e consultas no banco de dados. Além disso, também havia a necessidade da correção de eventuais defeitos que eram percebidos durante a interação dos usuários com o sistema.

Durante os anos seguintes, outros alunos trabalharam em seus TCCs provendo

essa manutenção contínua necessária ao SODD. Silva (2016) trabalhou promovendo melhorias na usabilidade do sistema, enquanto Naves (2016) foi encarregado de criar um módulo administrativo para facilitar as operações de configuração do sistema e um módulo de relatórios para fornecer informações precisas de maneira ágil acerca do processo de distribuição. Oliveira (2017) continuou o desenvolvimento e manutenção do SODD, acrescentando novas funcionalidades e melhorando as já existentes, avaliando os *feedbacks* recebidos dos docentes, além de corrigir os problemas que eventualmente surgiam durante o uso do sistema.

Como o processo de evolução e manutenção de um software deve ser contínuo, o presente TCC tem como objetivo principal continuar o desenvolvimento do sistema, implementando novos requisitos que surjam durante o desenvolvimento do trabalho e provendo a manutenção necessária ao SODD. Buscando tornar o processo de distribuição de disciplinas mais simples para a CDD e proporcionar uma melhor experiência aos professores.

1.1 Evolução de *software*

Segundo Sommerville (2011), o processo de desenvolvimento do *software* não termina após a entrega do sistema, mas deve continuar por todo o período de vida útil do mesmo. Após sua implantação, para que o sistema mantenha-se útil é imprescindível que que se atenda as novas demandas por mudanças no *software*. O *software* pode precisar de mudanças para a correção de erros encontrados, para a sua adaptação às alterações em seu ambiente ou para melhoria de suas características ou de seu desempenho.

A Figura 1 mostra o modelo em espiral do processo de evolução de *software*. No processo ocorrem quatro etapas: especificação, implementação, validação e operação. Essas etapas se repetem de maneira cíclica durante toda a vida do *software*.

Com o SODD não foi diferente, isto é, desde a entrega do sistema, constantemente surgem novos requisitos, seja para aumentar a automatização do processo de distribuição de disciplinas ou para melhor a experiência dos usuários. Além disso, sempre que é relatado um novo defeito no sistema, é necessário que seja desenvolvida uma correção.

1.1.1 Manutenção de software

Manutenção de *software* é o termo usado para o processo de alterações em um sistema após sua entrega, seja para correção de erros, melhorias ou adaptações. Essas alterações são realizadas modificando elementos do sistema ou adicionando novos, quando necessário (SOMMERVILLE, 2011).

Há três tipos de manutenção de *software*:

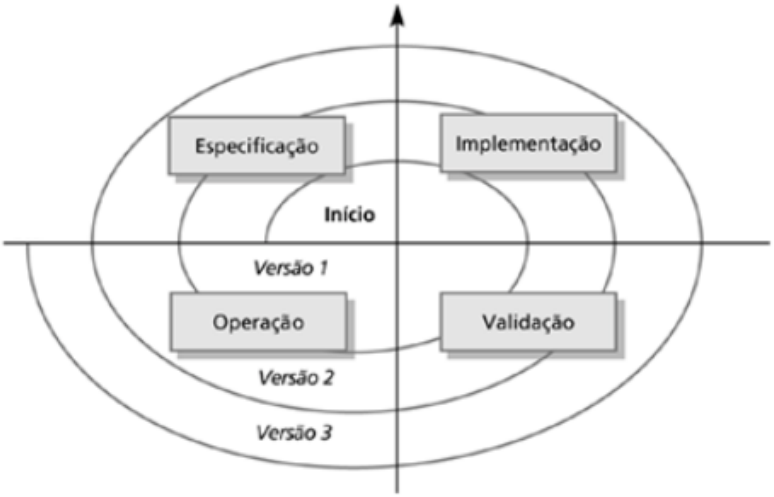


Figura 1 – Modelo em espiral de desenvolvimento e evolução. (Figura adaptada de Sommerville (2011)).

- Correção de defeitos: três tipos de erros podem ser encontrados. Erros de codificação são os mais baratos de se corrigir. Erros de projeto podem resultar na reprogramação de vários elementos do software. E por fim, erros de requisitos, os mais caros para serem corrigidos, pois pode ser necessário uma grande mudança no projeto do sistema;
- Adaptação ambiental: tipo de manutenção necessária quando ocorrem mudanças no ambiente do sistema e o *software* necessita ser adaptado a essas mudanças (e.g. mudança no sistema operacional, mudança em um sistema de apoio ou de *hardware*);
- Adição de funcionalidade: manutenção necessária quando ocorre alteração dos requisitos do sistema devido a mudanças de negócios ou da organização. Normalmente, é necessário um volume maior de esforço dedicado a esse tipo de manutenção em relação aos outros.

De acordo com Sommerville (2011), vários estudos foram feitos observando os relacionamentos entre o desenvolvimento e a manutenção, e entre os diferentes tipos de manutenção. Em geral, essas pesquisas concordam que a manutenção de software utiliza uma parcela maior de recursos de Tecnologia da Informação do que o desenvolvimento. Elas também compartilham que a implementação de novos requisitos consome uma parte maior de recursos do que a correção de *bugs*. A Figura 2 mostra como os esforços de manutenção se distribuem aproximadamente. Apesar desses percentuais variar para cada organização, geralmente, a adição ou modificação de funcionalidades para atender novos requisitos consome mais esforço de manutenção do que o reparo de defeitos e a adaptação de *software*.

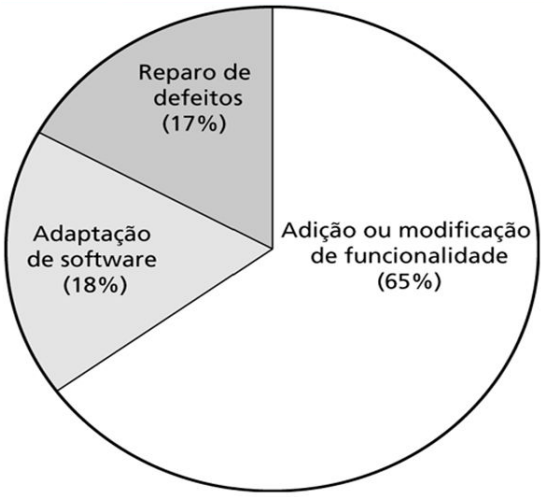


Figura 2 – Distribuição do esforço de manutenção. (Figura adaptada de Sommerville (2011)).

O processo de manutenção do SODD seguiu essa tendência de distribuição dos esforços de manutenção, como podemos verificar no Capítulo 3. A maior parte das alterações realizadas no *software*, durante este trabalho, foram para atender a demanda por novas funcionalidades que surgiram com o uso do sistema, promovendo a evolução do mesmo.

1.2 Objetivos

Nesta seção serão apresentados os objetivos geral e específicos deste trabalho.

1.2.1 Objetivo Geral

O objetivo geral deste projeto é continuar o desenvolvimento e prover a manutenção do SODD, facilitando o trabalho realizado pela CDD e buscando melhor a usabilidade do sistema.

1.2.2 Objetivos Específicos

Tendo o objetivo geral como base, é possível definir os seguintes objetivos específicos:

- Adicionar novas funcionalidades ao sistema, de acordo com as necessidades da CDD.
- Corrigir eventuais falhas do sistema, relatadas pelos usuários ou identificadas pelos desenvolvedores.

- Melhorar usabilidade de funcionalidades existentes, com base no *feedback* recebido dos docentes.
- Implementar no SODD tarefas até então realizadas utilizando *scripts* em SQL.

2 Materiais e Métodos

Neste capítulo serão apresentados o ambiente e as ferramentas utilizados para a manutenção e a implementação dos novos requisitos do SODD durante o desenvolvimento deste trabalho.

2.1 Materiais

Os materiais utilizados para o desenvolvimento deste trabalho foram, em sua maioria, os mesmos utilizados por Locatelli (2015) durante o desenvolvimento da primeira versão do SODD, além de outros que foram incluídos ao longo do processo de evolução do sistema. A Tabela 1, adaptada de Locatelli (2015), detalha esse materiais.

Tabela 1 – Ferramentas utilizadas.

Ferramenta	**Versão**	**Descrição**
Computador usado como servidor	-	Especificações: Processador Intel(R) Core(TM) Quad Q8400 CPU @ 2.66GHz, 4GB de RAM, disco rígido com 300GB de capacidade e sistema operacional Windows 7 Professional 32-bit.
Computador para desenvolvimento	-	Especificações: Processador Intel(R) Core(TM) i7-5500U CPU @ 2.40GHz, 16GB de RAM, disco rígido com capacidade de 1TB e sistema operacional Windows 10 Home 64-bit.
Java (DEITEL; DEITEL, 2016)	1.8.0-151	Linguagem de programação usada no *backend* do SODD.
Apache Tomcat	8.0.26	Servidor de aplicações que facilita a execução de sistemas Java para a *web*.
Eclipse (VOGEL, 2013)	4.6.1	Ambiente de Desenvolvimento Integrado com suporte para a linguagem Java e fácil integração com o Apache Tomcat.

Continuação da Tabela 1

Ferramenta	Versão	Descrição
Hibernate (KONDA, 2014)	4.2.15	O Hibernate é um *framework* que auxilia na utilização de bancos de dados relacionais em conjunto com a linguagem Java, pois permite o mapeamento de entidades presentes no banco de dados em classes da linguagem Java além de tornar consultas e persistência das informações mais simples.
Apache Maven (SIRIWARDENA, 2014)	3.3.3	Ferramenta utilizada no gerenciamento de dependências, tornando a instalação e atualização de bibliotecas e *frameworks* mais ágil e simplificada.
RESTEasy	3.0.4	*Framework* que auxilia na criação de *Web Services RESTful*. É uma implementação da especificação JAX-RS 2.1, que provê uma API Java para Web Services RESTful sobre o protocolo HTTP.
Google Gson (BASSETT, 2015)	2.2.2	Biblioteca Java utilizada para converter objetos do Java em strings no formato JSON e vice-versa.
Bootstrap (SILVA, 2015)	3.3.5	*Framework* para criação de *front-ends* em aplicações *web* utilizando HTML, CSS e JS.
AngularJS (WILLIAMSON, 2015)	1.4.6	AngularJS é um *framework* cridado pela Google em JavaScript para criação de aplicações *web*.

2.2 Métodos

Serão demonstrados nesta seção os conceitos e métodos utilizados na execução deste TCC.

2.2.1 Programação Orientada a Objetos

Dentre os paradigmas de programação existentes, a Programação Orientada a Objetos (POO) é o mais difundido nos dias atuais. Analisando o grupo das oito primeiras linguagens do índice da comunidade de programação TIOBE (2018), *ranking* que lista as cinquenta linguagens de programação mais populares e que quantificam suas popularidades, pode-se perceber que, com exceção da linguagem C, todas são linguagens de POO e juntas somam 75% de popularidade dentro do grupo (Figura 3).

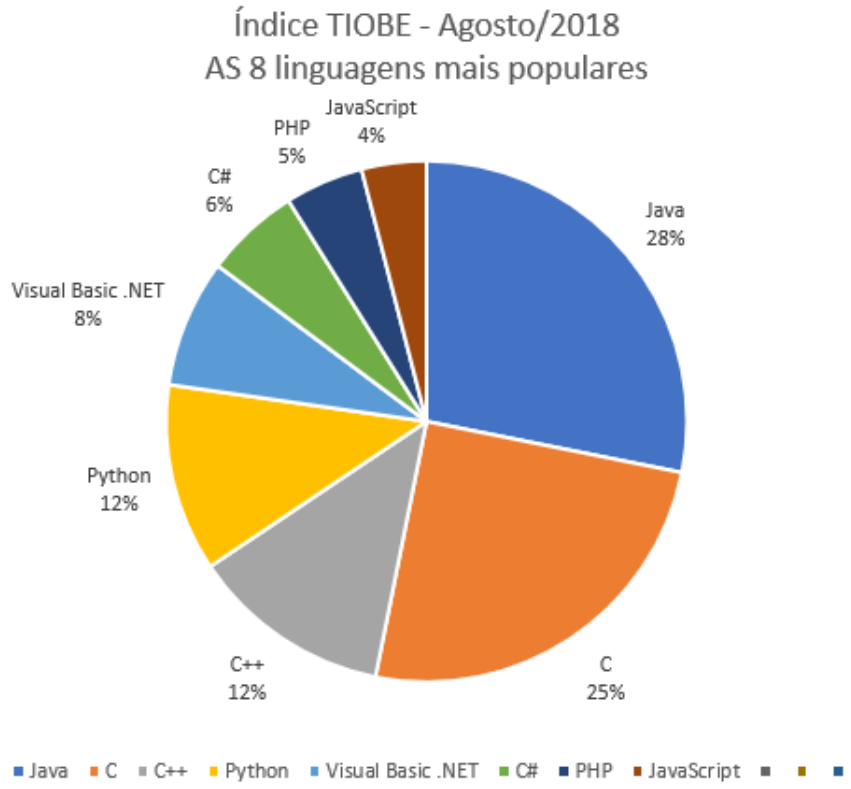


Figura 3 – Gráfico de popularidade das oito primeiras linguagens do índice TIOBE (2018).

A POO propõe que os elementos sejam representados em termos de objeto ou classe. Essa abordagem busca uma aproximação do software em desenvolvimento com o observado no mundo real, tornando a abstração e entendimento dos componentes do sistema mais natural.

Segundo Gasparotto (2014), uma das principais necessidades no desenvolvimento de software é a reutilização de código. Pois, com os sistemas se tornando cada vez maiores e mais complexos, sem a reutilização de código, o tempo de desenvolvimento aumentaria muito. A POO possibilita essa reutilização de código por apresentar cada elemento de maneira clara e, geralmente, sem interdependência. Essa ausência de dependência é o que torna possível que o código seja reutilizado.

2.2.2 Arquitetura MVC

A arquitetura MVC (*Model-View-Controller*) é um padrão de arquitetura que propõe a separação da aplicação em camadas bem definidas: Modelo, Visão e Controle (Figura 4). A separação dessas camadas, que possuem funções específicas e se interconectam, tem o objetivo de facilitar a compreensão e a manutenção do software.

A camada da Visão está relacionada à visualização da aplicação, ou seja, a interface de interação com o usuário. Essa camada deve conter apenas os recursos visuais e

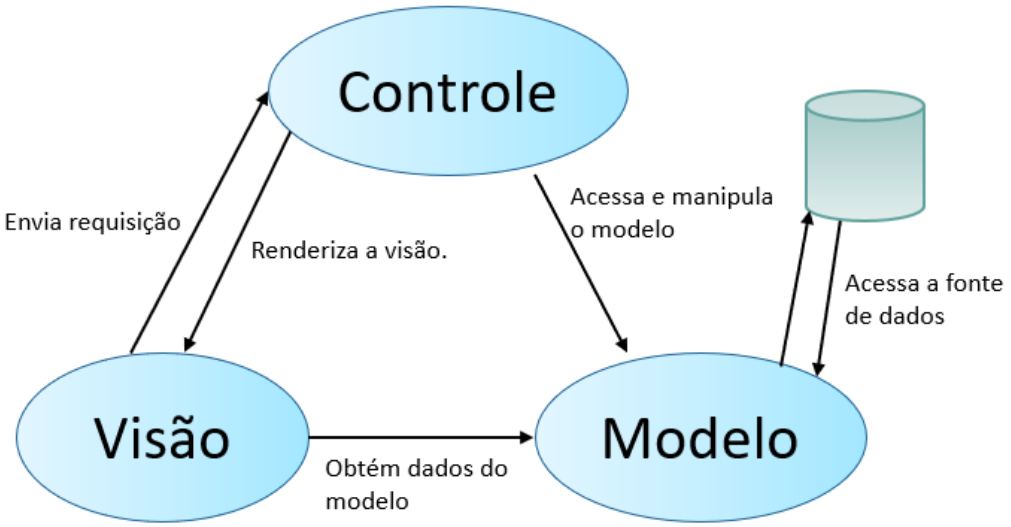


Figura 4 – Componentes da arquitetura MVC e suas interações. (Figura adaptada de ASPArticles (2017)).

de interação, como botões, caixas de texto e de seleção, além de criar uma representação visual do Modelo. A camada Controle tem o papel de intermediar a interação da camada Visão com as regras de negócio, processando a entrada fornecida pelo usuário e transmitindo comandos para a camada Modelo. Finalmente, a camada Modelo é responsável pelas regras de negócio e o acesso aos dados do sistema (CELESTINO, 2014).

A abordagem MVC possibilita que o sistema fique bem estruturado, com cada elemento tendo suas funções bem definidas. Ela cria desacoplamento entre as funções do sistema, permitindo alterações em uma das camadas sem haver interferências em outras. Por exemplo, possibilita realizar a troca da interface gráfica do sistema sem necessidade de alterações nas camadas de Controle ou Modelo.

2.2.3 REST

O SODD realiza a comunicação entre o *back-end* e o *front-end* utilizando os conceitos de Transferência de Estado Representacional (*Representational State Transfer* - REST). O REST é um estilo de arquitetura utilizado para criar padrões entre sistemas na *Web* tornando mais fácil a comunicação entre eles (CODECADEMY).

Utilizando REST, a implementação do cliente e do servidor podem ser feitas de forma totalmente independente, sem que um lado conheça a implementação do outro, contanto que os dois lados saibam o formato das mensagens a serem enviadas para o outro. Com uma interface REST, clientes diferentes podem realizar uma mesma requisição à um sistema final e obter a mesma resposta.

As requisições ao servidor são feitas por meio de mensagens do protocolo HTTP,

normalmente utilizando os verbos GET (obter), POST (postar), PUT (pôr) e DELETE (excluir) de acordo com a operação a ser realizada. O servidor as responde com uma mensagem com o resultado da operação.

Quando necessário, há troca de dados no corpo das mensagens. Neste caso, os formatos comumente utilizados são JSON (JavaScript *Object Notation* - Notação de Objetos JavaScript) e XML (*Extensible Markup Language* - Linguagem Extensível de Marcação Genérica). Pois esses formatos tornam fácil a tarefa de transformar dados recebidos em objetos da linguagem de programação utilizada e vice-versa.

3 Resultados

São mostrados, neste capítulo, os resultados obtidos durante este trabalho. O capítulo está dividido em duas seções. Na primeira, são apresentadas as novas funcionalidades e melhorias inseridas no sistema. Na segunda, são detalhadas as correções dos erros detectados durante este TCC.

3.1 Novas funcionalidades e melhorias

Nesta seção serão detalhadas as novas funcionalidades e melhorias que foram implementadas no SODD.

3.1.1 Visualização de filas de disciplinas e turmas

Durante o período em que o professor está decidindo as prioridades das disciplinas que tem interesse em ministrar, é comum que consulte as filas de tais disciplinas, ou de suas respectivas turmas, para saber sua posição. Nesse momento, também é frequente a necessidade de consultar as filas em que os outros professores estão e suas prioridades. Anteriormente para se fazer essa consulta era necessário utilizar várias telas espalhadas no sistema e fazer a pesquisa utilizando o nome de cada um dos professores que estão na fila.

Ao identificar essa necessidade e perceber que essas consultas eram, de certa forma, trabalhosa para os professores, foram criados *links* nos nomes dos professores presentes na fila (Figura 5). Isso possibilitou que essa consulta fosse realizada de forma muito mais fácil e rápida, pois com apenas um clique o sistema apresenta em uma caixa de diálogo *modal* com as informações das filas (Figura 6).

Figura 5 – Consulta à fila de uma turma com os *links* criados nos nomes dos professores, para consultá-los.

Ivan da Silva Sendin - 2018 / 2

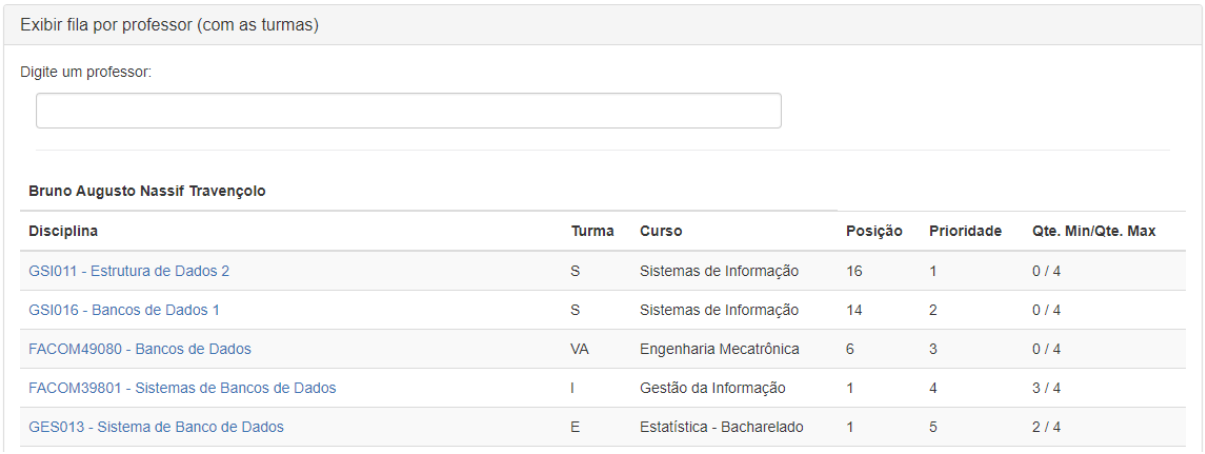
Disciplina	Turma	Curso	Posição	Prioridade	Min/Máx
GBC083 - Segurança da Informação	C	Ciência da Computação	1	1	2 / 4
GSI027 - Otimização	S	Sistemas de Informação	1	2	1 / 4
GBC015 - Introdução à Ciência da Computação	C	Ciência da Computação	3	3	3 / 4
GBC042 - Teoria dos Grafos	C	Ciência da Computação	12	4	0 / 4
GSI035 - Auditoria e Segurança da Informação	S	Sistemas de Informação	5	5	0 / 4
GBC052 - Análise de Algoritmos	C	Ciência da Computação	8	6	0 / 4
GBC065 - Modelagem e Simulação	C	Ciência da Computação	7	7	0 / 4
GBC025 - Programação Lógica	C	Ciência da Computação	12	8	0 / 4
GBC016 - Lógica para Computação	C	Ciência da Computação	14	9	0 / 4
GBC034 - Algoritmos e Estruturas de Dados 2	C	Ciência da Computação	20	10	0 / 4

Primeira Anterior 1 2 3 4 Próxima Última

Figura 6 – Resultado da consulta a um professor após o uso do *link* mostrado na Figura 5.

3.1.2 Visualização de filas de um professor

Quando visualizadas as filas de um professor, ocorre uma situação parecida com aquela citada na seção anterior. Pois consultando as filas em que um professor está escrito é frequente o interesse em ver a situação de uma ou mais dessas filas. Considerando isso, foram implementados *links* similares para cada uma das filas que o professor está presente (Figura 7), possibilitando que as informações sejam apresentadas com um único clique (Figura 8).



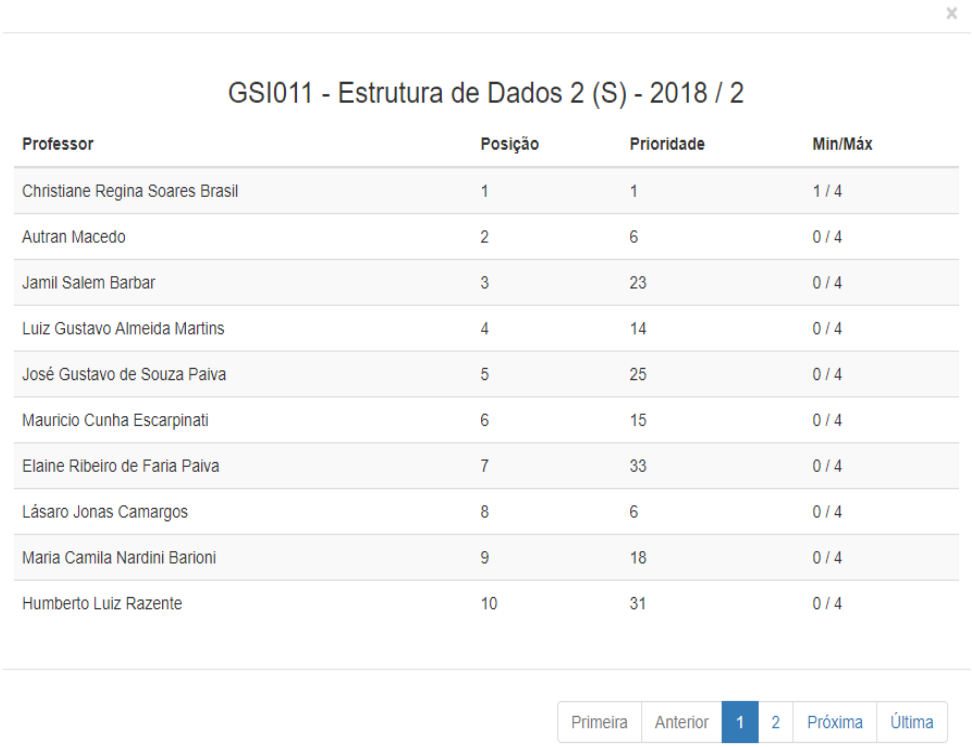
Exibir fila por professor (com as turmas)

Digite um professor:

Bruno Augusto Nassif Travençolo

Disciplina	Turma	Curso	Posição	Prioridade	Qte. Min/Qte. Max
GSI011 - Estrutura de Dados 2	S	Sistemas de Informação	16	1	0 / 4
GSI016 - Bancos de Dados 1	S	Sistemas de Informação	14	2	0 / 4
FACOM49080 - Bancos de Dados	VA	Engenharia Mecatrônica	6	3	0 / 4
FACOM39801 - Sistemas de Bancos de Dados	I	Gestão da Informação	1	4	3 / 4
GES013 - Sistema de Banco de Dados	E	Estatística - Bacharelado	1	5	2 / 4

Figura 7 – Consulta às inscrições de um professor com *links* implementados.



GSI011 - Estrutura de Dados 2 (S) - 2018 / 2

Professor	Posição	Prioridade	Min/Máx
Christiane Regina Soares Brasil	1	1	1 / 4
Autran Macedo	2	6	0 / 4
Jamil Salem Barbar	3	23	0 / 4
Luiz Gustavo Almeida Martins	4	14	0 / 4
José Gustavo de Souza Paiva	5	25	0 / 4
Mauricio Cunha Escarpinati	6	15	0 / 4
Elaine Ribeiro de Faria Paiva	7	33	0 / 4
Lásaro Jonas Camargos	8	6	0 / 4
Maria Camila Nardini Barioni	9	18	0 / 4
Humberto Luiz Razente	10	31	0 / 4

Figura 8 – Resultado da consulta à fila de uma disciplina após o uso do *link* mostrado na Figura 7.

3.1.3 Visualização de horários

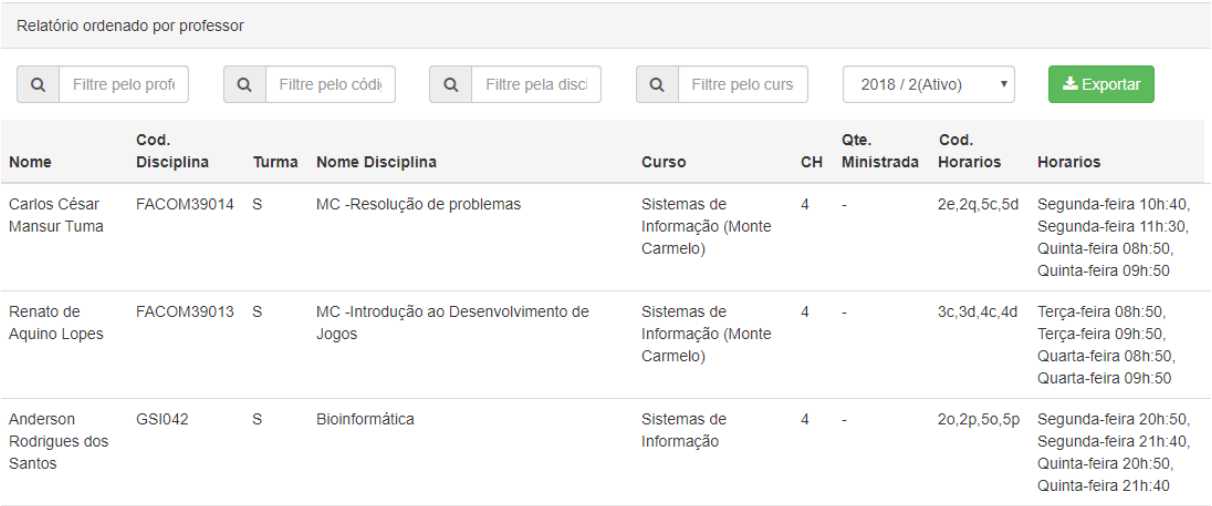
Existe uma página em que é possível visualizar os horários de disciplinas ou de professores na forma de uma tabela, sendo que cada dia da semana é uma coluna e cada horário é uma linha (Figura 9). Alguns professores relataram que às vezes era difícil identificar em qual linha e coluna o mouse estava posicionado. Partindo desse *feedback*, a *interface* foi alterada, criando um realçamento que permite visualizar, de forma fácil e rápida, para qual linha e coluna o ponteiro do mouse está apontado.

	Segunda-feira	Terça-feira	Quarta-feira	Quinta-feira	Sexta-feira	Sabado
07:10 (a)						
08:00 (b)						
08:50 (c)				GBC035(C) -		
09:50 (d)				GBC035(C) -		
10:40 (e)				GBC035(C) -		
11:30 (q)				GBC035(C) -		
13:10 (f)						
14:00 (g)						

Figura 9 – Consulta aos horários de uma disciplina indicando as linhas e colunas em que o mouse está posicionado.

3.1.4 Relatório de distribuição de turmas

Existe um relatório no sistema em que é apresentada a distribuição das turmas e seus respectivos professores (Figura 10). Pensando na melhoria da usabilidade, foram acrescentas duas colunas contendo os horários de cada turma, dispensando, assim, a necessidade de outras consultas caso o usuário queira saber o horário de determinadas turmas.



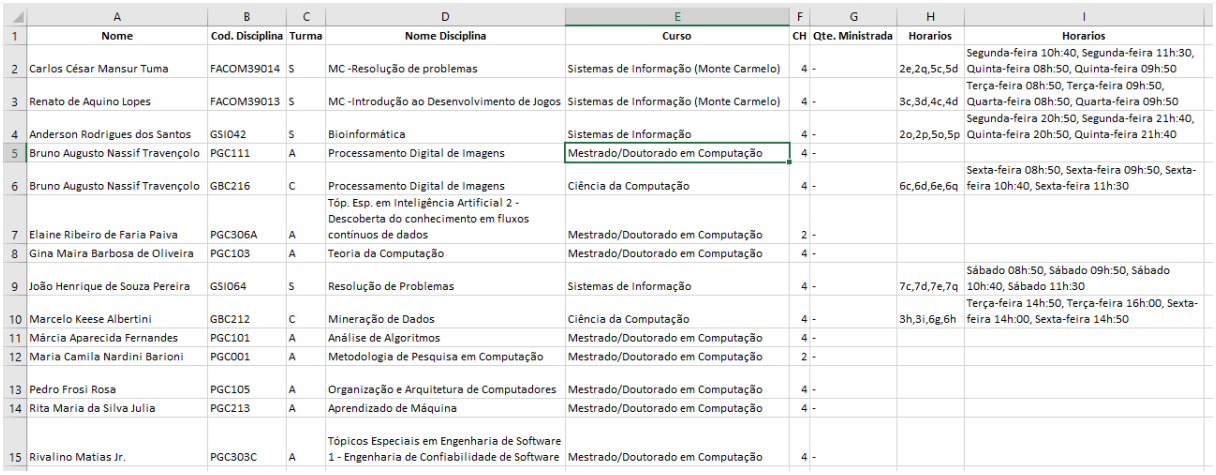
Relatório ordenado por professor

Filtre pelo prof | Filtre pelo códi | Filtre pela disc | Filtre pelo curs | 2018 / 2(Ativo) | Exportar

Nome	Cod. Disciplina	Turma	Nome Disciplina	Curso	CH	Qte. Ministrada	Cod. Horarios	Horarios
Carlos César Mansur Tuma	FACOM39014	S	MC -Resolução de problemas	Sistemas de Informação (Monte Carmelo)	4	-	2e,2q,5c,5d	Segunda-feira 10h:40, Segunda-feira 11h:30, Quinta-feira 08h:50, Quinta-feira 09h:50
Renato de Aquino Lopes	FACOM39013	S	MC -Introdução ao Desenvolvimento de Jogos	Sistemas de Informação (Monte Carmelo)	4	-	3c,3d,4c,4d	Terça-feira 08h:50, Terça-feira 09h:50, Quarta-feira 08h:50, Quarta-feira 09h:50
Anderson Rodrigues dos Santos	GSI042	S	Bioinformática	Sistemas de Informação	4	-	2o,2p,5o,5p	Segunda-feira 20h:50, Segunda-feira 21h:40, Quinta-feira 20h:50, Quinta-feira 21h:40

Figura 10 – Relatório de distribuição de disciplinas, com o acréscimo dos horários das turmas.

Além disso, foi criado um botão (botão “Exportar”, Figura 10) que permite a exportação do relatório no formato de planilha, possibilitando a manipulação dos dados no *Microsoft Excel* ou software similar (Figura 11).



	A	B	C	D	E	F	G	H	I
1	Nome	Cod. Disciplina	Turma	Nome Disciplina	Curso	CH	Qte. Ministrada	Horarios	Horarios
2	Carlos César Mansur Tuma	FACOM39014	S	MC -Resolução de problemas	Sistemas de Informação (Monte Carmelo)	4	-	2e,2q,5c,5d	Segunda-feira 10h:40, Segunda-feira 11h:30, Quinta-feira 08h:50, Quinta-feira 09h:50
3	Renato de Aquino Lopes	FACOM39013	S	MC -Introdução ao Desenvolvimento de Jogos	Sistemas de Informação (Monte Carmelo)	4	-	3c,3d,4c,4d	Terça-feira 08h:50, Terça-feira 09h:50, Quarta-feira 08h:50, Quarta-feira 09h:50
4	Anderson Rodrigues dos Santos	GSI042	S	Bioinformática	Sistemas de Informação	4	-	2o,2p,5o,5p	Segunda-feira 20h:50, Segunda-feira 21h:40, Quinta-feira 20h:50, Quinta-feira 21h:40
5	Bruno Augusto Nassif Travençolo	PGC111	A	Processamento Digital de Imagens	Mestrado/Doutorado em Computação	4	-		
6	Bruno Augusto Nassif Travençolo	GBC216	C	Processamento Digital de Imagens	Ciência da Computação	4	-	6c,6d,6e,6q	Sexta-feira 08h:50, Sexta-feira 09h:50, Sexta-feira 10h:40, Sexta-feira 11h:30
7	Elaine Ribeiro de Faria Paiva	PGC306A	A	Tóp. Esp. em Inteligência Artificial 2 - Descoberta do conhecimento em fluxos contínuos de dados	Mestrado/Doutorado em Computação	2	-		
8	Gina Maira Barbosa de Oliveira	PGC103	A	Teoria da Computação	Mestrado/Doutorado em Computação	4	-		
9	João Henrique de Souza Pereira	GSI064	S	Resolução de Problemas	Sistemas de Informação	4	-	7c,7d,7e,7q	Sábado 08h:50, Sábado 09h:50, Sábado 10h:40, Sábado 11h:30
10	Marcelo Keese Albertini	GBC212	C	Mineração de Dados	Ciência da Computação	4	-	3h,3i,6g,6h	Terça-feira 14h:50, Terça-feira 16h:00, Sexta-feira 14h:00, Sexta-feira 14h:50
11	Márcia Aparecida Fernandes	PGC101	A	Análise de Algoritmos	Mestrado/Doutorado em Computação	4	-		
12	Maria Camila Nardini Barioni	PGC001	A	Metodologia de Pesquisa em Computação	Mestrado/Doutorado em Computação	2	-		
13	Pedro Frosi Rosa	PGC105	A	Organização e Arquitetura de Computadores	Mestrado/Doutorado em Computação	4	-		
14	Rita Maria da Silva Julia	PGC213	A	Aprendizado de Máquina	Mestrado/Doutorado em Computação	4	-		
15	Rivalino Matias Jr.	PGC303C	A	Tópicos Especiais em Engenharia de Software 1 - Engenharia de Confiabilidade de Software	Mestrado/Doutorado em Computação	4	-		

Figura 11 – Relatório aberto no *Microsoft Excel*.

3.1.5 Indicação de processamento em andamento

Algumas funcionalidades do sistema, seja pelo processamento de grande volume de dados ou pela troca de grande quantia de dados pela rede, demandam um tempo acima da média para realizar a transação e entregar o resultado ao usuário. Exemplos disso são o relatório de filas não inscritas, em que o sistema demora alguns segundos para gerar o relatório, e também algumas funcionalidades do módulo administrativo. Levando em conta que houveram algumas ocasiões em que essa demora causou dúvidas aos usuários e ao administrador do sistema, foi implementada uma tela de carregamento para fornecer *feedback* ao usuário nas transações mais demoradas (Figura 12).

Figura 12 – Tela solicitando que o usuário aguarde o fim do processamento.

3.1.6 Duplicação de turmas

O SODD possui uma função em seu módulo administrativo que permite duplicar turmas. O administrador escolhe o semestre de origem e as turmas que devem ser copiadas para o semestre de destino. Anteriormente, por padrão, todas as turmas listadas já eram selecionadas para serem duplicadas, porém quando se desejava duplicar apenas uma ou poucas turmas, o gestor precisava desmarcar uma por uma, todas as outras turmas da lista. Pensando em melhorar a usabilidade dessa funcionalidade foi criado uma *checkbox* que marca ou desmarca todas as turmas listadas, com apenas um clique (Figura 13).

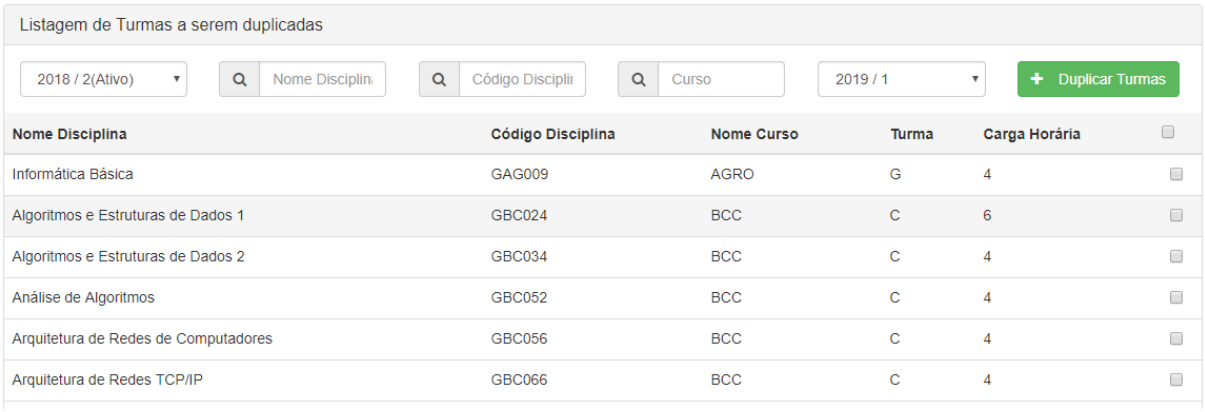


Figura 13 – Página com a listagem das turmas que permite a duplicação das mesmas, com a *checkbox* incluída no cabeçalho da tabela.

3.1.7 Algoritmo de rotação de filas

Após o período em que os professores se inscrevem nas filas das disciplinas pretendidas, é realizado o rotacionamento das filas das disciplinas. Esse processo de rotação atualiza a quantidade ministrada para cada disciplina do professor e em seguida move para o final da fila de cada disciplina os professores que a tenham ministrado a quantidade máxima permitida ou tenham abandonado a disciplina.

Até o início deste trabalho, essa tarefa era realizada por meio de *queries* e *functions* escritas em linguagem SQL. Foi implementado um algoritmo em linguagem Java que pode ser iniciado por meio da interface *web* do SODD e executado no próprio servidor da aplicação. Essa nova abordagem tornou o processo menos suscetível a falhas e deixou sua execução mais simples para o administrador do sistema.

Depois que a execução do algoritmo é concluída, o sistema salva um arquivo de *log* (Figura 14) e permite que o gestor faça o download desse arquivo.

3.1.8 *Status* de professores

Na área para cadastramento e edição dos professores (Figura 15), foi incluída a informação do *status* do professor (e.g., ativo, aposentado, exonerado). Essa mudança foi proposta pelo administrador do sistema, pois se trata de uma informação útil durante a distribuição das disciplinas.

log_rodar_fila.txt

```
Inserindo professores nas filas em que estão ministrando disciplinas.

Nenhum professor a ser inserido

Rodando filas referente as mudanças de 2016-02 para 2017-01.
Isso é feito somente após a distriuição final de 2017-01.
Note que os casos 4/4 de 2017-01 ainda não são rodados pois ainda ocorrerão as inserções em filas.
A inserção deve vir antes de rodar a fila, para garantir que quem ministrou 4/4 vá para o final da fila.

Fila rodada: GBC084       Programação para Internet(4/4) prof:Sílvio Bacalá Júnior
Fila rodada: GSI536       MC -Auditoria e Segurança da Informação(4/4) prof:Murillo Guimarães Carneiro
Fila rodada: GSI010       Programação Lógica(4/4) prof:Anderson Rodrigues dos Santos
Fila rodada: GGI024       Projeto e Desenvolvimento de Software(4/4) prof:Maria Adriana Vidigal de Lima
Fila rodada: GBC034       Algoritmos e Estruturas de Dados 2(4/4) prof:Luiz Gustavo Almeida Martins
Fila rodada: GBC035       Programação Orientada a Objetos 1(4/4) prof:Marcelo de Almeida Maia

Alterando a quantidade ministrada


Identificando abandonos (que vai para o final da fila) ou redução da carga/afastamentos/dobras em que o
docente não vai para o final da fila.

Abandono disciplina: GSI526        - MC -Engenharia de Software - João Batista Simão
```

Figura 14 – Exemplo de arquivo de *log*.

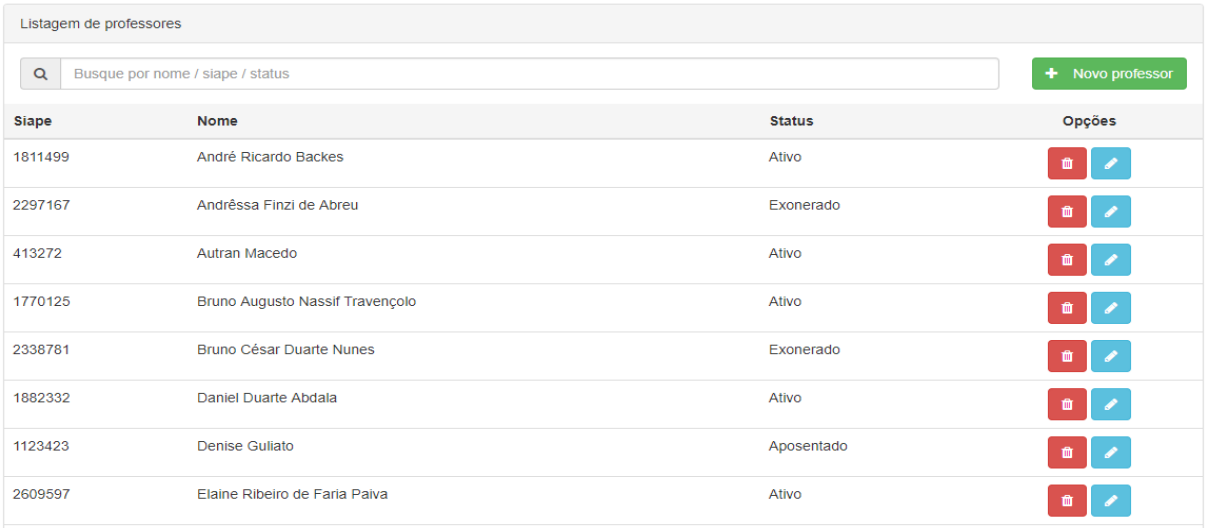


Figura 15 – Cadastramento e edição de professores com a informação de *status* incluída.

3.1.9 Cadastramento de novo semestre

Anteriormente, ao criar um novo semestre (Figura 16), era necessário que o administrador informasse o ano e o semestre manualmente. Foi implementado um algoritmo que determina o próximo semestre a ser criado, com base nos existentes no banco de dados e o sugere ao usuário, porém deixando-o livre para escolher um outro semestre, se assim desejar. Com essa alteração espera-se reduzir as chances de falha humana durante o procedimento.

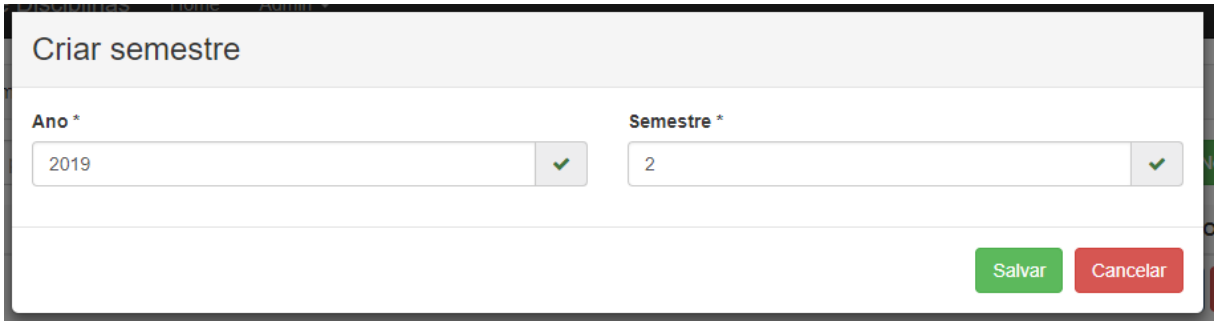


Figura 16 – Cadastramento de um novo semestre com a sugestão do próximo semestre.

3.2 Correção de erros

Serão mostradas nesta seção os erros encontrados no sistema e as correções realizadas durante este trabalho.

3.2.1 Remoção de turmas

Ao acessar a página de gerenciamento de turmas (Figura 17), são carregadas todas as turmas existentes para o semestre selecionado, em formato de tabela, e o sistema permite a criação, edição e remoção das turmas. Quando é realizada a remoção de uma turma, o sistema deve solicitar atualizar a tabela com as turmas. Essa tabela é a representação visual de um vetor que armazena as turmas.

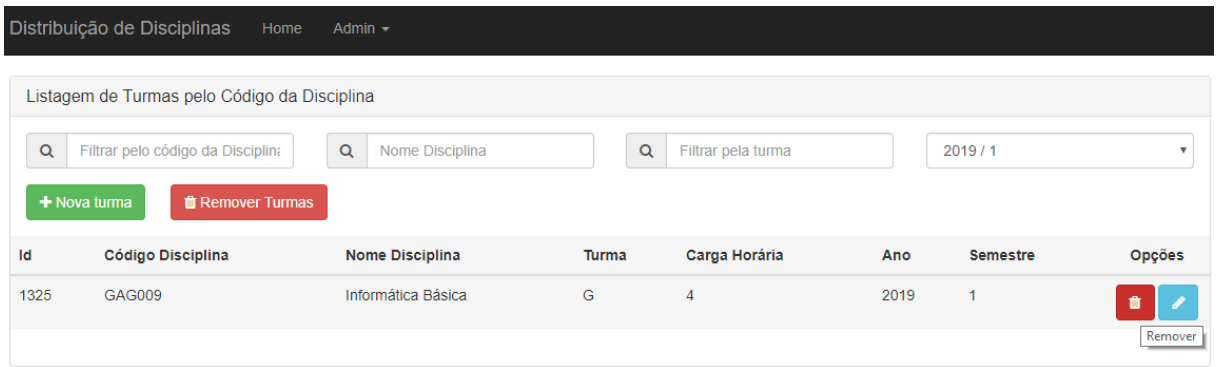


Figura 17 – Página de gerenciamento de turmas, com uma única turma no semestre.

Quando uma turma era excluída, caso fosse a única do semestre, ela era removida do banco de dados, mas a tabela não era atualizada e a turma permanecia visível. Foi verificado que o problema ocorria por não haver uma tratativa para os casos em que o vetor que armazenava as turmas se tornasse vazio. O problema foi corrigido incluindo uma condição extra na estrutura condicional responsável pela atualização dessa tabela (Figura 18).

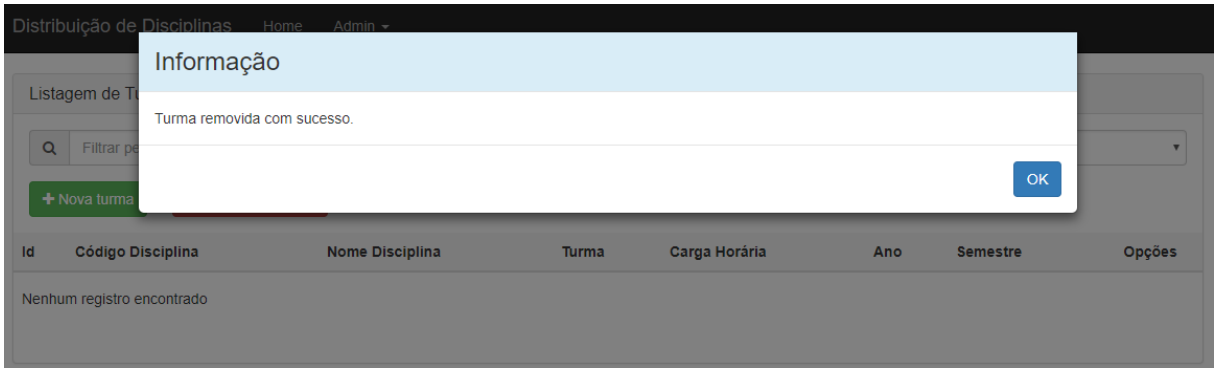


Figura 18 – Resultado da remoção da turma com a tabela atualizada.

3.2.2 Consulta aos horários de disciplinas

Houve um relato de que a página de consulta aos horários das disciplinas (Figura 9), não estava apresentando os horários de certas disciplinas. Fazendo um busca pela causa do erro, foi identificado que o erro ocorria apenas com disciplinas que possuíam hífen em seus nomes e era causado por um trecho de código que fazia o *parsing* de um conjunto de informações. O problema foi resolvido alterando a forma como esse *parsing* estava sendo feito.

3.2.3 Gerenciamento de prioridades

Ao acessar a página de gerenciamento de prioridades, no módulo administrativo, é possível consultar e alterar as prioridades das turmas que os professores estão na fila. Pesquisando pelo nome de um professor, é criada uma lista com cada uma das turmas que está na fila, a qual permite arrastar as turmas para cima e para baixo para definir suas prioridades. Porém, quando as prioridades eram salvas, ao pesquisar pelo professor novamente, se houvessem disciplinas com mais de uma turma, as turmas eram apresentadas sempre uma seguida da outra, independente da ordem que fossem salvas.

Procurando pela causa do erro, foi confirmado que as prioridades estavam sendo salvas corretamente no banco de dados, então a busca pelo erro voltou-se ao processo de criação da lista, onde a falha foi constatada. As informações estavam sendo carregadas da tabela errada do banco de dados, pois havia ocorrido uma reorganização das tabelas recentemente, e essa funcionalidade do sistema não havia sido adaptada para fazer uso da nova tabela.

Para a correção do erro, foi modificado o trecho de código responsável pelo carregamento da lista, que passou a fazer as consultas utilizando a tabela correta. O resultado da correção é demonstrado na Figura 19, em que o professor está inscrito em duas turmas diferentes das disciplinas GSI002 e GBC014, e cada turma aparece em sua ordem de prioridade correta.

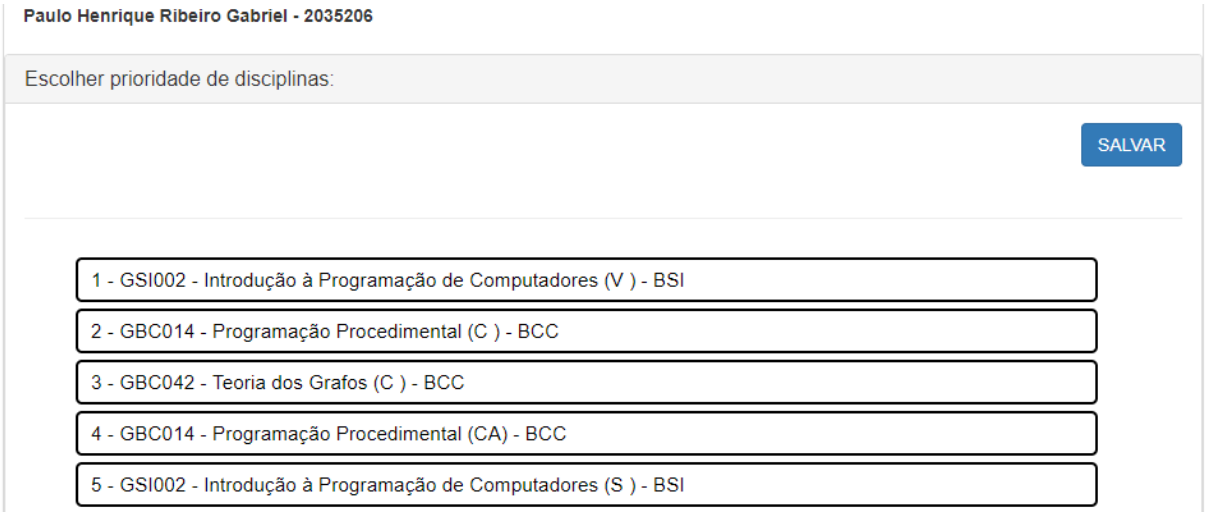


Figura 19 – Página de gerenciamento de prioridades.

4 Conclusão

A automação de processos com o auxílio de sistemas informatizados traz grandes ganhos às organizações. Ela propicia ganhos em termos de agilidade, eficiência, segurança e experiência dos usuários.

No decorrer do projeto, desde as primeiras melhorias implementadas no sistema, o coordenador do trabalho tem recebido *feedbacks* positivos sobre a melhoria na usabilidade do sistema por parte dos docentes. Como atual administrador do sistema, ele fez comentários positivos sobre as melhorias no módulo administrativo que tornaram algumas das tarefas realizadas por ele mais rápidas, simples e seguras.

Utilizando tecnologias e métodos bastante difundidos nos dias atuais, foi possível manter o bom funcionamento do sistema e promover a evolução necessária para atender as necessidades dos usuários e da administração do sistema.

Com a finalização do desenvolvimento, foi possível verificar que novas funcionalidades demandadas foram implementadas e diversas receberam melhorias para fornecer uma experiência melhor aos usuários (vide Seção 3.1). Também pôde-se constatar que as falhas encontradas no sistema, muitas não relatadas aqui por serem demasiadas simples e também para evitar um texto muito longo, foram corrigidas (vide Seção 3.2). Considerando as tarefas realizadas, foram alcançados os objetivos estabelecidos inicialmente, cumprindo a proposta deste trabalho de conclusão de curso.

4.1 Trabalhos futuros

Continuarão surgindo, tanto no módulo do usuário, quanto no módulo administrativo, novas demandas por novas funcionalidades, por melhorias nas existentes e por correção de erros eventualmente encontrados. Como o processo de manutenção de *software* deve ser cíclico e constante durante toda a sua vida útil, é imprescindível que sejam realizados trabalhos de manutenção dando continuidade a este, assim como este deu continuidade aos trabalhos anteriores, para que o SODD mantenha-se útil para a CDD e para os docentes da FACOM.

Referências

ASPARTICLES. *ASP.NET MVC architecture overview (model, view, controller)*. 2017. Disponível em: <http://www.asparticles.com/2017/01/aspnet-mvc-architecture-overview-model-view-controller.html>. Acesso em: 25/07/2018. Citado 2 vezes nas páginas 6 e 18.

BASSETT, L. *Introdução ao JSON*. 1. ed. São Paulo: Novatec, 2015. Citado na página 16.

CELESTINO, A. L. *O conceito e as dúvidas sobre o MVC*. 2014. Disponível em: <https://www.profissionaisti.com.br/2014/10/o-conceito-e-as-duvidas-sobre-o-mvc/>. Acesso em: 24/07/2018. Citado na página 18.

CODECADEMY. *What is REST?* Disponível em: <https://www.codecademy.com/articles/what-is-rest>. Acesso em: 26/07/2018. Citado na página 18.

DEITEL, P.; DEITEL, H. *Java - Como programar*. 10. ed. São Paulo: Pearson, 2016. Citado na página 15.

GASPAROTTO, H. M. *Os 4 pilares da Programação Orientada a Objetos*. 2014. Disponível em: <https://www.devmedia.com.br/os-4-pilares-da-programacao-orientada-a-objetos/9264>. Acesso em: 22/07/2018. Citado na página 17.

KONDA, M. *Introdução ao Hibernate*. 1. ed. São Paulo: Novatec, 2014. Citado na página 16.

LOCATELLI, J. A. *Sistema Online para Distribuição de Disciplinas*. Monografia (Graduação) — Faculdade de Computação, Universidade Federal de Uberlândia, Uberlândia, 2015. Citado 2 vezes nas páginas 10 e 15.

NAVES, P. E. d. P. *Novas Funcionalidades para o Sistema Online de Distribuição de Disciplinas entre docentes*. Monografia (Graduação) — Faculdade de Computação, Universidade Federal de Uberlândia, Uberlândia, 2016. Citado na página 11.

OLIVEIRA, A. S. d. *Novas Funcionalidades e Manutenção no Sistema Online de Distribuição de Disciplinas*. Monografia (Graduação) — Faculdade de Computação, Universidade Federal de Uberlândia, Uberlândia, 2017. Citado na página 11.

SILVA, M. S. *Bootstrap 3.3.5*. 1. ed. São Paulo: Novatec, 2015. Citado na página 16.

SILVA, S. S. d. *Implementação de Novos Requisitos para o Sistema Online para Distribuição de Disciplinas*. Monografia (Graduação) — Faculdade de Computação, Universidade Federal de Uberlândia, Uberlândia, 2016. Citado na página 11.

SIRIWARDENA, P. *Mastering Apache Maven 3*. 1. ed. Birmingham, Reino Unido: Packt Publishing, 2014. Citado na página 16.

SOMMERVILLE, I. *Engenharia de Software*. 9. ed. São Paulo: Pearson Education, 2011. Citado 4 vezes nas páginas 6, 11, 12 e 13.

TIOBE. *TIOBE Index for August 2018*. 2018. Disponível em: <https://www.tiobe.com/tiobe-index/>. Acesso em: 02/08/2018. Citado 3 vezes nas páginas 6, 16 e 17.

VOGEL, L. *Eclipse IDE*. 3. ed. [S.l.]: Lars Vogel, 2013. Citado na página 15.

WILLIAMSON, K. *Introdução ao AngularJS*. 1. ed. São Paulo: Novatec, 2015. Citado na página 16.